INTERROS des LYCÉES
Collection dirigée par Éric MAURETTE

numérique et sciences informatiques

1re

Stéphane PASQUET

Nathan

Remerciements

L'auteur et l'équipe de Prepamath Édition tiennent à remercier Bruno Mermet et David Guénée pour l'aide précieuse qu'ils ont apportée à cet ouvrage.

Illustrations : Laurine Moreau, Mickaël Cellier
Coordination éditoriale : François Déliac
Conception couverture : Simon Geliot

Votre avis nous intéresse : `contact@prepamath.fr`

 ISBN 9782095050177

Avant-propos

Le programme de l'enseignement de spécialité Numérique et Sciences Informatiques (NSI) entré en vigueur à la rentrée 2019 est conçu autour des thèmes suivants :

- comprendre les bases de l'informatique : historique, méthodes de codage d'un nombre, d'un caractère, d'un fichier, types de variables ;
- aborder et approfondir les connaissances en programmation en utilisant comme langage de référence Python (version 3) ;
- comprendre ce qu'est une interface entre l'humain et la machine (IHM) sur le web ;
- aborder les bases des architectures matérielles et du fonctionnement des systèmes d'exploitation ;
- comprendre les similarités entre plusieurs langages ;
- optimiser un algorithme.

La majeure partie de l'ouvrage concernant la programmation, nous avons souhaité proposer un grand nombre de programmes variés, de manière à pouvoir apprendre par des exemples concrets.

Les programmes proposés dans cet ouvrage sont téléchargeables gratuitement sur ordinateur, smartphone et tablette. Ceci vous évitera de perdre du temps à retaper vous-même les programmes, et vous permettra de les compiler chez vous, les tester, et même les modifier de manière à parfaitement assimiler leur fonctionnement.

Dans la mesure où Python est le langage de référence, nous vous conseillons avant tout de l'installer sur votre machine en vous rendant sur le site suivant :

https://www.python.org/downloads/

Dans ce livre, les méthodes les plus utiles vous sont présentées ; mais pour avoir une liste exhaustive des possibilités qu'offre Python 3, vous pouvez également aller sur le site suivant pour prendre connaissance de la documentation :

https://docs.python.org/fr/3/

Même si cette spécialité s'appuie sur une base théorique, nous insistons sur l'importance fondamentale de la pratique : chaque élève doit être capable de prendre des initiatives pour construire ses propres programmes, se poser des questions pour les faire évoluer, et faire ses propres recherches sur les meilleures manières de programmer. Les projets proposés par l'enseignant au lycée seront de bons tremplins pour atteindre ces objectifs.

Nous vous souhaitons une bonne lecture et un bon apprentissage dans cette matière passionnante.

L'auteur.

Table des matières

Chap 1 Encodages 1
- Cours 1
- Interros 14
- Corrigés 18

Chap 2 Types et valeurs de base 25
- Cours 25
- Interros 45
- Corrigés 59

Chap 3 Types construits 79
- Cours 79
- Interros 102
- Corrigés 115

Chap 4 Langages et programmation 133
- Cours 133
- Interros 144
- Corrigés 151

Chap 5 Interactions humain-machine sur le Web 157
- Cours 157
- Interros 174
- Corrigés 179

Chap 6 Architectures matérielles et systèmes d'exploitation 187
- Cours 187
- Interros 201
- Corrigés 206

Chap 7 Algorithmique 211
- Cours 211
- Interros 226
- Corrigés 233

Annexe Liste des vidéos 247

Méthodes de travail

Tous les conseils qui suivent sont ceux utilisés par un grand nombre de majors (sortis premiers) de Polytechnique ou de l'ÉNA, par des professionnels de l'organisation et sont également recommandés par de nombreux professeurs. Pour en savoir plus sur le sujet, nous vous conseillons le livre « **Comment travailler plus efficacement** », par F. Déliac, U. Hadrien, E. Matrullo et E. Maurette, aux Éditions Prepamath.

Faire des « feed-back »

Le « feed-back » est le conseil le plus important et le plus utilisé par ceux qui réussissent brillamment leurs études. Il consiste à contrôler systématiquement, sans s'aider de notes, ce que l'on vient d'apprendre (exercices et cours). Ce contrôle peut se faire mentalement, oralement ou par écrit.

- Dans les transports, essayez de vous rappeler mentalement, et sans vous aider de vos notes, le cours et les exercices vus le matin en classe (feed-back mental).
- Après avoir relu votre cours le soir, essayez de retrouver par écrit les principaux paragraphes et démonstrations sans regarder votre leçon (feed-back écrit).
- Après avoir résolu un problème, prenez 5 minutes pour contrôler par écrit que vous vous rappelez clairement l'énoncé ainsi que la démarche de résolution (feed-back écrit).
- Expliquez à des amis la leçon que vous venez d'apprendre ou l'exercice que vous venez de résoudre : c'est un excellent feed-back oral. Choisissez le type de « feed-back » qui vous convient le mieux et faites-en le plus régulièrement possible (après chaque cours et chaque série d'exercices). Pour être efficace, un « feed-back » doit se faire sans l'aide de vos notes. Ainsi, faire des fiches de résumés de cours à partir de vos cahiers ouverts ne constitue nullement un « feed-back ».

Miser sur la qualité

De nombreux témoignages démontrent que pour obtenir de bons résultats, il est préférable de faire un nombre limité d'exercices, mais plus approfondis, que d'en survoler une grande quantité de piètre qualité. Une tendance très répandue consiste à abattre une grande quantité d'exercices, à la chaîne, mais superficiellement, en espérant que le jour du contrôle, l'on aura déjà vu ce type de problème et que l'on saura s'en souvenir. Cette méthode est absolument inefficace car la seule manière de se souvenir d'un exercice de mathématiques ou de physique, c'est de l'avoir parfaitement compris et assimilé.

Ainsi :

- À la fin d'un problème, prenez 5 à 10 minutes pour essayer de trouver un moyen de le généraliser ou de le compliquer (c'est ce que font souvent les professeurs pour concevoir leurs contrôles écrits) ; trouvez ce que cela pourrait changer dans la solution.
- Prenez également l'habitude, après chaque exercice, de faire un « feed-back » en faisant ressortir la démarche générale et en tissant des liens avec le cours. Bref, il ne faut pas vous contenter de résoudre l'exercice, mais il vous faut lui apporter de la valeur ajoutée et vous interroger sur son contenu.
- Idem pour le cours. Ne vous contentez pas de le parcourir de manière passive. Il vous faut avoir la rigueur d'effacer toutes les zones d'ombre. Pour chaque théorème, il faut vous demander quels types d'exercices son utilisation permettra de résoudre.

Travailler par « couches successives »

Cette méthode, très utile pour les étudiants préparant des examens ou des révisions, peut également être utilisée dès le lycée.

On observe que pour apprendre un gros volume de cours, rien n'est plus inefficace que de l'attaquer de front, de manière linéaire. La bonne manière consiste à d'abord survoler l'ensemble, en ne retenant que la structure, c'est-à-dire les grands titres, ainsi que les noms des paragraphes (première couche, étape devant durer 5 minutes). Dans l'étape suivante (deuxième couche, d'une durée de 10 minutes), on reprend son cours du début en retenant cette fois également les théorèmes et résultats importants. Après cette deuxième couche, on a déjà une idée claire de la structure de l'ensemble du cours. On peut alors aborder la dernière étape (troisième couche) : on reprend son cours au début pour, cette fois-ci, l'étudier en profondeur en apprenant le détail des démonstrations.

Il est à noter que cette méthode peut être également appliquée avec succès à des matières littéraires, ainsi qu'aux révisions du bac de français. Par exemple :

- Pour la préparation d'un contrôle, on commencera par passer en revue rapidement l'ensemble du cours et des exercices du chapitre précédemment étudiés, avant de les réviser en détail. Ainsi aura-t-on développé une compréhension synthétique et claire.
- De même, avant d'aborder un problème volumineux (tel qu'un contrôle écrit), il est préférable de survoler l'ensemble du problème avant de l'attaquer.

Travailler sa rapidité

Pour acquérir de la rapidité, 3 voies sont possibles :

- Prenez l'habitude, en travaillant chez vous, de vous concentrer sur une seule chose à la fois, c'est-à-dire ne pas attaquer un problème ou une dissertation, en rêvassant à ce que vous pourriez trouver à manger dans le réfrigérateur ou en écoutant de la musique.
- Prenez l'habitude de travailler chez vous dans les mêmes conditions qu'en devoirs surveillés. Le minutage de chacun des exercices de ce livre est fait en ce sens. Cependant, cela ne devrait pas, une fois la résolution faite, vous empêcher d'y réfléchir plus calmement afin de vérifier la bonne assimilation du problème. Si les seuls moments où vous vous pressez sont les contrôles écrits, vous ne deviendrez jamais rapide.
- Essayez de contenir tout votre travail à la maison dans une plage horaire serrée. Engagez-vous, par exemple, à travailler chez vous tous les jours entre 18 h et 20 h et efforcez-vous de ne jamais déborder (quelle que soit votre charge de travail). En effet, si l'on ne se donne pas de limite de temps pour accomplir un travail, l'on a naturellement tendance à le laisser traîner en longueur et à rêvasser. L'étroitesse de la plage horaire vous obligera à ne pas vous endormir et à devenir efficace.


ECE
ÉCOLE D'INGÉNIEURS
ENGINEERING SCHOOL
Tu prévois quoi après ton Bac ?
Rejoins l'ECE :
l'école d'ingénieurs qui rend l'Intelligence Artificielle indispensable
6 Campus Paris - Lyon - Bordeaux Rennes - Marseille - Toulouse
3 Programmes Bachelor - Ingénieur - MSc
Inscris-toi à nos prochains événements
Retrouvez-nous sur

Chapitre 1

Encodages

Plan du chapitre

1. Historique de l'informatique
2. Les encodages
3. Les bases

Exercice type — Lycée Raoul Follereau, Nevers

1 Sachant que le binaire 1000001 correspond à la lettre « A », quel est le binaire correspondant à la lettre « B » ?

2 Convertir $\overline{75}^{10}$ en binaire.

3 Convertir $\overline{2001}^{10}$ en hexadécimal.

4 Convertir $\overline{ABC}^{16}$ en binaire.

Voir corrigé page 11

1 Historique de l'informatique

La notion d'informatique est née il y a fort longtemps. Il y a eu beaucoup d'événements qui nous ont menés, petit à petit, à l'informatique telle que nous la connaissons aujourd'hui. En voici, dans un premier temps, un résumé :

- −3000 : l'empereur chinois Fou-Hi adopte comme symbole un octogone contenant les huit premiers nombres sous forme binaire.
- −500 : apparition de l'*abaque* et du *boulier*, premiers outils de calculs.
- −300 : le philosophe Aristote définit la notion de *logique*, nécessaire à l'informatique contemporaine.
- 820 : travaux du mathématicien arabe Al-Khwârizmî.
- 1000 : le zéro est inventé en Inde, et importé en occident au Moyen-Âge. Il ne sera définitivement accepté qu'au XIVe siècle.

- 1600 : l'écossais John Napier invente les logarithmes, qui permettent de ramener les multiplications et les divisions à des additions et des soustractions.
  À la même époque, l'allemand Wilhelm Schickard invente ce qu'il appelle une *horloge calculante*, pouvant réaliser des additions, soustractions, multiplications et mémorisations des résultats.
- 1697 : introduction du binaire en Europe par le mathématicien Gottfried Leibniz.
- 1840 : Augusta Ada Lovelace pose le principe d'une machine à calculer. Trois années plus tard, elle définit la théorie de la programmation.
- 1854 : George Boole définit la théorie de la logique binaire.

Développons certains points de cette chronologie.

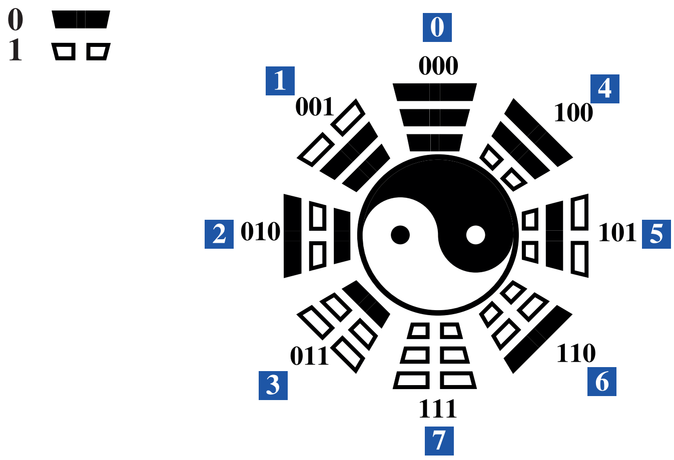
1.1 L'octogone de l'empereur Fou-Hi

Retrouvez plus d'explications sur l'octogone en vidéo.

La notation binaire est plus ancienne que l'on pourrait l'imaginer... En -3000, l'empereur chinois Fou Hi créé un symbole magique que l'on peut considérer comme la première expression d'un codage en binaire, sous forme d'un octogone comportant huit *trigrammes*.

Chaque trigramme est une association de trois signes qui peuvent être soit une bande pleine noire, soit une bande creuse coupée en deux.

Si une bande noire représente 0 et une bande creuse coupée représente 1, on peut compter de 0 à 7 en binaire avec les huit trigrammes, ainsi que le montre la figure suivante.



1.2 Aristote et la logique

La logique *aristotélicienne* sera essentielle pour clarifier tout raisonnement déductif, et l'informatique nécessite de tels raisonnements.

Aristote introduit les *syllogismes*, raisonnements logiques mettant en relation trois propositions. Les deux premières sont appelées *prémisses* et la dernière, obtenue à partir des deux précédentes, est appelée *conclusion*.

Définition 1 : syllogisme

Un *syllogisme* est un raisonnement où deux prémisses permettent d'émettre une conclusion.

Exemples

- « Tous les hommes sont mortels, or Socrate est un homme donc Socrate est mortel ». Ce syllogisme est dû à Aristote lui-même.
  Ici, le premier prémisse est : « tous les hommes sont mortels » ; c'est un fait avéré.
  Le second prémisse est : « Socrate est un homme » ; c'est aussi un fait avéré, qui n'a pas besoin d'être prouvé.
  La conclusion est : « Socrate est mortel » ; des deux faits avérés cités précédemment, on peut conclure ceci.
- Toute planète de notre système solaire possède un rayon supérieur à 2 000 kilomètres. Mercure est une planète de notre système solaire. Donc Mercure a un rayon supérieur à 2 000 km.

ATTENTION

Il existe des syllogismes menant à une conclusion déroutante. Le plus connu est le suivant :
« Tout ce qui est rare est cher. Or, un cheval bon marché est rare. Donc un cheval bon marché est cher. »
On appelle cela un *sophisme*.

1.3 Al-Khwârizmî

Al-Khwârizmî, de son vrai nom *Muhammad Ibn Mūsā al-Khuwārizmī* était un savant perse, né dans les années 780. Son nom latinisé en *Algoritmi* est à l'origine du mot *algorithme*.

Il est l'auteur d'un ouvrage mathématique, *Kitābu al-mukhtasar fī hisābi al-jabr waal-muqābalah*, considéré comme le premier ouvrage d'algèbre.
Dans ce titre, on distingue la partie « al-jabr », qui désigne une opération, et qui est à l'origine du mot « algèbre ».

COURS
INTERROS
CORRIGÉS

COURS

1.4 L'invention du zéro

Il est sûrement surprenant de savoir que le chiffre « 0 » fut inventé après les chiffres 1, 2, etc., mais c'est le cas.

Avant son invention, les Indiens n'avaient pas l'utilité de ce nombre (le zéro en tant que chiffre existait déjà depuis l'époque mésopotamienne, mais n'avait pas d'utilité dans les calculs).

C'est le mathématicien et astronome indien Brahmagupta qui l'introduisit en tant que nombre dans son ouvrage *Brâhma Siddhânta*. Le mot fut traduit ensuite en arabe par le mot « sifr », qui désignait le vide, qui donna plus tard le mot « chiffre ».

Le mot « zéro », quant à lui, vient de l'italien *zephiro*, traduction de *sifr* dans cette langue.

Le zéro fut ensuite introduit en Europe par le mathématicien italien Leonardo Fibonacci, et simplifia considérablement les calculs (jusque-là faits à l'aide d'abaques, donc peu pratiques).

Quant au symbole « 0 », il semblerait provenir de la représentation (initialement un cercle) de la voûte céleste.

1.5 Le binaire et Leibniz

Gottfried Leibniz, personnage allemand aux multiples talents (mathématiques, philosophie, diplomatie, jurisme, philologie, ...), était passionné par la *Dyadique*, c'est-à-dire par tout ce qui se rapporte à la réunion de deux principes philosophiques qui se complètent réciproquement.

S'interrogeant sur l'utilité d'un système connu des chinois, qui peut se comparer à un système binaire, il expose ses idées à l'Académie des sciences de Paris, et celles-ci furent publiées dans un ouvrage intitulé *Explication de l'arithmétique binaire avec des remarques sur son utilité et sur le sens qu'elle donne des anciennes figures chinoises de Fou-Hi*.

1.6 Augusta Ada Lovelace

Augusta Ada Lovelace était la fille du poète romantique Lord George Byron et d'une mathématicienne et féministe, épouse de William King (futur comte de Lovelace). Elle est à l'origine du « Principe des machines à calculer ». Un langage de programmation porte son prénom en mémoire de ses travaux.

Pour elle, une machine à calculer doit comporter :

- un dispositif permettant d'introduire les données numériques (cartes perforées, roues dentées...) ;
- une mémoire pour conserver les valeurs numériques entrées ;

- une unité de commande grâce à laquelle l'utilisateur va indiquer à la machine les tâches à effectuer ;
- un « moulin » chargé d'effectuer les calculs ;
- un dispositif permettant de restituer les résultats (imprimante, écran...) ;

Ces principes seront, un siècle plus tard, à la base des premiers ordinateurs.

Elle définit ensuite le principe d'itérations successives dans l'exécution d'une opération. En l'honneur du mathématicien arabe Al-Khwârizmî, elle appelle « algorithme » le processus logique permettant l'exécution d'un programme.

1.7 L'algèbre de Boole

Dans son ouvrage *Les lois de la pensée*, George Boole explique que l'on peut coder les démarches de la pensée à l'aide de systèmes n'ayant que deux états : ZERO-UN ; OUI-NON ; VRAI-FAUX, etc. L'algèbre de Boole, ou *calcul booléen*, voit alors le jour.

Il existe deux types d'opérations dans l'algèbre de Boole.

Propriété 1 : conjonction logique

Soit B un ensemble fini de deux éléments ($B = \{0\ ;\ 1\}$ ou $B = \{\text{faux}\ ;\ \text{vrai}\}$ par exemple). La fonction « ET » est définie sur B par le tableau suivant :

ET	0	1
0	0	0
1	0	1

équivalent à :

ET	faux	vrai
faux	faux	faux
vrai	faux	vrai

Propriété 2 : disjonction logique

Soit B un ensemble fini de deux éléments ($B = \{0\ ;\ 1\}$ ou $B = \{\text{faux}\ ;\ \text{vrai}\}$ par exemple). La fonction « OU » est définie sur B par le tableau suivant :

OU	0	1
0	0	1
1	1	1

équivalent à :

OU	faux	vrai
faux	faux	vrai
vrai	vrai	vrai

Il existe d'autres propriétés sur cette algèbre, mais ce n'est pas l'objet de ce cours.

Les booléens sont très utiles en programmation ; nous aurons l'occasion de nous en apercevoir ultérieurement dans ce livre.

COURS
INTERROS
CORRIGÉS

COURS

1.8 Le début du XXe siècle

C'est en 1936 qu'apparaît le concept de machine universelle, capable d'exécuter tous les algorithmes, et que les notions de machine, algorithme, langage et information sont pensées comme un tout cohérent. Les premiers ordinateurs ont été construits en 1948 et leur puissance a ensuite évolué exponentiellement, c'est-à-dire avec une croissance très rapide.

Un ordinateur ne peut stocker que des nombres. Pour écrire du texte, on associe à chaque caractère un entier naturel et la correspondance entre le caractère et son code est appelé un *charset*. La façon de transcrire un texte grâce aux codes des caractères qui le composent est appelé *Encoding* (encodage).

2 Les encodages

2.1 Le binaire

Définition 2 : bit

Le *bit* (contraction de *Binary Digit*) est l'unité la plus simple en informatique. Il ne peut prendre que deux valeurs (désignées le plus souvent par 0 et 1).

Un bit sert essentiellement à représenter un « état » : vrai ou faux, oui ou non, le courant électrique passe ou ne passe pas, etc.

Remarque : le mot « bit » est certes une contraction, mais il y a aussi un jeu de mot avec le mot anglais « bit » qui signifie « petit morceau ».
Le mot « bit » a été popularisé par Claude Shannon, ingénieur en génie électrique et mathématicien américain du XXe siècle, qui en attribue l'invention à John Tukey, l'un des plus importants mathématiciens américains du XXe siècle.

Définition 3 : système binaire

Le *système binaire* est le système de numération utilisant comme base le bit.

Le système binaire permet d'écrire ou d'encoder des nombres ou des caractères.

2.2 L'ASCII

Les premiers textes étaient encodés à l'aide d'une sorte d'alphabet, un tableau contenant 128 caractères codés en binaire, c'est-à-dire 2^7 caractères, afin que l'on puisse encoder les caractères avec seulement 7 bits. En effet, les ordinateurs utilisaient des cases mémoires d'un octet, mais ils réservaient toujours le 8e bit pour le contrôle de parité (c'est une sécurité pour éviter les erreurs, qui étaient très fréquentes dans les premières mémoires électroniques).

Ce tableau contenait les caractères les plus utilisés dans la langue anglaise : les lettres de l'alphabet en majuscule (de A à Z) et en minuscule (de a à z), les dix chiffres arabes (de 0 à 9), des signes de ponctuation (point, virgule, point-virgule, deux points, points d'exclamation et d'interrogation, apostrophe ou quote, guillemets ou double quote, parenthèses, crochets etc.), quelques symboles et certains caractères spéciaux invisibles (espace, retour-chariot, tabulation, retour-arrière, etc.).

ASCII est l'acronyme d'*American Standard Code for Information Interchange*.

Code décimal	Caractère
0	[NUL]
...	...
27	[ESC]
...	...
33	!
34	"
35	#
...	...
48	0
49	1
...	...
57	9
...	...

Code décimal	Caractère
65	A
66	B
...	...
90	Z
91	[
...	...
97	a
98	b
...	...
122	z
123	{
...	...
127	[DEL]

Extrait de la table ASCII

Exemple : le caractère « A » est codé en ASCII par le nombre 65 (dans le système décimal), qui correspond au nombre binaire 1000001, et au nombre hexadécimal 41.

Définition 4 : octet

Un *octet* est une suite de 8 bits (un *octuplet*).

Exemple : « 10000001 » est l'octet représentant le caractère « A ».

Ainsi, chaque caractère d'un texte est codé sur un octet.

Un texte de 10 000 caractères est donc codé sur 10 $\times$ 1 000 octets, soit approximativement 10 kilo-octets (1 ko = 10^8 octets = 1 024 octets).

L'ASCII a tout de même un inconvénient majeur : celui de ne pouvoir coder les caractères accentués, qui apparaissent dans plusieurs langues, dont le français.

Il a donc fallu trouver un moyen d'étendre la table ASCII...

COURS

2.3 La naissance de l'ISO

La norme ISO 8859-1, dont le nom complet est ISO/CEI 8859-1 et qui est souvent appelée Latin-1 ou Europe occidentale, fit son apparition dans le but de remédier à l'inconvénient de l'ASCII. Elle forme la première partie de la norme internationale ISO/CEI 8859, qui est une norme de l'*Organisation internationale de normalisation* pour le codage des caractères en informatique.

En France, depuis l'apparition de l'euro, c'est le codage ISO-8859-15 (souvent appelé Latin-9) qui permet d'écrire tout ce que l'on veut dans notre langue. C'est une sorte de mise à jour de la norme ISO 8859-1.

2.4 Au niveau mondial : l'Unicode

Le problème de la norme ISO 8859-15 se trouve principalement dans les échanges de textes à l'échelle mondiale.

Imaginez que vous codiez un programme en France et que vous souhaitiez collaborer avec un collègue américain. Ce dernier n'utilise pas nécessairement le même encodage que vous et va donc avoir quelques problèmes quand il lira vos propres codes.

Il est donc primordial, dans ce cas de figure, d'utiliser un encodage qui puisse être utilisé dans tous les pays. C'est le but d'Unicode, dont la première publication remonte à octobre 1991.

Définition 5 : point de code

Un *point de code* est la valeur numérique qui représente un caractère dans l'espace de codage des caractères.

Unicode accepte plusieurs formes de transformations universelles pour représenter un point de code valide. Citons par exemple :

- UTF-8 ;
- UTF-16 ;
- UTF-32.

Le nombre après UTF représente le nombre minimal de bits des codets (c'est-à-dire des groupes de signes représentant une information) avec lesquels un point de code valide est représenté.

L'UTF-8 est par exemple le plus commun pour les applications comme Unix et Internet. Son codage de taille variable lui permet d'être en moyenne moins coûteux en occupation mémoire

L'UTF-16 est un bon compromis lorsque la place mémoire n'est pas trop restreinte, car la grande majorité des caractères Unicode assignés pour les écritures des langues modernes (dont les caractères les plus fréquemment utilisés) le sont dans le plan multilingue de base et peuvent donc être représentés sur 16 bits.

L'UTF-32 est utilisé lorsque la place mémoire n'est pas un problème et que l'on a besoin d'avoir accès à des caractères de manière directe et sans changement de taille (par exemple les hiéroglyphes égyptiens). L'avantage de cette transformation normalisée est que tous les codets ont la même taille. Il n'est donc pas nécessaire de lire des codets supplémentaires pour déterminer le début de la représentation d'un point de code. Toutefois, ce format est particulièrement peu économique (y compris en mémoire) puisqu'il « gaspille » inutilement au moins un octet (toujours nul) par caractère.

2.5 Les logiciels

Selon le système d'exploitation que l'on utilise, il existe quelques éditeurs de textes qui permettent de choisir l'encodage des caractères.
Par exemple,

- sous Windows, *Notepad++* ;
- sous OSX, *TextEdit* ;
- sous Unix (Linux par exemple), *Kwrite* ou *Geany*.

3 Les bases

Définition 6 : intervalle d'entiers

Soient m et n deux nombres entiers.
On note $[\![m\ ;\, n]\!]$ l'ensemble de tous les entiers compris entre m (inclus) et n (inclus).

3.1 Principe général

Soit N un entier naturel. Si :

$$N = a_n \times b^n + a_{n-1} \times b^{n-1} + \cdots + a_1 \times b^1 + a_0 \times b^0 \quad , \quad a_k \in [\![0\ ;\ b-1]\!],$$

alors on dira que l'écriture de n en base b est :

$$N = \overline{a_n a_{n-1} a_{n-2} \cdots a_2 a_1 a_0}^{\,b}.$$

Remarque : afin de savoir en quelle base on écrit un nombre, on l'écrit en le surmontant d'une barre suivie de la base.

Exemple : soit N tel que :

$$N = 3 \times 5^4 + 4 \times 5^3 + 1 \times 5^1 + 2 \times 5^0.$$

Alors,

$$N = \overline{34012}^{\,5}.$$

3.2 En base 2

La base 2 est le système binaire. Elle est donc très importante en informatique. La base est constituée uniquement de deux chiffres : 0 et 1. Ainsi, tout nombre écrit en base 2 ne comportera que des 0 et des 1.

Pour convertir un nombre décimal en base 2, il faut effectuer des divisions euclidiennes successives par 2 et écrire les restes obtenus en « remontant ».

Exemple : on souhaite convertir $\overline{25}^{10}$ en base 2.

$$\begin{array}{r|l} 25 & 2 \\ \cline{2-2} 05 & 12 \\ 1 & \end{array} \rightarrow \begin{array}{r|l} 12 & 2 \\ \cline{2-2} 0 & 6 \end{array} \rightarrow \begin{array}{r|l} 6 & 2 \\ \cline{2-2} 0 & 3 \end{array} \rightarrow \begin{array}{r|l} 3 & 2 \\ \cline{2-2} 1 & 1 \end{array} \rightarrow \begin{array}{r|l} 1 & 2 \\ \cline{2-2} 1 & 0 \end{array}$$

On en déduit que :

$$25 = 1 \times 2^0 + 0 \times 2^1 + 0 \times 2^2 + 1 \times 2^3 + 1 \times 2^4$$

soit :

$$25 = \boxed{1} \times 2^4 + \boxed{1} \times 2^3 + \boxed{0} \times 2^2 + \boxed{0} \times 2^1 + \boxed{1} \times 2^0.$$

Ainsi,

$$\overline{25}^{10} = \overline{11001}^{2}.$$

MÉTHODE

Après avoir posé toutes les divisions, on écrit tous les restes en partant du dernier obtenu jusqu'au premier.

3.3 La base hexadécimale

Elle est constituée de 16 caractères :

0; 1; 2; 3; 4; 5; 6; 7; 8; 9; A; B; C; D; E; F.

3.3.1 - Du décimal à l'hexadécimal

Pour convertir un nombre décimal en hexadécimal, on effectue des divisions euclidiennes par 16, sur le même principe que l'écriture en base 2.

Exemple : on souhaite convertir le nombre décimal 1 234 en hexadécimal.

$$\begin{array}{r|l} 1234 & 16 \\ \cline{2-2} 114 & 77 \\ 2 & \\ \downarrow & \\ 2 & \end{array} \rightarrow \begin{array}{r|l} 77 & 16 \\ \cline{2-2} 13 & 4 \\ \downarrow & \\ 13 \rightarrow D & \end{array} \rightarrow \begin{array}{r|l} 4 & 16 \\ \cline{2-2} 4 & 0 \\ \downarrow & \\ 4 & \end{array}$$

Ainsi,

$$\overline{1234}^{10} = \overline{4D2}^{16}.$$

3.3.2 - Du binaire à l'hexadécimal

Pour convertir du binaire à l'hexadécimal, il suffit de regrouper les bits par groupe de 4 en allant de la droite à la gauche.

Exemple : on souhaite convertir $\overline{01001101}^{2}$ en hexadécimal.

Binaire	0100	1101
Pseudo-décimal	4	13
Hexadécimal	4	D

Ainsi,

$$\overline{01001101}^{2} = \overline{4D}^{16}.$$

3.3.3 - De l'hexadécimal au décimal

Le principe est le même que pour passer du binaire au décimal, si ce n'est que l'on va remplacer les lettres par les nombres correspondants (on remplace A par 10, B par 11...).

Exemple : on souhaite convertir $\overline{7EF}^{16}$ en décimal.

- F, correspondant au nombre 15, est au rang 0 ;
- E, correspondant au nombre 14, est au rang 1 ;
- 7 est au rang 2.

On doit donc calculer :

$$7 \times 16^2 + 14 \times 16^1 + 15 \times 16^0 = 2\,031.$$

Ainsi, $\overline{7EF}^{16} = \overline{2\,031}^{10}$.

Solution de l'exercice type — Lycée Raoul Follereau, Nevers

1 $\overline{1000001}^{2}$ correspond à la lettre « A ». Pour obtenir le binaire correspondant à la lettre « B », il faut lui ajouter 1.

Il y a deux façons de faire : soit on convertit 66 en binaire (car 65 est le code ASCII de « A », donc 66 est celui de « B »), soit on prend le suivant de $\overline{1000001}^{2}$. Ce binaire se terminant par 1, son suivant se termine par 0, et il faut ajouter 1 au bit précédent.

On trouve alors que $\overline{1000010}^{2}$ est le binaire correspondant à « B ».

COURS

Solution de l'exercice type

Lycée Raoul Follereau, Nevers

2 Pour convertir $\overline{75}^{\,10}$ en binaire, il faut regarder les divisions euclidiennes par 2 :

$$\begin{array}{r|l} 75 & 2 \\ \cline{2-2} 15 & 37 \\ 1 & \end{array} \rightarrow \begin{array}{r|l} 37 & 2 \\ \cline{2-2} 17 & 18 \\ 1 & \end{array} \rightarrow \begin{array}{r|l} 18 & 2 \\ \cline{2-2} 0 & 9 \end{array} \rightarrow$$

$$\begin{array}{r|l} 9 & 2 \\ \cline{2-2} 1 & 4 \end{array} \rightarrow \begin{array}{r|l} 4 & 2 \\ \cline{2-2} 0 & 2 \end{array} \rightarrow \begin{array}{r|l} 2 & 2 \\ \cline{2-2} 0 & 1 \end{array} \rightarrow \begin{array}{r|l} 1 & 2 \\ \cline{2-2} 1 & 0 \end{array}$$

On prend ensuite les restes dans l'ordre inverse dans lequel nous les avons trouvés.
On obtient alors que $\overline{75}^{\,10} = \overline{1001011}^{\,2}$.

3 Pour convertir $\overline{2001}^{\,10}$ en hexadécimal, il faut diviser par 16.

$$\begin{array}{r|l} 2001 & 16 \\ \cline{2-2} 40 & 125 \\ 81 & \\ 1 & \end{array} \rightarrow \begin{array}{r|l} 125 & 16 \\ \cline{2-2} 13 & 7 \end{array} \rightarrow \begin{array}{r|l} 7 & 16 \\ \cline{2-2} 7 & 0 \end{array}$$

On prend alors les restes dans l'ordre inverse dans lequel nous les avons trouvés : il y en a un qui dépasse 9. Comme 10 correspond à A, 11 correspond à B, 12 correspond à C et donc 13 correspond à D.
On obtient alors $\overline{2001}^{\,10} = \overline{7D1}^{\,16}$.

Remarque : le film de Stanley Kubrick *2001, l'Odyssée de l'espace* se serait ainsi appelé en hexadécimal *7D1, l'Odyssée de l'espace*.

4 Pour convertir $\overline{ABC}^{\,16}$ en binaire, on peut commencer par le convertir en décimal.

- C, correspondant au nombre 12, occupe le rang 0 ;
- B, correspondant au nombre 11, occupe le rang 1 ;
- A, correspondant au nombre 10, occupe le rang 2.

On doit donc calculer :

$$10 \times 16^2 + 11 \times 16^1 + 12 \times 16^0 = 2\,748.$$

Ainsi, $\overline{ABC}^{\,16} = \overline{2\,748}^{\,10}$.

Solution de l'exercice type (suite)

Lycée Raoul Follereau, Nevers

Pour convertir $\overline{2\,748}^{10}$ en binaire, afin d'éviter les divisions euclidiennes successives (ce qui peut paraître long), on peut se baser sur les puissances successives de 2 :

À RETENIR

n	2^n
0	1
1	2
2	4
3	8
4	16
5	32

n	2^n
6	64
7	128
8	256
9	512
10	1 024
11	2 048

Les coefficients des puissances de 2 sont donc répartis ainsi :

$$\begin{aligned} 2\,748 &= 2\,048 + 512 + 128 + 32 + 16 + 8 + 4 \\ &= 2^{11} + 2^9 + 2^7 + 2^5 + 2^4 + 2^3 + 2^2. \end{aligned}$$

On a donc :

2^{11}	2^{10}	2^9	2^8	2^7	2^6	2^5	2^4	2^3	2^2	2^1	2^0
1	0	1	0	1	0	1	1	1	1	0	0

Ainsi, $\overline{\mathrm{ABC}}^{16} = \overline{2\,748}^{10} = \overline{101010111100}^{2}$.

À RETENIR

À l'avenir, pour aller plus vite dans les conversions, vous pourrez utiliser le convertisseur accessible à la page :

`http://sebastienguillon.com/test/javascript/convertisseur.html`

C'est un convertisseur binaire ↔ hexadécimal ↔ décimal.

Voir énoncé page 1

INTERROS

1 QCM Généralités

5 min

Corrigé p. 18

1 Le concept de machines universelles est apparu au :

a Moyen-Âge

b XVII^{e} siècle

c XIX^{e} siècle

d XX^{e} siècle

2 Le système binaire repose sur les chiffres :

a 0 et 1

b 1 et 2

3 Le système hexadécimal comporte :

a 14 caractères

b 15 caractères

c 16 caractères

d 17 caractères

Encodages

2 Taille d'un ficher

★ 5 min

Corrigé p. 18

1 Quel est la taille (en ko) d'un fichier texte contenant 75 000 caractères ?

2 **(a)** On tape la phrase « Bonjour à tous. » dans un fichier créé avec le logiciel Notepad++ sous Windows. Quelle est la taille (en octets) de ce fichier quand on le sauvegarde sur le disque dur ?

(b) On tape la même phrase dans un fichier créé avec le logiciel Open Writer de la suite Open Office. La taille du fichier sauvegardé est 8 683 octets.

Comment expliquer cette taille ?

3 Coder en binaire

★ 10 min

Corrigé p. 18

À l'aide d'une table ASCII (voir page 7), répondre aux questions suivantes.

1 Convertir en binaire la phrase suivante :

```
Bonjour cher ami.
```

2 Que se cache-t-il derrière le code binaire suivant ?

```
01101001 01101110 01100110 01101111 00100000 01100110
01101111 01110010 00100000 01100101 01110110 01100101
01110010 00100000 00100001
```

4 Dans les années 1990

★★ 5 min Corrigé p. 18

Avant l'apparition de l'Unicode, il n'était pas rare de recevoir un courriel comme celui-ci :

« Bonjour mon ami BarnabÃ©. Jâ€™espÃ¨ que vous allez bien. Moi, pas trop. Jâ€™ai besoin de 100 000 francs pour rentrer chez moi. Si vous pouviez mâ€™envoyer cette somme par virement bancaire, je vous en serez Ã©ternelement reconnaissant. Votre dévouÃ©e JosÃ©phine. »

1. Expliquer la raison pour laquelle ce message comporte des caractères non désirés.
2. A priori, quel est l'encodage du message original ?

5 Précisions sur l'UTF-8

★★★ 30 min Corrigé p. 19

L'encodage UTF-8 utilise 1, 2, 3 ou 4 octets en respectant certaines règles :

- Un texte en ASCII de base (appelé aussi US-ASCII) est codé de manière identique en UTF-8. On utilise un octet commençant par un bit 0 à gauche (bit de poids fort).
  Par exemple, le caractère « A » est codé par le nombre décimal 65, ce qui correspond au nombre binaire 1000001 (7 bits). En utilisant le codage UTF-8, le binaire devient 01000001 (8 bits).
- Les octets ne sont pas remplis entièrement. Les bits de poids fort du premier octet forment une suite de 1 indiquant le nombre d'octets utilisés pour coder le caractère. Les octets suivants commencent tous par le bloc binaire 10.

Représentation binaire UTF-8	Signification
0xxxxxxx	1 octet codant 1 à 7 bits
110xxxxx 10xxxxxx	2 octets codant 8 à 11 bits
1110xxxx 10xxxxxx 10xxxxxx	3 octets codant 12 à 16 bits
11110xxx 10xxxxxx 10xxxxxx 10xxxxxx	4 octets codant 17 à 21 bits

Le symbole € correspond à la valeur décimale 8364.

1. Convertir 8364 en binaire.
2. Donner le codage UTF-8 de ce caractère.
3. Étant donné un caractère codé en binaire sur n bits, combien d'octets sont nécessaires pour le coder en UTF-8 ?
4. Écrire un algorithme qui permet, à partir de la valeur décimale d'un caractère, de trouver le nombre d'octets nécessaires pour coder ce caractère en UTF-8.

COURS INTERROS CORRIGÉS

Conversions

6 Du décimal au binaire

★ 10 min Corrigé p. 20

Convertir en binaire les nombres suivants, donnés en base 10.

1. $\overline{458}^{10}$
2. $\overline{133}^{10}$
3. $\overline{47}^{10}$
4. $\overline{1\,024}^{10}$
5. $\overline{65}^{10}$

7 Du binaire au décimal

★ 10 min Corrigé p. 21

Convertir les nombres suivants (écrits en binaire) en décimal.

1. $\overline{101010101}^{2}$
2. $\overline{111000}^{2}$
3. $\overline{00110011}^{2}$
4. $\overline{1010010001}^{2}$

8 Du binaire à l'hexadécimal

★ 10 min Corrigé p. 21

Convertir les nombres binaires suivants en hexadécimal.

1. $\overline{10000000011001}^{2}$
2. $\overline{1000100010001}^{2}$
3. $\overline{100110000111}^{2}$
4. $\overline{101110101100}^{2}$
5. $\overline{110010101111111100010}^{2}$

9 Du décimal à l'hexadécimal

★ 10 min Corrigé p. 22

Convertir en hexadécimal les nombres suivants, exprimés en base 10.

1. $\overline{2020}^{10}$
2. $\overline{1234}^{10}$
3. $\overline{56026}^{10}$
4. $\overline{64218}^{10}$

10 De l'hexadécimal au binaire

★ 10 min Corrigé p. 23

Convertir en binaire les nombres suivants, écrits en hexadécimal.

1. $\overline{\text{A320}}^{16}$
2. $\overline{\text{FABE51}}^{16}$
3. $\overline{\text{101010}}^{16}$
4. $\overline{\text{59A75}}^{16}$

11 Codes transpondeur et base octale

★★ 10 min Corrigé p. 24

Lycée Augustin Fresnel, Caen

En aéronautique, un transpondeur est un dispositif électronique embarqué dans un aéronef qui émet une message radio (codé) quand il reçoit une interrogation par radio émise par un faisceau radar. Les avions possèdent des transpondeurs pour permettre aux radars de les identifier et de faciliter leur suivi. Afin de bien identifier les avions dont ils ont la charge sur leurs écrans, les contrôleurs aériens demandent aux pilotes, par radio, de régler un code de 4 chiffres sur leurs transpondeurs. Les codes transpondeur sont toujours constitués de *4 chiffres en base 8*, par exemple « 7040 ».

1. Donner la séquence de chiffres utilisée pour exprimer un nombre en base 8.
2. Quel est le nombre de bits nécessaires à l'écriture en base 2 du plus grand chiffre cité à la question précédente ?
3. En déduire le nombre de bits total nécessaires pour représenter en base 2 un code transpondeur complet.
4. Quel est le nombre total de codes transpondeur disponibles avec ce système ?

12 Algorithme de conversion

★★ 10 min Corrigé p. 24

Retrouvez le corrigé de cet exercice en vidéo.

Écrire un algorithme permettant de saisir un nombre décimal et de le convertir en binaire.

COURS
INTERROS
CORRIGÉS

1 QCM Généralités

Énoncé p. 14

1 Réponse $\boxed{\text{d}}$. En 1936 est apparu le concept de machine universelle.

2 Réponse $\boxed{\text{b}}$. En binaire, il n'y a que deux caractères : le 0 et le 1.

3 Réponse $\boxed{\text{c}}$. En hexadécimal, il y a 16 caractères :

0, 1, 2, 3, 4, 5, 6, 7, 8, 9, A, B, C, D, E et F.

2 Taille d'un ficher

Énoncé p. 14

1 D'après la correspondance « 1 caractère = 1 octet », la taille d'un fichier texte contenant 75 000 caractères est 75 000 octets, soit 75 ko.

2 **(a)** Il y a 15 caractères dans la phrase (ponctuation et espaces compris), donc la taille du fichier texte est 15 octets.

(b) Contrairement à Notepad++, qui sauvegarde le texte en *brut* (c'est-à-dire sans rien de plus que le texte lui-même), les éditeurs de textes plus sophistiqués (comme Open Writer) enregistrent la configuration du fichier (sa mise en forme, la position du curseur...). Ainsi, la taille du fichier sauvegardé est toujours strictement plus grande que celle du texte brut.

3 Coder en binaire

Énoncé p. 14

1 Le caractère « B » a pour code binaire 01000010. On fait de même pour les autres caractères, y compris les espaces entre les mots et le point final.

On obtient alors le code suivant :

```
01000010 01101111 01101110 01101010 01101111 01110101
01110010 00100000 01100011 01101000 01100101 01110010
00100000 01100001 01101101 01101001 00101110
```

2 Le code binaire donné correspond à la phrase : « info for ever ! ».

MÉTHODE

Pour aller plus vite, on peut utiliser un convertisseur en ligne, par exemple celui de la page :
`http://usefulwebtool.com/fr/convertir-texte-en-binaire.php.`

4 Dans les années 1990

Énoncé p. 15

1 La raison pour laquelle des caractères bizarres apparaissent est que le texte original a été encodé avec une norme différente de celle utilisée pour le lire.

2 Nous pouvons constater que les caractères accentués ont été codés sur deux caractères ; par exemple, le « é » apparaît sous la forme « Ã© ». Ces derniers caractères apparaissant dans la norme UTF-8, on peut supposer que le message original a été tapé sous cette norme, et que le lecteur de courriel l'interprète comme étant sous la norme ISO 8859-1.

5 Précisions sur l'UTF-8

Énoncé p. 15

1 $\overline{8364}^{10} = \overline{10000010101100}^{2}$. On pourra regarder l'exercice 6 pour voir la méthode à utiliser.

2 Pour obtenir le codage UTF-8 du caractère « € », on part de la droite de sa représentation binaire et on compte 6 bits ; on a alors :

$$101100.$$

Ensuite, on met « 10 » devant ces 6 bits :

$$10101100.$$

On prend ensuite les 6 bits précédents, à savoir « 000010 », devant lesquels on met « 10 ».

Il ne reste plus que « 10 », que nous mettons à la fin du premier octet, celui qui commence nécessairement par « 1110 » car nous venons d'obtenir 3 octets.

Au final, on obtient que le codage UTF-8 du caractère « € » est :

$$11100010\ 10000010\ 10101100.$$

3 À travers l'exemple de la question précédente, on peut voir que l'on regroupe les bits par 6 pour former les octets, et que l'on ajoute un octet pour indiquer le nombre total d'octets.

Ainsi, dans l'exemple précédent, $n = 14$ et le nombre d'octets nécessaires en UTF-8 est :

$$E\left(\frac{14}{6}\right) + 1,$$

où $E(x)$ désigne la partie entière de x (par exemple, $E(\pi) = 3$).

On peut donc dire que le nombre d'octets nécessaires, dans un cas général, est $E\left(\frac{n}{6}\right) + 1$.

4 Pour avoir le nombre de bits du nombre binaire correspondant à la valeur décimale D du caractère, il faut connaître le nombre n tel que :

$$2^{n-1} < D \leqslant 2^n.$$

Par exemple, $2^{13} < 8\,364 \leqslant 2^{14}$ donc le nombre de bits nécessaires en binaire est 14.

Une fois que l'on a ce nombre, on utilise la formule de la question précédente. Cela donne :

Algorithme

```
D est un entier naturel
n ← 0
Tant que 2^n ⩽ D+1 faire
   n ← n+1
Fin du Tant que
Afficher E(n/6)+1
```

En Python

```python
D = int(input("Nombre décimal : "))
n = 0

while (2**n <= D+1):
    n += 1

print(int(n/6)+1)
```

6 Du décimal au binaire

Énoncé p. 16

1 Convertissons $\overline{458}^{10}$.

$$\begin{array}{r|l} 458 & 2 \\ \cline{2-2} 05 & 229 \\ 18 & \\ 0 & \end{array}$$

$$\begin{array}{r|l} 229 & 2 \\ \cline{2-2} 02 & 114 \\ 09 & \\ 1 & \end{array}$$

$$\begin{array}{r|l} 114 & 2 \\ \cline{2-2} 14 & 57 \\ 0 & \end{array}$$

$$\begin{array}{r|l} 57 & 2 \\ \cline{2-2} 17 & 28 \\ 1 & \end{array}$$

$$\begin{array}{r|l} 28 & 2 \\ \cline{2-2} 08 & 14 \\ 0 & \end{array}$$

$$\begin{array}{r|l} 14 & 2 \\ \cline{2-2} 0 & 7 \end{array}$$

$$\begin{array}{r|l} 7 & 2 \\ \cline{2-2} 1 & 3 \end{array}$$

$$\begin{array}{r|l} 3 & 2 \\ \cline{2-2} 1 & 1 \end{array}$$

$$\begin{array}{r|l} 1 & 2 \\ \cline{2-2} 1 & 0 \end{array}$$

Donc $\overline{458}^{10} = \overline{111001010}^{2}$.

2 Convertissons $\overline{133}^{10}$.

$$\begin{array}{r|l} 133 & 2 \\ \cline{2-2} 13 & 66 \\ 1 & \end{array}$$

$$\begin{array}{r|l} 66 & 2 \\ \cline{2-2} 06 & 33 \\ 0 & \end{array}$$

$$\begin{array}{r|l} 33 & 2 \\ \cline{2-2} 13 & 16 \\ 1 & \end{array}$$

$$\begin{array}{r|l} 16 & 2 \\ \cline{2-2} 0 & 8 \end{array}$$

$$\begin{array}{r|l} 8 & 2 \\ \cline{2-2} 0 & 4 \end{array}$$

$$\begin{array}{r|l} 4 & 2 \\ \cline{2-2} 0 & 2 \end{array}$$

$$\begin{array}{r|l} 2 & 2 \\ \cline{2-2} 0 & 1 \end{array}$$

$$\begin{array}{r|l} 1 & 2 \\ \cline{2-2} 1 & 0 \end{array}$$

Donc $\overline{133}^{10} = \overline{10000101}^{2}$.

3 Convertissons $\overline{47}^{10}$.

$$\begin{array}{r|l} 47 & 2 \\ \cline{2-2} 07 & 23 \\ 1 & \end{array} \rightarrow \begin{array}{r|l} 23 & 2 \\ \cline{2-2} 03 & 11 \\ 1 & \end{array} \rightarrow \begin{array}{r|l} 11 & 2 \\ \cline{2-2} 1 & 5 \end{array} \rightarrow \begin{array}{r|l} 5 & 2 \\ \cline{2-2} 1 & 2 \end{array} \rightarrow \begin{array}{r|l} 2 & 2 \\ \cline{2-2} 0 & 1 \end{array} \rightarrow \begin{array}{r|l} 1 & 2 \\ \cline{2-2} 1 & 0 \end{array}$$

Donc $\overline{47}^{10} = \overline{101111}^{2}$.

4 On sait que $1\,024 = 2^{10}$ donc $\overline{1\,024}^{10} = \overline{10000000000}^{2}$.

5 $65 = 64 + 1 = 2^6 + 1$ donc $\overline{65}^{10} = \overline{1000001}^{2}$.

7 Du binaire au décimal

Énoncé p. 16

1 $\overline{101010101}^{2} = 1 \times 2^8 + 1 \times 2^6 + 1 \times 2^4 + 1 \times 2^2 + 1 \times 2^0 = 341.$

2 $\overline{111000}^{2} = 1 \times 2^5 + 1 \times 2^4 + 1 \times 2^3 = 56.$

3 $\overline{00110011}^{2} = 2^5 + 2^4 + 2^1 + 2^0 = 51.$

4 $\overline{1010010001}^{2} = 2^9 + 2^7 + 2^4 + 2^0 = 657.$

8 Du binaire à l'hexadécimal

Énoncé p. 16

1 On regroupe par groupes de 4 bits :

Binaire	10	0000	0001	1001
Pseudo-décimal	2	0	1	9
Hexadécimal	2	0	1	9

Ainsi, $\overline{10000000011001}^{2} = \overline{2019}^{16}$.

CORRIGÉS

2 On regroupe par groupes de 4 bits :

Binaire	1	0001	0001	0001
Pseudo-décimal	1	1	1	1
Hexadécimal	1	1	1	1

Ainsi, $\overline{1000100010001}^{2} = \overline{1111}^{16}$.

3 On regroupe par groupes de 4 bits :

Binaire	1001	1000	0111
Pseudo-décimal	9	8	7
Hexadécimal	9	8	7

Ainsi, $\overline{100110000111}^{2} = \overline{987}^{16}$.

4 On regroupe par groupes de 4 bits :

Binaire	1011	1010	1100
Pseudo-décimal	11	10	13
Hexadécimal	*B*	*A*	*C*

Ainsi, $\overline{101110101100}^{2} = \overline{\text{BAC}}^{16}$.

5 On regroupe par groupes de 4 bits :

Binaire	1100	1010	1111	1110	0010
Pseudo-décimal	13	10	16	15	2
Hexadécimal	*C*	*A*	*F*	*E*	2

Ainsi, $\overline{11001010111111100010}^{2} = \overline{\text{CAFE2}}^{16}$.

9 Du décimal à l'hexadécimal

Énoncé p. 16

1 Convertissons $\overline{2020}^{10}$.

$$\begin{array}{r|l} 2020 & 16 \\ \cline{2-2} 42 & 126 \\ 100 & \\ 4 & \end{array}$$

$$\begin{array}{r|l} 126 & 16 \\ \cline{2-2} 14 & 7 \end{array}$$

$$\begin{array}{r|l} 7 & 16 \\ \cline{2-2} 7 & 0 \end{array}$$

Donc $\overline{2020}^{10} = \overline{7\text{E}4}^{16}$.

2 Convertissons $\overline{1234}^{10}$.

$$\begin{array}{r|l} 1234 & 16 \\ \cline{2-2} 114 & 77 \\ 2 & \end{array}$$

$$\begin{array}{r|l} 77 & 16 \\ \cline{2-2} 13 & 4 \end{array}$$

$$\begin{array}{r|l} 4 & 16 \\ \cline{2-2} 4 & 0 \end{array}$$

Ainsi, $\overline{1234}^{10} = \overline{4\text{D}2}^{16}$.

3 Convertissons $\overline{56026}^{10}$.

$$\begin{array}{r|l} 56026 & 16 \\ \cline{2-2} 80 & 3501 \\ 026 & \\ 10 & \end{array}$$

$$\begin{array}{r|l} 3501 & 16 \\ \cline{2-2} 30 & 218 \\ 141 & \\ 13 & \end{array}$$

$$\begin{array}{r|l} 218 & 16 \\ \cline{2-2} 58 & 13 \\ 10 & \end{array}$$

$$\begin{array}{r|l} 13 & 16 \\ \cline{2-2} 13 & 0 \end{array}$$

Donc $\overline{56026}^{10} = \overline{\text{DADA}}^{16}$.

4 Convertissons $\overline{64218}^{10}$.

$$\begin{array}{r|l} 64218 & 16 \\ \cline{2-2} 021 & 4013 \\ 058 & \\ 10 & \end{array}$$

$$\begin{array}{r|l} 4013 & 16 \\ \cline{2-2} 81 & 250 \\ 13 & \end{array}$$

$$\begin{array}{r|l} 250 & 16 \\ \cline{2-2} 90 & 15 \\ 10 & \end{array}$$

$$\begin{array}{r|l} 15 & 16 \\ \cline{2-2} 15 & 0 \end{array}$$

Donc $\overline{64218}^{10} = \overline{\text{FADA}}^{16}$.

10 De l'hexadécimal au binaire

Énoncé p. 17

1

Hexadécimal	*A*	3	2	0
Pseudo-décimal	10	3	2	0
Binaire	1010	0011	0010	0000

Donc $\overline{\text{A320}}^{16} = \overline{1010001100100000}^{2}$.

2

Hexadécimal	*F*	*A*	*B*	*E*	5	1
Pseudo-décimal	15	10	11	14	5	1
Binaire	1111	1010	1011	1110	0101	0001

Donc $\overline{\text{FABE}}^{16} = \overline{111110101011111001010001}^{2}$.

3

Hexadécimal	1	0	1	0	1	0
Pseudo-décimal	1	0	1	0	1	0
Binaire	0001	0000	0001	0000	0001	0000

Donc $\overline{101010}^{16} = \overline{100000001000000010000}^{2}$.

4

Hexadécimal	5	9	*A*	7	5
Pseudo-décimal	5	9	10	7	5
Binaire	0101	1001	1010	0111	0101

Donc $\overline{101010}^{16} = \overline{1011001101001110101}^{2}$.

COURS
INTERROS
CORRIGÉS

11 Codes transpondeur et base octale

Énoncé p. 17

Lycée Augustin Fresnel, Caen

1. En base 8, on utilise les chiffres 0, 1, 2, 4, 5, 6 et 7.
2. Il faut 3 bits pour coder 7 en binaire : $\overline{7}^{8} = \overline{111}^{2}$.
3. Il faut $4 \times 3 = 12$ bits pour encoder un code transpondeur en binaire.
4. On dispose au total de $8^4 = 2^{1}2 = 4096$ codes différents, ce qui est largement suffisant pour attribuer un code unique à chaque avion d'un même secteur.

12 Algorithme de conversion

Énoncé p. 17

Algorithme

```
n prend une valeur entière entrée par l'utilisateur
b est une chaîne de caractères vide
Tant que n ≠ 0:
   q ← partie entière de n/2
   reste ← n-2*q
   On convertit reste en chaîne de caractères
   b ← reste + b
   n ← q
Fin du Tant que
Afficher b
```

Notre objectif est de construire une chaîne de caractères, composée de 0 et de 1, en partant de la fin car pour convertir un nombre décimal en binaire, on prend les restes des divisions euclidiennes en partant de la fin. On initialise donc une chaîne de caractères vide (ici, `b`). Nous voyons sur les conversions faites dans l'exercice 6 que l'on s'arrête quand le quotient est nul. Notre objectif sera de créer une boucle qui transformera `n` en le quotient de la division euclidienne de n par 2. La condition pour exécuter les instructions de la boucle est donc « $n \neq 0$ ».

En Python

```python
n = int(input("Entrez un nombre décimal : "))
b = ''

while n != 0:
    q = int(n/2)
    b = str(n-q*2)+b
    n = q

print(b)
```

Chapitre 2

Types et valeurs de base

Plan du chapitre

1. Représentation binaire d'un entier
2. Représentation approximative des nombres réels
3. Les booléens
4. Les variables sous Python

En programmation informatique, il existe plusieurs types de données.
Nous allons voir dans ce chapitre les types élémentaires.

1 Représentation binaire d'un entier

Exercice type 1 — Lycée Sainte-Anne, Brest

On dispose d'une mémoire de 8 bits.

1. Donner le complément à 2 du nombre 78.
2. Donner le complément à 2 du nombre -111.
3. Donner alors le complément à 2 de la somme des nombres 78 et -111.
4. Vérifier que le nombre binaire trouvé est bien la somme de ces deux nombres.

Voir corrigé page 30

Définition 1

Soit $\overline{x_n x_{n-1} x_{n-2} \cdots x_1 x_0}^{\,2}$ la représentation binaire d'un entier naturel.

- Le bit de rang 0, c'est-à-dire x_0, est appelé le *bit de poids faible*.
- Le bit de rang n, c'est-à-dire x_n, est appelé le *bit de poids fort*.

1.1 Représentation binaire d'un entier naturel

Nous avons vu dans le chapitre précédent qu'un entier naturel pouvait être codé en binaire.

Définition 2

On nomme *binaire pur* le codage dans lequel sont écrits les entiers naturels.

COURS

Exemple : le nombre décimal 123 peut être codé en binaire par 1111011.

Ainsi, avec une mémoire de n bits, tous les entiers de 0 à $2^n - 1$ seront représentés. Par exemple,

- avec une mémoire de 8 bits, on peut coder tous les entiers naturels de 0 à 255 ;
- avec une mémoire de 16 bits, on peut coder tous les entiers naturels de 0 à 65 535 ;
- avec une mémoire de 32 bits, on peut coder tous les entiers naturels de 0 à 4 294 967 295 ;
- avec une mémoire de 64 bits, on peut coder tous les entiers naturels de 0 à 18 446 744 073 709 551 615.

1.2 Représentation binaire d'un entier relatif

1.2.1 - Le binaire signé

Définition 3

On nomme *binaire signé* le codage dans lequel sont écrits les entiers relatifs.

Dans la représentation en binaire signé, le bit de poids fort sert à représenter le signe (« 0 » pour un entier positif et « 1 » pour un entier négatif), les autres bits représentent le nombre (sans le signe) en binaire pur.

Exemples

- $\overline{01011001}^{2}$ est la représentation en binaire signé du nombre $\overline{89}^{10}$ avec une mémoire de 8 bits.
- $\overline{11011001}^{2}$ est la représentation en binaire signé du nombre $\overline{-89}^{10}$ avec une mémoire de 8 bits.

Ainsi, avec une mémoire de $n + 1$ bits, tous les entiers relatifs de $-(2^n - 1)$ à $2^n - 1$ seront représentés. Par exemple,

- avec une mémoire de 8 bits, on peut coder tous les entiers naturels de -127 à 127 ;
- avec une mémoire de 16 bits, on peut coder tous les entiers naturels de $-32\,768$ à 32 768 ;
- avec une mémoire de 32 bits, on peut coder tous les entiers naturels de $-2\,147\,483\,647$ à 2 147 483 647 ;
- avec une mémoire de 64 bits, on peut coder tous les entiers naturels de $-9\,223\,372\,036\,854\,775\,807$ à 9 223 372 036 854 775 807.

Remarque : le nombre zéro est représenté dans cette convention (dites du *zéro positif*) par : 0000...00000.
Remarquez alors que l'état binaire $\underbrace{1000\ldots000}_{n \text{ bits}}$ ne représente rien *a priori*.

Cependant, les informaticiens ont tout de même décidé qu'il représentait le nombre -2^n.

Fort de cette dernière remarque, l'intervalle de représentation se trouve alors augmenté d'un nombre ; il devient : $[\![-2^n\ ;\ 2^n - 1]\!]$.

Dans la pratique, nous n'utilisons pas ce dernier codage. En effet, le traitement spécifique du signe coûte cher en circuits électroniques et en temps de calcul.

On utilise plutôt une version améliorée : le complément à 2.

1.2.2 - Le complément à 2

Ce codage est légèrement plus complexe que le binaire signé.

MÉTHODE

Soit n un nombre exprimé en base 10 ; on cherche son complément à 2.

1. si $n > 0$, on utilise le binaire signé ;
2. si $n < 0$,
   - on code $|n|$ en binaire signé (c'est-à-dire n sans son signe),
   - on prend le complément à 1 du binaire trouvé (c'est-à-dire que l'on remplace tous les 0 par des 1, et tous les 1 par des 0),
   - on additionne 1 au nombre binaire ainsi obtenu.

Exemple : on souhaite trouver le complément à 2 de $\overline{-57}^{10}$ avec une mémoire de 8 bits. Il est négatif donc :

- on commence par coder le nombre sans son signe en binaire signé, et on obtient : $\overline{00111001}^{2}$;
- on prend le complément à 1, et on obtient : $\overline{11000110}^{2}$;
- on ajoute 1 au binaire obtenu, et on obtient : $\overline{11000111}^{2}$.

Remarques

- L'opération consistant à prendre le complément de chaque bit est appelée le *non logique* :

$$\text{non } 0 = 1 \qquad \text{et} \qquad \text{non } 1 = 0.$$

- C'est de la dernière opération que le codage tire son nom : le fait d'ajouter 1 au binaire s'appelle « prendre le complément à 2 ».

COURS

INTERROS

CORRIGÉS

COURS

ATTENTION

En binaire, si ajouter 1 à 0 donne 1, ajouter 1 à 1 donne 10.

L'avantage du complément à 2 se trouve dans les opérations sur les entiers (voir paragraphe 1.3).

Remarque : avec une mémoire de 8 bits, le complément à 2 du nombre -128 est $\overline{10000000}^{2}$. Le complément à 2 de $+128$ n'existe pas (avec une mémoire de 8 bits.

1.2.3 - Entiers de taille arbitraire

Faisons une petite expérience : ouvrons la console Python, et affectons à la variable x le plus grand entier possible, puis affichons la valeur de x :

```
x = 0x7fffffffffffffff
print(x)
```

La console affiche alors :

`9223372036854775807.`

Maintenant, demandons d'ajouter 1 à x :

```
x + 1
```

La console affiche :

`9223372036854775808.`

Que s'est-il passé ? `x` est sensé être le plus grand entier... et pourtant, `x+1` existe !

En fait, Python a détecté le dépassement de capacité et il décide de ne plus représenter le résultat comme un entier mais comme un *entier long*. Et contrairement au complément à 2, cette représentation n'est pas limitée en taille. Cela peut paraître intéressant comme représentation, mais elle a des inconvénients. Par exemple, les opérations sur les entiers longs prennent plus de temps que celles sur les entiers relatifs représentés en complément à 2.

La représentation des entiers longs étant bien plus complexe que le complément à 2, nous ne la traiterons pas ici.

1.3 Addition et soustraction de deux entiers

Que ce soit pour additionner ou pour soustraire, on utilise le complément à 2.

1.3.1 - Addition

La règle d'or pour additionner en binaire est de retenir, comme nous l'avons vu précédemment, que $1 + 1 = 10$.

Exemple : on souhaite additionner -91 et 113 avec un codage sur 8 bits. On peut donc coder tout nombre entre -128 et 127.

- Le complément à 2 de $\overline{-91}^{10}$ est : $\overline{10100101}^{2}$.
- Le complément à 2 de $\overline{113}^{10}$ est : $\overline{01110001}^{2}$.

On pose l'opération :

$$\begin{array}{ccccccccccc}
 & & & 1 & 0 & 1 & 0 & 0 & 1 & 0 & 1 \\
+ & & & 0 & 1 & 1 & 1 & 0 & 0 & 0 & 1 \\
\hline
 & & 1 & 0 & 0 & 0 & 1 & 0 & 1 & 1 & 0
\end{array}$$

On ne garde que les 8 derniers bits : 00010110.
Le bit de poids fort étant 0, le résultat est positif. Ce dernier représente donc $\overline{00010110}^{2} = \overline{22}^{10}$.

1.3.2 - Opposé d'un nombre

Voyons sur un exemple comment trouver le complément à 2 d'un entier négatif.

Exemple : on souhaite trouver le complément à 2 de $\overline{-113}^{10}$ sur une mémoire de 8 bits.

- Le complément à 2 de $\overline{113}^{10}$ est $\overline{01110001}^{2}$.
- Le complément à 1 de $\overline{01110001}^{2}$ est $\overline{10001110}^{2}$ (on change tous les 0 en 1 et tous les 1 en 0).
- On ajoute 1 au binaire obtenu : $\overline{10001111}^{2}$.

Le complément à 2 de $\overline{-113}^{10}$ est donc $\overline{10001111}^{2}$.

1.3.3 - Soustraction

Calculer $a - b$ revient à calculer $a + (-b)$.
Ainsi, effectuer une soustraction revient à faire une addition.

1.4 Type entier en Python

La fonction `type()` permet de connaître le type d'une donnée. Si on tape :

```
a = -3
type(a)
```

on voit s'afficher :

```
<class 'int'>.
```

Cela signifie que le contenu de la variable a est bien considéré comme un entier.

En Python, le type d'une variable n'est pas spécifié explicitement. Le type dépend de la valeur qui lui est affectée. La fonction `int()` convertit une valeur en entier :

```
int(3/2)
```

La console affichera l'entier obtenu après conversion du résultat de la division de 3 par 2 (qui est initialement un nombre décimal : 1,5). La fonction `int()` extrait la partie entière en supprimant la partie décimale, sans effectuer d'arrondi.

Solution de l'exercice type 1

Lycée Sainte-Anne, Brest

1 Donner le complément à 2 du nombre 78 est le binaire signé correspondant car il est positif.

$$78 = 64 + 8 + 4 + 2 = 2^6 + 2^3 + 2^2 + 2.$$

On en déduit que $\overline{78}^{10} = \overline{01001110}^{2}$.

2 $111 = 64 + 32 + 8 + 4 + 2 + 1 = 2^6 + 2^5 + 2^3 + 2^2 + 2^1 + 2^0$ donc :

$$\overline{111}^{10} = \overline{01101111}^{2}.$$

Le complément à 1 de ce binaire est donc : $\overline{10010000}^{2}$.

En ajoutant 1, on trouve : $\overline{10010001}^{2}$.

Le complément à 2 de $\overline{-111}^{10}$ est donc $\overline{10010001}^{2}$.

3 Ajoutons les compléments à 2 des nombres 78 et -111 :

$$\begin{array}{cccccccccc} & 0 & 1 & 0 & 0 & 1 & 1 & 1 & 0 \\ + & 1 & 0 & 0 & 1 & 0 & 0 & 0 & 1 \\ \hline & 1 & 1 & 0 & 1 & 1 & 1 & 1 & 1 \end{array}$$

Le complément à 2 de la somme $78 + (-111)$ est donc $\overline{11011111}^{2}$.

4 $78 + (-111) = -33$. Or, $33 = 32 + 1 = 2^6 + 2^0$.

Le binaire signé de $\overline{33}^{10}$ est donc $\overline{00100001}^{2}$.

Le complément à 1 de ce binaire est : $\overline{11011110}^{2}$.

Le complément à 2 de ce binaire est alors : $\overline{11011111}^{2}$.

On obtient bien le même binaire que dans la question précédente.

Voir énoncé page 25

2 Représentation approximative des nombres réels

Exercice type 2 **Lycée Henri Bergson, Angers**

1 Donner la représentation binaire de la fraction $\frac{11}{16}$.

2 Donner la représentation binaire de la fraction $\frac{11}{15}$.

3 Donner le début du développement dyadique de la somme $\frac{11}{16} + \frac{11}{15}$.
Le nombre affiché par un ordinateur sera-t-il représentatif de cette somme ?

Voir corrigé page 34

2.1 Nombres décimaux (rappel)

Un nombre décimal est un nombre s'écrivant sous la forme $\frac{x}{10^n}$, où x est un entier relatif.
Posons :

$$x = \pm\, \overline{a_p a_{p-1} \dots a_1 a_0}^{\,10}.$$

Alors,

$$\frac{x}{10^n} = \pm\, \overline{a_p a_{p-1} \dots a_n, a_{n-1} \dots a_0 a_1}^{\,10}.$$

Exemple : $\frac{13}{4} = \frac{325}{100} = \frac{325}{10^2} = 3{,}25.$

2.2 Nombres dyadiques

Définition 4

Par analogie avec les nombres décimaux, on appelle *nombres dyadiques* les nombres qui s'écrivent sous la forme $\frac{x}{2^n}$, où $x \in \mathbb{Z}$ et $n \in \mathbb{N}$.

Exemple : $\frac{13}{4} = \frac{13}{2^2}$ est un nombre dyadique.

COURS

Pour obtenir le *développement dyadique* d'un nombre dyadique $\frac{x}{2^n}$, on prend le binaire correspondant à x et on insère une virgule avant le n-ième bit en partant de la fin.

Exemple : $\overline{13}^{10} = \overline{1101}^{2}$.
Le développement dyadique de $\frac{13}{2^2}$ est donc $\overline{11{,}01}^{2}$.

Cela signifie que :

$$\frac{13}{4} = 1 \times 2^1 + 1 \times 2^0 + 0 \times \frac{1}{2^1} + 1 \times \frac{1}{2^2}.$$

Il existe des nombres décimaux qui ne sont pas dyadiques, et par conséquent, qui n'admette pas de développement dyadique fini.

Exemple : $\overline{0{,}1}^{10} = \overline{0{,}0001100110011001100110011001100110011001100110011\ldots}^{2}$.

Par conséquent, leur représentation machine sera nécessairement tronquée, et cela engendrera des erreurs dans les opérations.
Sous Python par exemple, quand on tape :

```
0.1+0.2
```

la console nous retourne :

```
0.30000000000000004
```

ANECDOTE

Le 25 février 1991, à Dharan en Arabie Saoudite, un missile Patriot américain a raté l'interception d'un missile Scud irakien, ce dernier provoquant la mort de 28 personnes. La commission d'enquête chargée d'expliquer la raison de cet échec a mis en évidence le défaut suivant : l'horloge interne du missile Patriot mesure le temps en dixième de seconde. Pour obtenir le temps en seconde, le système multiplie ce nombre par 10 en utilisant un registre de 24 bits en virgule fixe. Or 1/10 n'est pas un nombre dyadique à développement fini donc a été arrondi : le registre de 24 bits contient $\overline{0{,}00011001100110011001100}^{2}$ et induit une erreur binaire de $\overline{0{,}0000000000000000000000011001100\ldots}^{2}$, soit approximativement $\overline{0{,}000\,000\,095}^{10}$.
En multipliant cette quantité par le nombre de dixièmes de seconde pendant 100 heures (le temps écoulé entre la mise en marche du système et le lancement du missile Patriot), on obtient le décalage entre l'horloge interne de missile et le temps réel, soit :

$$0{,}000\,000\,095 \times 100 \times 3\,600 \times 10 \approx 0{,}34 \text{ s}.$$

ANECDOTE

Or un missile Scud vole à la vitesse approximative de 1 676 m/s donc parcourt plus de 500 m en 0,34 s, ce qui le fait largement sortir de la zone d'acquisition de sa cible par le missile d'interception.

Source : http ://ta.twi.tudelft.nl/nw/users/vuik/wi211/disasters.html.

2.3 Norme IEEE-754 (complément d'information)

Tout comme il existe une écriture scientifique pour les nombres décimaux, écriture sous la forme $a \times 10^n$ où $1 \leqslant a < 10$ et $n \in \mathbb{Z}$, il existe une écriture équivalente en binaire, écriture de la forme $\overline{1{,}b_1b_2\cdots b_k}^{\,2}$.

Par exemple, $\dfrac{13}{4} = 3{,}25 = \overline{11{,}01}^{\,2} = \overline{1{,}101}^{\,2} \times 2^1$.

La suite de bits $b_1b_2\ldots b_k$ est appelée la *mantisse* du nombre, et l'exposant de 2 est tout simplement appelé l'*exposant*.

Sur 64 bits, un nombre dyadique sera codé de telle façon que :

- le bit fort représente son signe ;
- les 11 bits suivants représentent l'exposant ;
- les 52 autres bits représentent la mantisse.

Remarque : la connaissance précise de cette norme n'est pas au programme.

2.4 Nombres réels

De façon analogue aux nombres décimaux, un nombre réel x sera codé en binaire à l'aide de son écriture décimale.

Voyons la méthode à travers un exemple.

MÉTHODE

On pose $x = \dfrac{1}{3} \approx 0{,}333333\ldots$.

- $0{,}333333\ldots \times 2 = \boxed{0}{,}666666\ldots$
- $0{,}666666\ldots \times 2 = \boxed{1}{,}333333\ldots$
- On ne prend que la partie décimale du résultat précédent :
  $0{,}333333\ldots \times 2 = \boxed{0}{,}666666\ldots$
- etc.

Ainsi, la représentation binaire du nombre $\dfrac{1}{3}$ est $\overline{0{,}010101\ldots}^{\,2}$.

COURS

2.5 Flottants, Python et tests d'égalité

En Python, quand on tape :

```
type(1/3)
```

la console nous renvoie :

```
<class 'float'>
```

En effet, dans un cas général, en informatique, les nombres à virgule sont appelés des *nombres flottants* ou *nombres à virgule flottante*.

Pour les raisons d'approximations que nous avons vues précédemment, il faudra éviter les tests d'égalité entre deux flottants.
En effet, en Python, quand on tape :

```
0.1+0.2 == 0.3
```

la console renvoie :

```
False
```

ce qui signifie que pour Python, la représentation binaire du résultat de $0,1 + 0,2$ n'est pas identique à celle du nombre 0,3.

À RETENIR

En Python, le développement dyadique $\overline{0,111111\ldots}^{\,2}$ correspond au nombre 1 (cela a un rapport avec la norme IEEE-754 mais nos connaissances actuelles ne nous permettent pas encore de comprendre la raison pour laquelle on a ce résultat).

Solution de l'exercice type 2 — Lycée Henri Bergson, Angers

1 $\dfrac{11}{16}$ est un nombre dyadique et :

$$\frac{11}{16} = \frac{8+2+1}{16} = \frac{8}{16} + \frac{2}{16} + \frac{1}{16} = \frac{1}{2} + \frac{1}{8} + \frac{1}{16}.$$

Donc $\dfrac{11}{16} = \overline{0,1011}^{\,2}$.

2 $\dfrac{11}{15}$ n'est pas un nombre dyadique car son dénominateur n'est pas une puissance de 2.

$$\frac{11}{15} \approx 0,7333\ldots$$

Solution de l'exercice type 2 (suite) — Lycée Henri Bergson, Angers

- $0{,}7333\ldots \times 2 = \boxed{1}{,}4666\ldots$
- $0{,}4666\ldots \times 2 = \boxed{0}{,}9333\ldots$
- $0{,}9333\ldots \times 2 = \boxed{1}{,}8666\ldots$
- $0{,}8666\ldots \times 2 = \boxed{1}{,}7333\ldots$ ← on est revenu au point de départ.

Donc $\dfrac{11}{15} = \overline{0{,}\underline{1011}10111011\ldots}^{2}$.

Remarque : nous avons souligné la période du binaire, c'est-à-dire la partie qui se répète indéfiniment.

3 Le binaire de la somme $\dfrac{11}{16} + \dfrac{11}{15}$ est :

$$\begin{array}{cccccccccccc} & 0 & , & 1 & 0 & 1 & 1 & 0 & 0 & 0 & 0 & \ldots \\ + & 0 & , & 1 & 0 & 1 & 1 & 1 & 0 & 1 & 1 & \ldots \\ \hline & 1 & , & 0 & 1 & 1 & 0 & 1 & 0 & 1 & 1 & \ldots \end{array}$$

Ce binaire correspond au nombre :

$$1 \times 2^0 + 0 \times \frac{1}{2^1} + 1 \times \frac{1}{2^2} + 1 \times \frac{1}{2^3} + 0 \times \frac{1}{2^4} + 1 \times \frac{1}{2^5} + \cdots$$

Le résultat en base 10 affiché ne sera pas la valeur exacte de la somme $\dfrac{11}{16} + \dfrac{11}{15}$ car l'un des deux nombres n'est pas dyadique et son développement dyadique n'est pas fini.

Voir énoncé page 31

3 Les booléens

Exercice type 3 — Lycée Jean Macé, Niort

Si `a` et `b` sont deux booléens tels que `a=1` et `b=0`, que vaut le booléen :
`not(a or b) or (a and not(b))` ?

Voir corrigé page 38

3.1 Définition

Définition 5

Un *booléen* est un type de donnée qui n'a que deux états possibles, généralement nommés *vrai* et *faux* (respectivement `True` et `False` en Python), correspondant aux deux valeurs possibles pour un bit : 1 et 0.

COURS

En Python, on peut déclarer un booléen directement par sa valeur. Par exemple :

```
couleur = True
nb = False
```

3.2 Opérateur de comparaison

En programmation, un test est une expression logique qui compare des valeurs ou vérifie une condition. Un test retourne toujours un booléen. Par exemple,

```
a = 10
print(a > 4)
```

Le résultat affiché est alors : `True` car a est effectivement strictement supérieur à 4, puisqu'il vaut 10.
Les différents opérateurs de comparaison possibles sont les suivants :

Opérateur	Signification
<	strictement inférieur
<=	inférieur ou égal
>	strictement supérieur
>=	supérieur ou égal
==	égal
!=	différent

ATTENTION

Quand on veut tester une égalité, il faut faire attention à mettre deux signes « = » à la suite, soit == (un seul signe égal correspondant à l'affectation) :

```
a = 2048
b = 2**11
print(a == b)
```

Ce programme affiche « True » car 2 048 $= 2^{11}$.

3.3 Opérateurs booléens

3.3.1 - L'opérateur « and »

L'opérateur « and » permet de tester si deux booléens sont vrais simultanément.

On a alors la table suivante :

and	0	1
0	0	0
1	0	1

Exemple :

```
note = 13.5
mention_ab = note >=12 and note < 14
print(mention_ab)
```

Le booléen `mention_ab` est le résultat de : `note >= 12` (qui vaut *True*) ET `note < 14` (qui vaut *True*).
D'après la table précédente, `mention_ab = True`.

3.3.2 - L'opérateur « or »

L'opérateur « or » permet de tester si l'un des deux booléens au moins est vrai.

On a alors la table suivante :

or	0	1
0	0	1
1	1	1

Exemple :

```
note = 16.5
extreme = note < 5 or note > 15
print(extreme)
```

Ici, on teste si l'un ou l'autre des booléens `note < 5` et `note > 15` est vrai. C'est effectivement le cas donc le booléen `extreme` vaut `True`.

3.3.3 - L'opérateur « not »

L'opérateur « not » retourne le booléen opposé.

On a alors la table suivante :

	0	1
not	1	0

COURS
INTERROS
CORRIGÉS

COURS

Exemple :

```
couleur = True
nb = not couleur
```

Ici, on définit le booléen `nb` comme étant le contraire du booléen `couleur`. Ainsi, `nb = False`.

3.3.4 - L'opérateur « xor » (ou exclusif)

Cet opérateur retourne « vrai » quand seulement l'un des deux booléens est vrai, et « faux » sinon.

On a alors la table suivante :

xor	0	1
0	0	1
1	1	0

3.3.5 - En Python : le mot-clé « in »

Ce mot-clé permet de vérifier si une chaîne de caractères est contenue dans une autre. Une expression utilisant `in` retourne un booléen.

Exemple :

```
phrase = "Le dormeur du val est un poème d'Arthur
Rimbaud"
resultat = "val"
print(resultat in phrase)
```

Le programme affiche « True » car la chaîne de caractères « val » fait bien partie de la phrase.

Remarque : ce mot-clé est aussi utile pour des *listes*, mais nous en parlerons dans le chapitre suivant, à la page 82.

Solution de l'exercice type 3 — **Lycée Jean Macé, Niort**

Prenons les choses petit à petit :

- `(a or b)` = `(1 or 0)` = 1 ; donc `not(a or b)` = `not(1)` = 0 ;
- `not(b)` = `not(0)` = 1, donc `a and not(b)` = `1 and 1` = 1 ;

On en déduit que `not(a or b) or (a and not(b))` = `0 or 1` = 1.

Voir énoncé page 35

4 Les variables sous Python

Exercice type 4 — Lycée André Theuriet, Civray

Écrire un programme Python :

1. qui demande à l'utilisateur de saisir une phrase ;
2. qui convertit la saisie en minuscules ;
3. qui détecte si la chaîne « ui » apparaît dans la saisie ;
4. qui affiche la longueur de la saisie ;
5. qui affiche les 5 derniers caractères si la saisie est strictement plus longue que 5 caractères ;
6. qui remplace tous les « e » en « * » et qui affiche le résultat.

Voir corrigé page 44

4.1 Les types numériques

En Python, il est possible d'effectuer des opérations entre différents types numériques (entiers et flottants, par exemple). Python adapte automatiquement les types si nécessaire : lors d'une opération entre un entier et un flottant, le résultat est converti en flottant, comme dans cet exemple :

```
a = 5
b = 3.8
print(a+b)
```

La dernière ligne calcule la somme des valeurs contenues dans les deux variables, et affiche donc : « 8.8 ».

Rappelons ici les opérations de base sur les données numériques.

- `a += 1` : ajoute 1 à la variable `a`.
  On peut aussi écrire : `a = a+1`.
- `a -= 1` : enlève 1 à la variable `a`.
  On peut aussi écrire : `a = a-1`.
- `a *= 2` : multiplie par 2 la variable `a`.
  On peut aussi écrire : `a = a*2`.
- `a /= n` : divise par `n` la variable `a`.
  On peut aussi écrire : `a = a/n`.

COURS

- `a**n` : élève à l'exposant `n` la variable `a`.
  Par exemple, `2**3` donne « 8 » car $2^3 = 8$.
- `a**0.5` : calcule la racine carrée de la variable `a`.
  Par exemple, `9**0.5` donne « 3 » car $\sqrt{9} = 3$.
- `a//b` : donne le quotient de la division euclidienne de `a` par `b`.
  Par exemple, « 87//42 » donne « 2 » car $87 = 2 \times 42 + 3$.
- `a%b` : donne le reste de la division euclidienne de `a` par `b`.
  Par exemple, « 87%42 » donne « 3 » car $87 = 2 \times 42 + 3$.

4.2 Les chaînes de caractères

Sous Python, elles sont déclarées entre guillemets ou entre deux apostrophes, comme dans l'exemple suivant :

```
prenom = "Kévin"
surnom = 'kéké'
```

ATTENTION

En Python, il faut éviter de déclarer une chaîne de caractères avec des variables en majuscules.
Ainsi, « MaVille » est à éviter ; on préfèrera : « ma_ville ».
C'est plus une convention qu'une règle fondamentale ; les variables `MaVille` et `ma_ville` sont différentes : elles ne désignent pas la même chose. Ainsi, pour ne pas confondre les variables écrites avec ou sans majuscules, il est préférable de ne mettre aucune majuscule.

4.2.1 - Concaténation

Définition 6

La *concaténation* de deux chaînes de caractères est le fait de les mettre l'une à la suite de l'autre.

Exemple :

```
a = "Ludovic"
n = "Lulu"
phrase = a + ", surnommé " + b + ", habite Bordeaux."
print(phrase)
```

Ce programme affiche : « Ludovic, surnommé Lulu, habite Bordeaux. »

Si on veut écrire une centaine de fois une chaîne de caractères, on peut faire ainsi :

```
lettre = "A"
print(lettre*100)
```

4.2.2 - Longueur d'une chaîne de caractères

Dans de nombreux programmes, connaître la longueur d'une chaîne de caractères s'avère utile. Dans la plupart des langages de programmation, ce sera possible à l'aide de la fonction `len` (de l'anglais « length », signifiant *longueur*).

Remarque : on dit que `len` est une *apocope* du mot anglais « length ».

Exemple :

```
phrase = "Le dormeur du val ne se réveille pas."
print(len(phrase))
```

Ce programme affiche le nombre 37, nombre de caractères dans la chaîne `phrase`.

Remarque : cette fonction sert aussi à compter le nombre d'items dans une liste (nous en parlerons dans le chapitre suivant, page 81).

4.2.3 - Partie de chaînes de caractères

Si `chaine` est une chaîne de caractères alors :

- `chaine[0]` représente le premier caractère ;
- `chaine[1]` représente le deuxième caractère ;
- `chaine[-1]` représente le dernier caractère ;
- `chaine[-5:]` représente les cinq derniers caractères.

Exemple : `phrase="Le dormeur du val ne se réveille pas."`.

- `phrase[0]` représente « L » ;
- `phrase[1]` représente « e » ;
- `phrase[-1]` représente « . ».

On peut aussi aller au-delà du programme avec :

- `chaine[2:7]` représente la partie de la chaîne comprise entre le troisième et le huitième caractère ;
- `chaine[5:]` représente la chaîne privée des 5 premiers caractères.

Exemple : `phrase="Le dormeur du val ne se réveille pas."`.

- `phrase[3:9]` représente « dormeur » ;
- `phrase[-4:]` représente « pas. » ;
- `phrase[3:]` représente « dormeur du val ne se réveille pas. ».

COURS

4.2.4 - Le retour à la ligne

Quand on concatène plusieurs chaînes de caractères et que l'on souhaite revenir à la ligne à chaque concaténation, on utilisera « \n ».

Par exemple, "Barbara\nCassandre" donne :

Barbara

Cassandre

4.2.5 - Caractère d'échappement

Si une chaîne de caractères est définie entre deux apostrophes, et si elle contient elle-même une apostrophe, il est nécessaire d'« échapper » cette dernière.

Par exemple, si la chaîne de caractères est « Aujourd'hui », on la définira ainsi :

```
chaine = 'Aujourd\'hui'
```

Mais pour éviter ce cas de figure, on peut aussi définir la chaîne de caractères à l'aide de guillemets :

```
chaine = "Aujourd'hui"
```

4.2.6 - Majuscules et minuscules

Python est sensible à la casse d'une chaîne de caractères : cela signifie qu'il distingue les majuscules et les minuscules.

Ainsi, la chaîne « Bordeaux » est différente de la chaîne « bordeaux ». Il sera donc parfois utile de transformer une chaîne de caractères afin qu'elle ne contienne que des minuscules. On utilisera pour cela la méthode `lower`.

Exemple :

```
ville = input("Où habitez-vous ?")
if (ville.lower() != "bordeaux"):
    print("Vous ne connaissez sans doute pas la place
Pey-Berland !")
```

Dans ce programme, il est plus simple de transformer en minuscules la réponse saisie au clavier afin de la comparer à la chaîne de caractères « bordeaux » ; sans cela, il aurait fallu comparer toutes les écritures possibles de la ville (tout en majuscules, tout en minuscules ou en minuscule à part l'initiale, etc.).

Si on souhaite au contraire transformer la chaîne de caractères en majuscules, on utilisera la méthode `upper`.

Exemple :

```
ville = input("Où habitez-vous ?")
if (ville.upper() != "BORDEAUX"):
    print("Vous ne connaissez sans doute pas la place
Pey-Berland !")
```

Remarque : les caractères accentués sont aussi mis en majuscules. Par conséquent, la chaîne « Stéphane » sera transformée en « STÉPHANE » et non en « STEPHANE ».

4.2.7 - Remplacer un caractère par un autre

La méthode `replace(a,b)` permet de remplacer toutes les occurrences de `a` par `b`.

Exemple :

```
chaine = "Levez-vous vite, orages désirés."
chaine.replace('e','Z')
```

Python renvoie :

```
'LZvZz-vous vitZ, oragZs désirés.'
```

4.3 Types et conversions

4.3.1 - Connaître le type d'une variable

Si l'on ne sait pas comment est interprétée une variable par Python, on peut lui demander d'afficher son type comme ceci :

```
chiffre = "3"
print(type(chiffre))
```

s'affichera alors : « `<class 'str'>` », ce qui signifie que la variable contient un type « string », c'est-à-dire « caractère ».
En revanche, si l'on tape :

```
chiffre = 3
print(type(chiffre))
```

s'affichera la réponse : « `<class 'int'>` », qui signifie « integer », soit « entier ».
Et si on tape :

```
n = 3.7
print(type(n))
```

on verra s'afficher la réponse : « `<class 'float'>` », pour « flottant ».

COURS

4.3.2 - Changer le type d'une variable

Il sera parfois utile de transformer une variable de type caractère en une variable de type nombre, ne serait-ce que pour effectuer des calculs numériques, et inversement.

On peut alors faire comme dans l'exemple suivant :

```
chiffre, nombre = "3", 3.7
chiffre = int(chiffre)
print(chiffre+nombre)
```

Ici, nous admettons que nous partons d'une variable `chiffre` de type caractère et nous souhaitons effectuer la somme des variables `chiffre` et `nombre`. Sans transformer le type de la variable `chiffre`, cette opération mène à une erreur.

En tapant « `chiffre=int(chiffre)` », on transforme le type « caractère » en type « entier », et l'addition peut alors s'effectuer.

Pour transformer une variable entière en caractères, on procède de la même manière :

```
chiffre, nombre = "3", 3.7
nombre = str(nombre)
print(chiffre+nombre)
```

La réponse affichera : « 33.7 » car les variables `chiffre` et `nombre` sont interprétées comme des chaînes de caractères ; ainsi, l'instruction « `chiffre+nombre` » concatène les deux variables et l'instruction « print » affiche cette concaténation.

Solution de l'exercice type 4 — Lycée André Theuriet, Civray

```
phrase = input("Entrez une phrase : ")
phrase = phrase.lower()

if "ui" in phrase:
  print("La chaîne 'ui' apparaît dans la saisie.")
else:
  print("La chaîne 'ui' n'apparaît pas dans la saisie."
    )

print("Il y a",len(phrase),"caractères dans la phrase
    saisie.")

if len(phrase)>5:
  print(phrase[-5:])

phrase.replace('e','*')
print(phrase)
```

Voir énoncé page 39

1 QCM Les types de données

5 min Corrigé p. 59

Pour chacune des questions suivantes, plusieurs propositions sont faites mais une seule est correcte. Laquelle ?

1 La réponse à l'instruction « `1 == 3` » est :

- a une variable numérique ;
- b une chaîne de caractères ;
- c un booléen.

2 On souhaite mettre le résultat de l'opération « $9 - 3 + (7 - 2) * 4$ » dans une variable `result`. Le type de variable le plus adapté de `result` est :

- a la classe `str` ;
- b la classe `int` ;
- c la classe `bool`.

3 Dans un programme Python, si `a = "8"` et et `b = 5`, que retourne ce programme en tapant : « `a+b` » ?

- a '13' ;
- b 13 ;
- c False ;
- d TypeError : must be str, not int.

4 Clément a 100 € dans sa tirelire. Chaque semaine, il dépense 10% de ce qu'il y a dans sa tirelire. Il souhaite savoir combien il possèdera après 10 semaines.

La classe Python de la variable représentant la somme de Clément est :

- a la classe `int` ;
- b la classe `float` ;
- c la classe `bool` ;
- d la classe `str`.

2 QCM Des programmes en Python

10 min Corrigé p. 59

Cet exercice est un QCM où chaque question comporte 4 ou 5 réponses. Une seule est bonne.

1 Qu'affiche le programme suivant ?

```
somme = 0
for i in range(5):
    somme = i
print(somme, end="")
```

- a 5
- b 4
- c 10
- d 0 1 2 3 4

COURS INTERROS CORRIGÉS

INTERROS

2 Parmi les programmes suivants, écrits en Python, quel est celui qui donne les valeurs de $y = x^2 + 3x + 1$ pour x de 1 à 20 ?

a
```python
def parabole(x):
    y = x**2 + 3*x + 1
x = 1
for i in range(20):
    x = x + i
    print(parabole(x))
```

b
```python
def parabole(x):
    y = x**2+3*x + 1
    return
x = 1
for i in range(21):
    x = x + i
    print(parabole(x))
```

c
```python
def parabole(x):
    y = x**2 + 3*x + 1
    return (x, y)
x = 1
for i in range(20):
    x = i
    print(parabole(x))
```

d
```python
def parabole(x):
    y = x**2 + 3*x + 1
    return (y)
x = 1
for i in range(1, 21):
    x = i
    print(parabole(x))
```

e
```python
def parabole(x):
    y = x**2 + 3*x + 1
    return y
for i in range(20):
    x = i
    print(parabole(x))
```

3 Que donne l'exécution du programme suivant ?

```python
x = 5
y = 4
if x == y - 1:
    y = x
else:
    x = y
if y == x - 1:
    x = x + 1
else:
    y = y + 1
```

a $x = 5$ et $y = 5$

b $x = 4$ et $y = 4$

c $x = 5$ et $y = 4$

d $x = 4$ et $y = 5$

e Autre réponse

4 Quel est le type de la valeur contenue dans la variable suivante ?

```
nombre = "3.0"
```

a `str` (chaîne)

b `float`

c `int` (entier)

d Autre réponse

5 Quelle valeur contient la variable x après exécution du programme suivant ?

```
x = 400
while x >= 25:
    x = x // 2
```

a 25

b 200

c 12

d 13

e 12.5

6 Que fait le programme suivant ?

```
somme = 0.1 + 0.1 + 0.1
if somme == 0.3:
    print("somme =", somme)
else:
    print("résultat =", somme)
```

a `somme = 0.3`

b `somme = 0.1 + 0.1 + 0.1`

c `résultat = 0.1 + 0.1 + 0.1`

d `résultat = 0.30000000000000004`

e Un message d'erreur

INTERROS

3 V/F Codage des nombres

10 min Corrigé p. 60

Les propositions suivantes sont-elles vraies ou fausses ?

1. Le nombre $\overline{1{,}25}^{10}$ a un développement dyadique fini.
2. En Python, quand on tape `1.2+1.25`, le résultat affiché est `1.45`.
3. $\overline{1{,}01}^{2}$ représente le nombre 1,25.
4. En Python, quand on tape `3*1/3`, il s'affiche « `1.0` ».

4 V/F Connaissances générales

5 min Corrigé p. 61

Les propositions suivantes sont-elles vraies ou fausses ?

1. Si `mot = "informatique"` alors `mot[2]="n"`.
2. Si `chaine = "Un signe."` alors `chaine[-1] = "."`
3. Si `rep = (8>4 or 1<2)` alors `rep = True`.
4. Si `mot = "éléphant"` alors `mot.upper() = "ELEPHANT"`.

Codage de nombres

5 Complément à 2

★ 10 min Corrigé p. 61

Lycée François Mauriac, Bordeaux

Donner le complément à 2 des nombres suivants avec une mémoire de 8 bits :

1. 100
2. 75
3. −50
4. −89

6 Somme de deux entiers

★ 15 min Corrigé p. 62

Lycée François Villon, Les Mureaux

Pour chacune des questions suivantes, avec une mémoire de 8 bits,

- donner le complément à 2 des deux nombres a et b ;
- calculer la somme des compléments à 2 de a et b ;
- vérifier que le résultat trouvé est bien le complément à 2 de la somme de a et b.

1. $a = 35$ et $b = 65$.
2. $a = -12$ et $b = 45$.
3. $a = -84$ et $b = 29$.
4. $a = -17$ et $b = -111$.

7 Nombres dyadiques

★ 10 min Corrigé p. 63

Lycée polyvalent Émile Zola, Aix-en-Provence

Coder en binaire les nombres suivants :

1 $\frac{75}{16}$ **2** $\frac{101}{8}$ **3** $\frac{59}{4}$ **4** $\frac{61}{2}$

8 À partir du complément à 2

★★ 10 min Corrigé p. 63

Lycée Camille Jullian, Bordeaux

Trouver le nombre en base 10 dont le complément à 2 est :

1 $\overline{11001011}^2$ **2** $\overline{11010100}^2$ **3** $\overline{10000010}^2$ **4** $\overline{10101010}^2$

Booléens

9 La valeur des booléens

★ 5 min Corrigé p. 65

Lycée François Mauriac, Bordeaux

Dans chacun des cas suivants, donnez la valeur du booléen `rep`.

1
```
x = -3
rep = x**2 == -9
```
2
```
x, y, z = 3, 4, 5
rep = x**2 + y**2 == z**2
```
3
```
a, b = 3, -7
rep = a**3 > 50 and b**2 < 50
```
4
```
a, b = 3, -7
rep = (a**3 > 50 and b**2 < 50) or
                      (a**2 < 10 and b**2 >10)
```

10 Booléens et chaînes de caractères

★ 5 min Corrigé p. 65

Lycée polyvalent Émile Zola, Aix-en-Provence

1 Qu'affiche le programme suivant ?

```
phrase1 = "il est actuellement midi."
phrase2 = "il est temps d'aller manger."
"il" in (phrase1 and phrase2)
```

INTERROS

2 Qu'affiche le programme suivant ?

```
phrase1 = "il est actuellement midi."
phrase2 = "il est temps d'aller manger."
"est" in (phrase1 and phrase2) and "na" in (phrase1
    or phrase2)
```

11 Opérations sur des booléens

★★ 5 min Corrigé p. 65

Lycée Camille Claudel, Blois

Compléter le tableau suivant :

a	b	not(a) and b	not(a) or b	not(a) and not(b)	not(a) or not(b)
0	0				
0	1				
1	0				
1	1				

12 Encore des opérations

★★ 10 min Corrigé p. 66

Lycée Turgot, Paris

1 Compléter le tableau suivant :

a	b	(a or b) and (a and b)
0	0	
0	1	
1	0	
1	1	

Comment peut-on alors simplifier la règle « (a or b) and (a and b) » ?

2 Compléter le tableau suivant :

a	b	(a or b) or (a and b)
0	0	
0	1	
1	0	
1	1	

Comment peut-on alors simplifier la règle « (a or b) or (a and b) » ?

13 La loi « Delta »

★★ 10 min Corrigé p. 66

Lycée Sainte-Anne, Brest

Soient a et b deux booléens. On définit la loi `Delta` telle que :

`a Delta b = (a or b) and not(a and b)`

1 Compléter la table de la loi Delta suivante :

Delta	0	1
0		
0		

2 Que se cache-t-il derrière la loi Delta ?

14 La fonction « Delta »

★★ 15 min Corrigé p. 67

Lycée Sophie Germain, Paris

On considère le programme suivant :

```
def delta(A,B,x):
    return (x in (A or B)) and not(x in (A and B))

A = "ABCDEFGHIJKLM"
B = "OKLM"

print(delta(A,B,"O"))
```

1 Qu'affiche-t-il ?

2 Trouver une valeur de x pour qu'il affiche « True ».

3 Que se passe-t-il quand x = « A » si on définit la fonction « Delta » de la manière suivante :

```
def delta(A,B,x):
    return (x in A or B) and not(x in A and B)
```

Chaînes de caractères

15 Remplacer une lettre

★ 5 min Corrigé p. 67

Lycée Camille Jullian, Bordeaux

On souhaite écrire un programme Python remplaçant la lettre « e » par la lettre « A » dans une chaîne de caractère `A`, sans utiliser la fonction `replace`.
Compléter le programme page suivante afin d'obtenir un tel programme.

COURS
INTERROS
CORRIGÉS

INTERROS

```
A = "Levez-vous vite, orages désirés qui devez emporter
     René dans les espaces d'une autre vie !"
B = ""

for i in range(...):
    if ... :
        B += 'a'
    else:
        B += A[i]

print(B)
```

16 Nombre de « a »

★ 10 min Corrigé p. 68

Lycée François Mauriac, Bordeaux

Écrire une fonction `nombre_de_a(chaîne)` qui admet une chaîne de caractères comme argument, et compte le nombre de caractères « a » ou « A » dans la chaîne.

On rappelle qu'une chaîne est un type particulier de liste, que `len(chaîne)` donne la longueur de la chaîne et que `chaîne[i]` permet d'accéder à l'élément d'indice `i` de la chaîne.

- `nombre_de_a("bonjour Ernest")` renvoie 0.
- `nombre_de_a("Salut Bernard")` renvoie 2.
- `nombre_de_a("Au revoir Anatole")` renvoie 3.

17 Modification

★ 5 min Corrigé p. 68

Lycée François Villon, Les Mureaux

On considère le programme suivant :

```
A = "Héautontimorouménos"
B = ""
for i in range(len(A)):
    if (i%2 == 0):
        B += A[i]
    else:
        B += "*"
print(B)
```

Compléter le tableau d'exécution de la page suivante.

Valeurs de `i`	...	...	...
Valeurs de `B`	...	...	...

En déduire ce qu'affiche ce programme.

18 Manipulation

★ 10 min Corrigé p. 69

Lycée Camille Claudel, Blois

On considère le programme suivant :

```
A = "Horloge"
B = ""

for i in range(len(A)-1,-1,-1):
    B += A[i]

print(B)
```

Compléter le tableau d'exécution suivant :

Valeurs de `i`	...	...	...
Valeurs de `A[i]`	...	...	...
Valeurs de `B`	...	...	...

En déduire ce qu'affiche ce programme.

19 Les initiales

★★ 10 min Corrigé p. 69

Lycée André Theuriet, Civray

Créer un programme Python qui :

- demande à l'utilisateur son nom et son prénom, séparés par un espace ;
- affiche : « Votre prénom est : » suivi du prénom ;
- affiche : « Votre nom est : » suivi du nom ;
- affiche : « Vos initiales sont : » suivi des initiales du nom et du prénom en majuscules.

Aide : vous pourrez recourir à un booléen pour détecter la présence de l'espace.

COURS
INTERROS
CORRIGÉS

20 Que fait ce programme ?

★★ 10 min Corrigé p. 70

Lycée Henri Alain-Fournier, Bourges

On considère le programme Python suivant :

```
deb = "C'est un trou de verdure où chante une rivière."
lettre = "r"
fin = ""
for i in deb:
    if i != lettre:
        fin += i
print(fin)
```

1. Quel est le type des variables `debut`, `lettre` et `fin` ?
2. Qu'affiche ce programme ?
3. Modifier ce programme afin que l'on puisse entrer n'importe quelle phrase et que l'on puisse choisir le caractère désigné par la variable `lettre`.

21 Donner le rang d'un caractère

★★★ 10 min Corrigé p. 71

Lycée Turgot, Paris

On souhaite créer un programme donnant tous les index d'un caractère donné dans une chaîne. On souhaite de plus afficher le message : « rang pair » dans le cas où l'index est pair.

Pour cela, on utilise le programme suivant :

```
phrase = input('Entrez une phrase : ')
lettre = input('Entrez le caractère à trouver : ')
presence = False
for i in range(len(phrase)):
    if phrase[i] == lettre:
        print("Le caractère",lettre,"est trouvé à l'
    index",str(i)+".")
        presence = True
if presence == False:
    print("Le caractère",lettre,"n'a pas été trouvé.")
```

1. À quoi sert le booléen `presence` ?
2. Expliquer la syntaxe de la ligne :

```
print("Le caractère",lettre,"est trouvé à
l'index",str(i)+".").
```

3 Expliquer pourquoi le programme suivant ne convient pas :

```
phrase = input('Entrez une phrase : ')
lettre = input('Entrez le caractère à trouver : ')
presence = False
for i in phrase:
    if i == lettre:
        print("Le caractère",lettre,"est trouvé à l'
    index",str(phrase.index(i))+".")
        presence = True
if presence == False:
    print("Le caractère",lettre,"n'a pas été trouvé.
    ")
```

Programmation et fonctions

22 Dans la tirelire de Clément ★ 5 min

Corrigé p. 72

Lycée Raoul Follereau, Nevers

Clément a 100 € dans sa tirelire. Chaque semaine, il dépense 10 % de ce qu'il y a dans sa tirelire. Il souhaite savoir combien il possèdera après 10 semaines.

Compléter le programme suivant afin qu'il affiche le résultat souhaité.

```
somme = ...

for i in range(...):
    somme *= ...

print(somme)
```

23 Somme des premiers entiers ★ 5 min

Corrigé p. 72

Lycée Sainte-Clotilde, Strasbourg

On souhaite créer une fonction Python qui affiche le résultat de l'opération :

$$1 + 2 + 3 + 4 + 5 + 6 + \cdots + n$$

où n est l'argument de la fonction.

INTERROS

Compléter le programme suivant afin de satisfaire cette requête.

```
def somme(n):
    s = ...
    for i in range(...):
        ...
    return s

s = somme(100)
print(s)
```

24 Le programme mystère

★★ 10 min Corrigé p. 73

Lycée Sainte-Anne, Brest

Que fait le programme Python suivant ?

```
phrase = "C'est ainsi que le cygne fit signe"
lettres = ""

for i in range(len(phrase)):
    presence = False
    for j in (range(len(lettres))):
        if phrase[i] == lettres[j]:
            presence = True
    if presence == False:
        if i != 0:
            lettres += " ; "+phrase[i]
        else:
            lettres += phrase[i]

print(lettres)
```

25 Le jeu du « plus » ou « moins »

★★★ 15 min Corrigé p. 73

Lycée Henri Alain-Fournier, Bourges

Retrouvez le corrigé de cet exercice en vidéo.

Le jeu du « plus » ou « moins » consiste à deviner un nombre en faisant de multiples propositions de nombres, auxquelles une personne répond par « plus » si le nombre à deviner est plus grand que la proposition, et par « moins » si le nombre à deviner est moins grand.

Nous souhaitons créer un programme Python qui :

- choisit au hasard un nombre entier entre 1 et 100 (compris) ;

- demande un nombre autant de fois que nécessaire jusqu'à trouver le nombre choisi au hasard ;
- affiche « plus » quand le nombre à trouver est supérieur à la proposition faite ;
- affiche « moins » quand le nombre à trouver est inférieur à la proposition faite ;
- compte le nombre de propositions faites et l'affiche à la fin.

Pour cela, il faudra faire appel au module `random` qui permet de choisir un nombre entier entre deux nombres a et b comme ceci :

```
import random
nombre = random.randint(a,b)
```

Il nous faut aussi utiliser une boucle `while`.

Proposer un tel programme.

26 Somme de nombres

★★★ 20 min Corrigé p. 74

Lycée François Mauriac, Bordeaux

On donne une chaîne de caractères, dans laquelle figurent également des nombres. Écrire une fonction `somme(chaîne)` qui renvoie la somme des nombres figurant dans la chaîne. On rappelle qu'une chaîne est un type particulier de liste, que `len(chaîne)` donne la longueur de la chaîne, que `chaîne[i]` permet d'accéder à l'élément d'indice `i` de la chaîne et que `chaine.isdigit()` est un booléen renvoyant `True` si `chaine` est un nombre.

Exemples :

- `somme("j'ai mangé 12 œufs")` renvoie 12 ;
- `somme("j'ai mangé 3 œufs")` renvoie 3 ;
- `somme("j'ai mangé 12 œufs,1 bœuf,5 gâteaux et un chapeau")` renvoie 18.

27 Black Jack

★★★ 30 min Corrigé p. 75

Lycée Sophie Germain, Paris

Dans cet exercice, il vous est demandé de programmer une version simplifiée du black jack (jeu de casino).

Règles : deux joueurs jouent l'un contre l'autre. L'un des joueurs est humain, l'autre est l'ordinateur. Le but du jeu est de marquer plus de points que l'autre, sans dépasser 21, qui est la plus grande valeur possible. Les cartes de 2 à 10 ont leur valeur (donc de 2 à 10 points). Le valet, la dame et le roi valent respectivement 11, 12 et 13 points. Dans un premier temps, l'as (le 1) vaut 11 points. Toutes les cartes sont tirées face visible.

- Les joueurs tirent d'abord une première carte, face visible.
- Ensuite, le joueur humain tire autant de cartes qu'il veut. S'il dépasse 21, il a perdu.
- Le deuxième joueur – l'ordinateur – tire ensuite autant de cartes qu'il veut. S'il dépasse 21, il a perdu.
- Si un des joueurs a plus de points que l'autre, il a gagné. Si les deux joueurs ont le même nombre de points, il y a égalité.

1 Écrire une première fonction qui tire un nombre aléatoire entre 1 et 13, et renvoie les points correspondant à la valeur de la carte. On considérera que 11, 12 et 13 correspondent respectivement au valet, à la dame et au roi.

2 Écrire une deuxième fonction qui initialise le jeu en tirant une carte pour le joueur humain et une carte pour l'ordinateur. Cette fonction renverra les points correspondant à chaque carte. On rappelle que l'on peut faire un retour multiple sur les fonctions :

```
truc, bidule = fonction(machin , chose, autre).
```

3 Écrire une troisième fonction qui fait jouer le joueur humain. Cette fonction renvoie le nombre de points du joueur humain.

4 Écrire une quatrième fonction qui fait jouer l'ordinateur. Vous programmerez une intelligence artificielle (rudimentaire) qui décide de la stratégie de l'ordinateur. Vous expliquerez cette stratégie dans les spécifications de la fonction. Cette fonction renvoie le nombre de points de l'ordinateur.

5 Écrire le programme principal, qui organise la partie et détermine le vainqueur.

1 QCM Les types de données

Énoncé p. 45

1 Réponse $\boxed{\text{c}}$. En effet, quand on tape l'instruction « `1 == 3` » en Python, on cherche à savoir si « 1 » est égal à « 3 » ; la réponse ne peut être que *True* ou *False* : c'est donc un booléen. Ici, Python renvoie *False* car « 1 » n'est bien entendu pas égal à « 3 ».

2 Réponse $\boxed{2}$. En effet, $9 - 3 + (7 - 2) * 4 = 26$ et « 26 » est un entier. Donc le type `int` est le plus adapté, même s'il est tout à fait possible de stocker ce résultat dans une chaîne de caractères en le convertissant à l'aide de la fonction `str`.

3 Réponse $\boxed{\text{d}}$. En effet, `a` est du type *chaîne de caractères* et `b` est du type *entier*. Or, on ne peut pas faire d'opérations entre deux variables de types différents. Python affiche donc qu'il y a une erreur.

Remarque : penser que Python renvoie *False* laisse à supposer que l'opération est possible à faire et qu'elle est fausse. Cela n'a donc ici aucun sens.

4 Réponse $\boxed{\text{b}}$. Ici, il est nullement question de trouver le résultat mais il faut anticiper sur le type de résultat. On sait que si on retire à un nombre 10% de ce nombre, on doit le multiplier par $\left(1 - \frac{10}{100}\right)$, c'est-à-dire par 0,9. Il y a donc fort à parier que les résultats seront des nombres réels et non entiers.

Par conséquent, la classe du résultat est `float`.

2 QCM Des programmes en Python

Énoncé p. 45

1 Réponse $\boxed{\text{b}}$. Le programme modifie la valeur de `somme` dans chaque itération. À la fin de la boucle, `somme` contient la dernière valeur de `i`, soit 4.

2 Réponse $\boxed{\text{d}}$.

Justification : le code $\boxed{\text{d}}$ retourne la valeur de y, calculée comme $y = x^2 + 3x + 1$, et parcourt les valeurs de x de 1 à 20 sans erreurs. Les autres codes comportent des erreurs, telles que l'absence de retour explicite ou une boucle mal configurée.

3 Réponse $\boxed{\text{d}}$. Le programme modifie les valeurs de x et y en fonction des conditions. Analyse :

- Initialement $x = 5$, $y = 4$.
- La première condition (`if x == y - 1`) est fausse, donc `y = x`, ce qui donne $x = 4$, $y = 4$.
- La deuxième condition (`if y == x - 1`) est fausse, donc $y = y + 1$, ce qui donne $x = 4$, $y = 5$.

CORRIGÉS

4 Réponse $\boxed{\text{a}}$. La variable `nombre` contient `"3.0"`, une chaîne de caractères.

Justification : les guillemets indiquent que `"3.0"` est de type `string`, même s'il ressemble à un nombre.

5 Réponse $\boxed{\text{c}}$. Après exécution, x contient la dernière valeur calculée par la boucle `while`.

Analyse : x commence à 400 et est divisé par 2 tant qu'il est $\geqslant 25$:

$$x = 400 \to 200 \to 100 \to 50 \to 25 \to 12.$$

La boucle s'arrête lorsque $x < 25$.

6 Réponse $\boxed{\text{d}}$. En Python, la somme `0.1 + 0.1 + 0.1` n'est pas strictement égale à 0.3 à cause des erreurs d'arrondi en virgule flottante.

Résultat affiché : `résultat = 0.30000000000000004`.

3 V/F Codage des nombres

Énoncé p. 48

1 La proposition est *Vraie*. En effet, $1{,}25 = \dfrac{125}{100} = \dfrac{5}{4}$, et 4 est une puissance de 2. Donc le $\dfrac{5}{4}$ est un nombre dyadique et par conséquent, son développement dyadique est fini.

2 La proposition est *fausse*. En effet, $\overline{1{,}2}^{10}$ n'est pas dyadique car $1{,}2 = \dfrac{12}{10} = \dfrac{6}{5}$, et 5 n'est pas une puissance de 2. Par conséquent, le développement dyadique de 1,2 est tronqué avec Python, et le résultat affiché n'est qu'une valeur approximative.

3 La proposition est *vraie*. En effet, $\overline{1{,}01}^{2}$ représente le résultat de $1 \times 2^0 + 0 \times 2^{-1} + 1 \times 2^{-2}$, c'est-à-dire $1 + 0{,}25$, soit 1,25.

4 La proposition est *vraie*. $\dfrac{1}{3} = \overline{0{,}010101\ldots}^{2}$ (voir cours) donc $\dfrac{1}{3} + \dfrac{1}{3}$ sera codé par :

$$\begin{array}{cccccccc} & 0 & , & 0 & 1 & 0 & 1 & \ldots \\ + & 0 & , & 0 & 1 & 0 & 1 & \ldots \\ \hline & 0 & , & 1 & 0 & 1 & 0 & \ldots \end{array}$$

et donc $\dfrac{2}{3} + \dfrac{1}{3}$ est codé par :

$$\begin{array}{cccccccc} & 0 & , & 1 & 0 & 1 & 0 & \ldots \\ + & 0 & , & 0 & 1 & 0 & 1 & \ldots \\ \hline & 0 & , & 1 & 1 & 1 & 1 & \ldots \end{array}$$

On reconnaît ici le développement dyadique de 1 (voir page 34).

C'est la raison pour laquelle le résultat affiché est bien « `1.0` ».

4 V/F Connaissances générales

Énoncé p. 48

1 Si `mot = "informatique"` alors `mot[2] = "n"`.

Réponse : *faux*. En effet, il ne faut pas oublier que l'indexation se fait à partir de 0. Donc `mot[0] = "i"` et par suite, `mot[2] = "f"`.

2 Si `chaine = "Un signe."` alors `chaine[-1] = "."`

Réponse : *vrai*. En effet, `chaine[-1]` désigne le dernier caractère de la chaîne `chaine`.

3 Si `rep = 8>4 or 1<2` alors `rep = True`.

Réponse : *vrai*. En effet, `rep` est un booléen, et la présence de « `or` » stipule qu'il vaut *True* si l'une des conditions au moins est vraie, ce qui est le cas car même si « 8 > 4 » est faux, « 1<2 » est vrai.

4 Si `mot = "éléphant"` alors `mot.upper() = "ELEPHANT"`.

Réponse : *faux*. La commande `upper` met certes en majuscules, mais la majuscule de « é » est « É » ; il en résulte que `mot.upper() = "ÉLÉPHANT"`.

5 Complément à 2

Énoncé p. 48

Lycée François Mauriac, Bordeaux

1 $100 > 0$ donc le complément à 2 de 100 est le binaire signé correspondant.

$$100 = 64 + 32 + 4 = 2^6 + 2^5 + 2^2$$

donc $\overline{100}^{10} = \overline{01100100}^{2}$ (le bit fort est 0 car 100 est positif).

2 $75 = 64 + 8 + 2 + 1 = 2^6 + 2^3 + 2^1 + 2^0$ donc $\overline{75}^{10} = \overline{01001011}^{2}$.

3 $-50 < 0$ donc on suit les étapes suivantes :

- $50 = 32 + 16 + 2 = 2^5 + 2^4 + 2^1$ donc $\overline{50}^{10} = \overline{00110010}^{2}$.
- On prend le complément à 1 : $\overline{11001101}^{2}$.
- On prend le complément à 2 : $\overline{11001110}^{2}$.

Le complément à 2 de $\overline{-50}^{10}$ est $\overline{11001110}^{2}$.

4
- $89 = 64 + 16 + 8 + 1 = 2^6 + 2^4 + 2^3 + 2^0$ donc $\overline{89}^{10} = \overline{01011001}^{2}$.
- On prend le complément à 1 : $\overline{10100110}^{2}$.
- On prend le complément à 2 : $\overline{10100111}^{2}$.

Le complément à 2 de $\overline{-89}^{10}$ est donc $\overline{10100111}^{2}$.

COURS INTERROS CORRIGÉS

CORRIGÉS

6 Somme de deux entiers

Énoncé p. 48

Lycée François Villon, Les Mureaux

1
- $a = 35 = 32 + 2 + 1$ donc le complément à 2 de $\overline{35}^{10}$ est $\overline{00100011}^{2}$.
- $b = 65 = 64 + 1$ donc le complément à 2 de $\overline{65}^{10}$ est $\overline{01000001}^{2}$.

$$\begin{array}{cccccccccc} & & 0 & 0 & 1 & 0 & 0 & 0 & 1 & 1 \\ + & & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 1 \\ \hline & & 0 & 1 & 1 & 0 & 0 & 1 & 0 & 0 \end{array}$$

Ainsi, $\overline{00100011}^{2} + \overline{01000001}^{2} = \overline{01100100}^{2}$, qui correspond au nombre $2^6 + 2^5 + 2^2 = 64 + 32 + 4 = 100$, qui est bien la somme de 35 et 65.

2
- $a = -12$. $12 = 8 + 4$ donc $\overline{12}^{10} = \overline{00001100}^{2}$.
  Le complément à 1 est $\overline{11110011}^{2}$.
  Le complément à 2 est alors $\overline{11110100}^{2}$.
- $45 = 32 + 8 + 4 + 1$ donc $\overline{45}^{10} = \overline{00101101}^{2}$.

$$\begin{array}{cccccccccc} & & 1 & 1 & 1 & 1 & 0 & 1 & 0 & 0 \\ + & & 0 & 0 & 1 & 0 & 1 & 1 & 0 & 1 \\ \hline & 1 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 1 \end{array}$$

On ne prend que les 8 derniers bits donc le complément à 2 de la somme est $\overline{00100001}^{2}$. Le bit fort étant 0, la somme est positive, et $\overline{00100001}^{2}$ représente donc $2^5 + 2^0 = 33$, qui est bien égal à $12 + 45$.

3
- $84 = 64 + 16 + 4$ donc $\overline{84}^{10} = \overline{01010100}^{1}$.
  Le complément à 1 est $\overline{10101011}^{2}$.
  Le complément à 2 est : $\overline{10101100}^{2}$.
  Le complément à 2 de $\overline{-84}^{10}$ est donc $\overline{10101100}^{2}$.
- $29 = 16 + 8 + 4 + 1$ donc $\overline{29}^{10} = \overline{00011101}^{2}$.

$$\begin{array}{ccccccccc} & 1 & 0 & 1 & 0 & 1 & 1 & 0 & 0 \\ + & 0 & 0 & 0 & 1 & 1 & 1 & 0 & 1 \\ \hline & 1 & 1 & 0 & 0 & 1 & 0 & 0 & 1 \end{array}$$

Le complément à 2 de la somme est donc $\overline{11001001}^{2}$. Le bit fort étant 1, elle correspond à un nombre négatif en base 10.

$a + b = -84 + 29 = -55$. Or, $55 = 32 + 16 + 4 + 2 + 1$ donc $\overline{55}^{10} = \overline{00110111}^{2}$.

Le complément à 1 est $\overline{11001000}^{2}$.

Le complément à 2 est alors $\overline{11001001}^{2}$, qui correspond bien à ce que nous avons trouvé précédemment.

4
- $17 = 16 + 1$ donc $\overline{17}^{10} = \overline{00010001}^{2}$.
  Le complément à 1 est $\overline{11101110}^{2}$.
  Le complément à 2 est alors $\overline{11101111}^{2}$.
- $111 = 64 + 32 + 8 + 4 + 2 + 1$ donc $\overline{111}^{10} = \overline{01101111}^{2}$.
  Le complément à 1 est $\overline{10010000}^{2}$.
  Le complément à 2 est alors $\overline{10010001}^{2}$.

$$\begin{array}{ccccccccccc} & & & 1 & 1 & 1 & 0 & 1 & 1 & 1 & 1 \\ + & & & 1 & 0 & 0 & 1 & 0 & 0 & 0 & 1 \\ \hline & & 1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \end{array}$$

On ne retient que les 8 derniers bits : $\overline{10000000}^{2}$, qui correspond à -128 (voir page 28), qui est bien la somme de -17 et -111.

7 Nombres dyadiques

Énoncé p. 49

Lycée polyvalent Émile Zola, Aix-en-Provence

1 $$\frac{75}{16} = \frac{64}{16} + \frac{8}{16} + \frac{2}{16} + \frac{1}{16}$$
$$= 4 + \frac{1}{2} + \frac{1}{8} + \frac{1}{16}$$
$$= 1 \times 2^2 + 0 \times 2^1 + 0 \times 2^0 + 1 \times 2^{-1} + 0 \times 2^{-2} + 1 \times 2^{-3} + 1 \times 2^{-4}.$$

Donc $\frac{75}{16} = \overline{100{,}1011}^{2}$.

2 $$\frac{101}{8} = \frac{64}{8} + \frac{32}{8} + \frac{4}{8} + \frac{1}{8}$$
$$= 8 + 4 + \frac{1}{2} + \frac{1}{8}.$$

Donc $\frac{101}{8} = \overline{1100{,}101}^{2}$.

3 $$\frac{59}{4} = \frac{32}{4} + \frac{16}{4} + \frac{8}{4} + \frac{2}{4} + \frac{1}{4}$$
$$= 8 + 4 + 2 + \frac{1}{2} + \frac{1}{4}$$

Donc $\frac{59}{4} = \overline{1110{,}11}^{2}$.

4 $$\frac{61}{2} = 30 + \frac{1}{2} = 16 + 8 + 4 + 2 + \frac{1}{2}.$$

Donc $\frac{61}{2} = \overline{11110{,}1}^{2}$.

CORRIGÉS

8 À partir du complément à 2

Énoncé p. 49

Lycée Camille Jullian, Bordeaux

MÉTHODE

Pour retrouver le nombre en base 10 d'un complément à 2, on exécute la méthode de la page 27 à l'envers :

- on enlève 1 au binaire ;
- on prend le complément à 1 du résultat ;
- on retrouve le nombre en base 10 (sans le signe).

1 $\overline{11001011}^{2}$.

- On enlève 1 : $\overline{11001010}^{2}$;
- on prend le complément à 1 : $\overline{00110101}^{2}$;
- on retrouve le nombre en base 10 : $32 + 16 + 4 + 1 = 53$.

Ainsi, $\overline{11001011}^{2} = \overline{-53}^{10}$.

2 $\overline{11010100}^{2}$.

- On enlève 1 : $\overline{11010011}^{2}$;
- on prend le complément à 1 : $\overline{00101100}^{2}$;
- on retrouve le nombre en base 10 : $4 + 8 + 32 = 44$.

Ainsi, $\overline{11010100}^{2} = \overline{-44}^{10}$.

3 $\overline{10000010}^{2}$.

- On enlève 1 : $\overline{10000001}^{2}$;
- on prend le complément à 1 : $\overline{01111110}^{2}$;
- on retrouve le nombre en base 10 : $2 + 4 + 8 + 16 + 32 + 64 = 126$.

Ainsi, $\overline{10000010}^{2} = \overline{-126}^{10}$.

4 $\overline{10101010}^{2}$.

- On enlève 1 : $\overline{10101001}^{2}$;
- on prend le complément à 1 : $\overline{01010110}^{2}$;
- on retrouve le nombre en base 10 : $2 + 4 + 16 + 64 = 86$.

Ainsi, $\overline{10101010}^{2} = \overline{-86}^{10}$.

9 La valeur des booléens

Énoncé p. 49

Lycée François Mauriac, Bordeaux

1. `rep = False`. En effet, $x = -3$ et donc $x^2 = 9$. Or, « `x**2` » représente x^2 et « `x**2 == -9` » teste si x^2 est égal à -9, ce qui n'est pas le cas ici. Donc la réponse est *False*.
2. `rep = True`. En effet, $x = 3$, $y = 4$ et $z = 5$ donc $x^2 + y^2 = 3^2 + 4^2$, soit $x^2 + y^2 = 25$ et $z^2 = 5^5 = 25$.

   Le test « `x**2 + y**2 == z**2` » est donc validé : la réponse est *True*.

   Remarque : on reconnaît ici la réciproque du théorème de Pythagore.
3. `rep = False`. En effet, $a = 3$ et $b = -7$ donc $a^3 = 27$ et $b^2 = 49$.

   On teste alors si les deux conditions `a**3>50` et `b**2<50` sont vraies, ce qui n'est pas le cas car l'une d'elles est fausse. La réponse renvoyée est donc *False*.
4. `rep = True`. En effet, nous retrouvons les mêmes valeurs de a et b que dans la question précédente et la première partie du test est la même : « `(a**3 > 50 and b**2 < 50)` » est donc faux. Mais la seconde partie du test : « `(a**2 < 10 and b**2 >10)` » est vraie car $a^2 = 9$ (qui est bien inférieur à 10) et $b^2 = 49$ (qui est bien supérieur à 10).

   La présence de `or` signifie que l'on teste si l'une au moins des deux propositions est vraie, ce qui est le cas, donc la réponse est *True*.

10 Booléens et chaînes de caractères

Énoncé p. 49

Lycée polyvalent Émile Zola, Aix-en-Provence

1. L'instruction « `"il" in (phrase1 and phrase2)` » teste si la chaîne de caractères « il » est présente dans les deux phrases, ce qui est le cas.

   Donc le programme affiche : « True ».
2. L'instruction « `"est" in (phrase1 and phrase2) and "na" in (phrase1 or phrase2)` » teste si la chaine de caractères « est » est présente dans les deux phrases (ce qui est le cas) et si la chaîne de caractères « na » est présente dans l'une des deux phrases (ce qui n'est pas le cas). Nous avons donc comme réponses : *True* ET *False*.

   Le programme affiche donc : « False ».

11 Opérations sur des booléens

Énoncé p. 50

Lycée Camille Claudel, Blois

Le tableau complété est donné page suivante.

COURS INTERROS CORRIGÉS

CORRIGÉS

a	b	not(a) and b	not(a) or b	not(a) and not(b)	not(a) or not(b)
0	0	0	1	1	1
0	1	1	1	0	1
1	0	0	0	0	1
1	1	0	1	0	0

12 Encore des opérations

Énoncé p. 50

Lycée Turgot, Paris

1

a	b	(a or b) and (a and b)
0	0	0
0	1	0
1	0	0
1	1	1

On s'aperçoit alors que la règle « (a or b) and (a and b) » est identique à l'opérateur « and ».

2

a	b	(a or b) or (a and b)
0	0	0
0	1	1
1	0	1
1	1	1

On s'aperçoit alors que la règle « (a or b) or (a and b) » est identique à l'opérateur « or ».

13 La loi « Delta »

Énoncé p. 50

Lycée Sainte-Anne, Brest

1 La table de la loi Delta est la suivante :

Delta	0	1
0	0	1
0	1	0

Explications :

- 0 Delta 0 = (0 or 0) and not(0 and 0)
  = 0 and not(0) = 0 and 1 = 0.
- 0 Delta 1 = (0 or 1) and not(0 and 1)
  = 1 and not(0) = 1 and 1 = 1.
- 1 Delta 0 = (1 or 0) and not(1 and 0)
  = 1 and not(0) = 1 and 1 = 1.
- 1 Delta 1 = (1 or 1) and not(1 and 1)
  = 1 and not(1) = 1 and 0 = 0.

2 On constate que la table de la loi Delta est celle de l'opérateur xor. Donc derrière la loi Delta se cache en définitive l'opérateur xor.

14 La fonction « Delta »

Énoncé p. 51

Lycée Sophie Germain, Paris

1 Dans la fonction « Delta », le test « `x in (A or B)` » renvoie *True* car le caractère « K » apparaît dans l'une des chaînes `A` ou `B`.

Le test « `x in (A and B)` » renvoie *True* car « K » apparaît dans les deux chaînes, donc `not(x in (A and B))` vaut *False*.

Ainsi, le test « `(x in (A or B)) and not(x in (A and B))` » est équivalent au test « True and False » et renvoie donc la valeur *False*.

Le programme affiche donc : « False ».

2 Nous l'avons vu à la question précédente, « K » apparaît dans les deux chaînes ; ceci nous pousse à prendre un caractère qui n'apparaît que dans l'une d'elle. Prenons par exemple : « A ». Alors :

- « `x in (A or B)` » renvoie *True* ;
- « `x in (A and B)` » renvoie *False* donc « `not(x in (A and B))` » renvoie *True* ;
- « `(x in (A and B)) and not(x in (A and B))` » est équivalent au test « True and True » et renvoie *True*.

3 Dans cette nouvelle définition, on a enlevé les parenthèses aux deux « `(A or/and B)` » ; il n'y a donc plus de priorité aux opérateurs logiques `or` et `and`.

Pour `x = 'A'`, le test « `x in A or B` » n'a pas de sens car on fait avant tout le test « `x in A` » (qui renvoie *True*) et on effectue le test « True or B », qui n'est pas un test sur booléens. Python renvoie à cela : « `'OKLM'` », et ce n'est pas ce que nous attendions (on n'attendait pas grand-chose d'ailleurs...).

Ainsi, enlever les parenthèses est une erreur.

15 Remplacer une lettre

Énoncé p. 51

Lycée Camille Jullian, Bordeaux

```python
A = "Levez-vous vite, orages désirés qui devez emporter
     René dans les espaces d'une autre vie !"
B = ""

for i in range(len(A)):
    if A[i] == 'e':
        B += 'a'
    else:
        B += A[i]

print(B)
```

COURS
INTERROS
CORRIGÉS

Explications :

- Nous devons parcourir la chaîne `A` ; la variable `i` représente ici l'index des caractères de la chaîne. Il faut donc que `i` prennent autant de valeurs qu'il y a de caractères dans `A`, donc `len(A)` valeurs.
- On souhaite construire la chaîne `B` comme étant identique à la chaîne `A` sauf quand le caractère est « e », auquel cas on le remplace par « a ». Il faut donc faire le test « si le caractère d'index `i` est 'e' », ce qui se traduit par « `if A[i] == 'e'` ».

16 Nombre de « a »

Énoncé p. 52

Lycée François Mauriac, Bordeaux

Pour résoudre cet exercice, nous définissons la fonction « `nombre_de_a` » comme suit :

```
def nombre_de_a(chaine):
    compteur = 0
    for caractere in chaine:
        if caractere == 'a' or caractere == 'A':
            compteur += 1
    return compteur
```

Cette fonction parcourt chaque caractère de la chaîne `chaine` et incrémente un compteur si le caractère est un « a » ou un « A ». Voici les résultats pour les cas donnés dans l'énoncé :

1. `nombre_de_a("bonjour Ernest")` : la chaîne ne contient aucun « a » ou « A ». La fonction renvoie 0.
2. `nombre_de_a("Salut Bernard")` : la chaîne contient deux caractères « a » (un dans « Salut » et un dans « Bernard »). La fonction renvoie 2.
3. `nombre_de_a("Au revoir Anatole")` : la chaîne contient trois caractères « a » ou « A » (un dans « Au », un dans « Anatole » et un dans « revoir »). La fonction renvoie 3.

Ainsi, cette fonction est valide pour tous les cas testés.

17 Modification

Énoncé p. 52

Lycée François Villon, Les Mureaux

`len(A)` correspond au nombre de caractères dans la variable `A`, donc représente ici le nombre 19.

L'instruction « `for i in range(len(A))` » signifie que la variable `i` prend 19 valeurs en partant de 0.

De plus, le test « `if i%2 == 0` » signifie : « si `i` est pair » car `i%2` représente le reste de la division euclidienne de `i` par 2, et si ce reste est égal à 0, alors cela signifie que `i` est pair.

Le tableau d'exécution du programme commence ainsi :

Valeurs de `i`	0	1	2	3	4	...
Valeurs de `B`	H	H*	H*a	H*a*	H*a*t	...

Le programme remplace donc toutes les lettres dont l'index est impair par une astérisque.

Il affiche donc : `H*a*t*n*i*o*o*m*n*s`

18 Manipulation

Énoncé p. 53

Lycée Camille Claudel, Blois

En premier lieu, on remarque que l'incrémentation de la fonction `range` n'est pas conventionnelle : n'oublions pas que `for i in range(10)` (par exemple) signifie que `i` varie de 0 à 9 (prenant ainsi 10 valeurs), mais `for i in range(a,b,k)` signifie que `i` prend des valeurs allant de `a` à `b` avec un pas égal à `k` (c'est-à-dire de k en k).

Ici, `len(A)-1` est égal à $7 - 1 = 6$ (donc les valeurs de `i` vont partir de 6 et vont aller jusqu'à $-1 + 1$ c'est-à-dire 0, avec un pas de -1); `i` va donc prendre les valeurs : 6, 5, 4, 3, 2, 1 et 0. D'où :

Valeurs de `i`	6	5	4	3	2	1	0
Valeurs de `A[i]`	e	g	o	l	r	o	H
Valeurs de `B`	e	eg	ego	egol	egolr	egolro	egolroH

Le programme affiche donc : « egolroH », c'est-à-dire le mot « Horloge » à l'envers.

19 Les initiales

Énoncé p. 53

Lycée André Theuriet, Civray

Le programme est donné page suivante.

Explications :

- On commence par demander de saisir les noms et prénoms à l'aide de la commande `input` ;
- On créé ensuite une boucle qui permet d'itérer sur la chaîne de caractères saisie ;
- Pour chaque itération (chaque caractère de la chaîne), on teste si ce n'est pas le caractère « espace », auquel cas on ajoute le caractère à la chaîne `nom` (si le booléen `separator` vaut « `False` », sinon on ajouter le caractère à la variable `prenom`).

CORRIGÉS

```
nom_prenom = input("Entrez vos noms et prénoms séparés
    par un espace : ")
prenom, nom, separator = "", "", False

for i in nom_prenom: # on parcourt l'entrée
  if separator == False: # si le booléen vaut encore
    False
    if i != " ": # si le caractère n'est pas l'espace
      nom += i # on concatène le caractère à "nom"
    else: # sinon : si on est sur l'espace
      separator = True # on définit le booléen à True
  else: # si le booléen vaut True
    prenom += i # on concatène i à "prenom"

print("Votre prénom est :",prenom+".")
print("Votre nom est :",nom+".")
print("Vos initiales sont :",nom[0].upper()+prenom[0].
    upper()+".")
```

20 Que fait ce programme ?

Énoncé p. 54

Lycée Henri Alain-Fournier, Bourges

1 Les trois premières lignes sont des affectations : la variable `debut` prend la valeur d'une phrase (c'est donc une chaîne de caractères), la variable `lettre` prend la valeur « r » (c'est donc aussi une chaîne de caractères), et la variable `fin` est vide, mais est déclarée comme une chaîne de caractères puisque des guillemets sont présents dans son affectation.

Les variables `debut`, `lettre` et `fin` sont donc des chaînes de caractères.

2 Analysons la boucle du programme :

- « `for i in debut:` » signifie que l'on parcourt la chaîne de caractères désignée par la variable `debut` ;
- « `if i != lettre: fin += i` » signifie que si la lettre sur laquelle on est est différente de celle stockée dans la variable `lettre` (donc ici, « r ») alors on concatène ce caractère à ce qui était stocké précédemment dans la variable `fin`.
  Cette boucle a donc pour mission de construire une chaîne de caractères `fin` sur la base de la chaîne de caractères stockée dans `debut` en évitant tous les caractères « r ».

Ainsi, le programme affiche la phrase stockée dans `debut` en enlevant les « r », ce qui donne :

```
C'est un tou de vedue où chante une iviè.
```

3 Si on souhaite choisir la phrase de début et le caractère à enlever, on utilise la fonction `input` :

```
debut = input("Entrez une phrase : ")
lettre = input("Lettre à enlever : ")
fin = ""
for i in debut:
    if i != lettre:
        fin += i
print(fin)
```

21 Donner le rang d'un caractère

Énoncé p. 54

Lycée Turgot, Paris

1 La valeur du booléen `presence` est par défaut égale à *False*. Elle change dans la boucle si la condition « `phrase[i] == lettre` » est vraie, c'est-à-dire si la lettre rencontrée est le caractère que l'on a entré au clavier.

Ainsi, `presence` vaut *True* si on rencontre au moins une fois dans la variable `phrase` le caractère entré au clavier. Si le caractère n'est pas dans `phrase` alors il n'y a aucun affichage dans la boucle. Dans ce cas, l'utilisateur ne voit donc rien à l'écran et peut se poser des questions quant à l'exécution du programme. Il faut donc afficher que le caractère n'a pas été trouvé dans ce cas précis ; c'est le but de la dernière partie du programme.

2 Le fait d'écrire dans la fonction `print` « `"Le caractère",lettre` » affiche à l'écran le texte suivi de la valeur de la variable `lettre`, les deux étant séparés par un espace : la virgule permet de mettre un espace automatiquement.

Le fait de ne pas utiliser la virgule avant le point final signifie que l'on ne veut pas insérer un espace entre l'index de la lettre et ce point. C'est pourquoi on utilise la concaténation (avec le « + »).

Or, la concaténation suppose que les deux variables concaténées sont du même type, d'où la présence de `str(i)` pour convertir l'index (qui est une variable numérique) en chaîne de caractères.

3 Dans ce programme, la variable `i` est du type *chaîne de caractères* ; la boucle parcourt la variable `phrase` et dès que le caractère sur lequel on est correspond à la lettre cherchée, on affiche la réponse. Mais ici, `phrase.index(i)` correspond à l'index de la première occurrence trouvée de la variable `i` dans la variable `phrase`, donc `phrase.index(i)` est une constante. Ce programme affiche donc toujours le même index.

Par exemple, si on entre comme phrase :

```
Bonjour. Je m'appelle Kévin.
```

et si on recherche le caractère « n », ce dernier programme affiche 2 fois :

```
Le caractère n est trouvé à l'index 2.
```

22 Dans la tirelire de Clément

Énoncé p. 55

Lycée Raoul Follereau, Nevers

```
somme = 100

for i in range(10):
    somme *= 0.9

print(somme)
```

Explications :

- la première ligne doit affecter à la variable `somme` la somme que possède Clément au début, donc 100 € ;
- la boucle doit s'exécuter 10 fois car nous souhaitons savoir la somme qu'il reste à Clément après 10 semaines ; on utilise donc « `range(10)` » ;
- le calcul de `somme` dans la boucle repose sur le fait que, quand on enlève 10 % à `somme`, il reste 90 % de `somme`.
  « `somme *= 0.9` » signifie que l'on multiplie par 0,9 le nombre stocké dans la variable `somme`. C'est bien ce que l'on veut.

23 Somme des premiers entiers

Énoncé p. 55

Lycée Sainte-Clotilde, Strasbourg

```
def somme(n):
    s = 0
    for i in range(n+1):
        s += i
    return s

s = somme(100)
print (s)
```

Explications :

- on part de $s = 0$ car la première somme possible est pour le premier entier naturel n possible, à savoir 0, et dans ce cas, la somme vaut 0 ;
- on doit parcourir l'ensemble $\{0; 1; 2; 3; \cdots; n\}$, c'est-à-dire $n + 1$ valeurs, d'où l'instruction « `range(n+1)` » ;
- dans la boucle, afin de calculer la somme, on doit ajouter à la valeur de `s` déjà stockée celle de `i`, d'où l'instruction « `s += i` » (que l'on peut aussi écrire : `s = s + i`).

24 Le programme mystère

Énoncé p. 56

Lycée Sainte-Anne, Brest

- L'instruction « `for i in range(len(phrase)):` » nous informe que l'on souhaite parcourir la chaîne de caractères `phrase`.
- L'instruction « `presence = False` » définit un booléen qui est faux par défaut lorsque l'on entame chaque étape de la boucle.
- L'instruction « `for j in (range(len(lettres))):` » nous informe que l'on parcourt la chaîne de caractères `lettres`.
- L'instruction `if phrase[i] == lettres[j]` est un test pour savoir si le caractère sur lequel on est pour la chaîne `phrase` est identique au caractère sur lequel on est pour la chaîne `lettres`. Si tel est le cas, le booléen `presence` vaut *True*.
- Une fois que l'on a parcouru la chaîne `lettres`, si le booléen `presence` vaut *False* (c'est-à-dire si aucun caractère de `lettres` n'a été trouvé dans `phrase`) alors on concatène à `lettres` le caractère de `phrase` sur lequel on est, précédé d'un point-virgule, si l'index ne vaut pas 0 ; si l'index vaut 0, alors on concatène à `lettres` directement le caractère de `phrase` sur lequel on est.

À la vue de ceci, ce programme a pour objectif de lister tous les caractères utilisés dans la variable `phrase` une et une seule fois, et d'afficher tous ces caractères séparés par des points-virgule.

25 Le jeu du « plus » ou « moins »

Énoncé p. 56

Lycée Henri Alain-Fournier, Bourges

```
import random
nombre = random.randint(1,100)
n, count = 0, 0
while n != nombre:
    count += 1
    n = int(input("Entrez un nombre : "))
    if n == nombre:
        if count == 1:
            terminaison = ""
        else:
            terminaison = "s"
        print("Bravo ! C'est exactement ce nombre !
    Vous avez trouvé en",count,"coup"+terminaison+".")
    else:
        if nombre > n:
            print("Plus !")
        else:
            print("Moins !")
```

COURS
INTERROS
CORRIGÉS

CORRIGÉS

Explications :

- on commence par choisir un nombre au hasard entre 1 et 100 avec l'instruction `nombre = random.randint(1,100)` ;
- on initialise les variables `n` (représentant le nombre entré au clavier) et `count` (représentant le nombre de propositions faites) à 0 ;
- tant que le nombre proposé est différent du nombre choisi aléatoirement (`while n != nombre`), on exécute les instructions qui suivent ;
- on incrémente le compteur `count` (on lui ajoute 1 à chaque fois) et on demande d'entrer un nombre ;
- si le nombre proposé est égal au nombre choisi aléatoirement alors on a gagné. Si le nombre de coup(s) est égal à 1, alors on définit la terminaison du mot « coup » par une chaîne vide, sinon cette terminaison est un « s ». On affiche alors la phrase finale en incluant le nombre de coup(s) ;
- si le nombre proposé n'est pas le nombre choisi aléatoirement alors il est soit plus petit (`nombre > n`) – auquel cas on affiche « Plus ! » – soit plus grand (`nombre < n`) et dans ce cas, on affiche « Moins ! »

26 Somme de nombres

Énoncé p. 57

Lycée François Mauriac, Bordeaux

Vous trouverez une proposition de programme page suivante.

Explications :

- Nous initialisons la variables `total` à 0 pour stocker la somme des nombres, `temp_number` à 0 pour stocker temporairement les nombres en cours de lecture, et `in_number` à False pour indiquer si nous sommes en train de lire un nombre.
- Nous utilisons une boucle `for` pour parcourir chaque caractère de la chaîne. Si le caractère est un chiffre, nous le convertissons en nombre [`int(caractère)`] et l'ajoutons à `temp_number`. Nous mettons également le booléen `in_number` à `True` pour indiquer que nous sommes en train de lire un nombre.
- « `else` » : si le caractère n'est pas un chiffre et que nous étions en train de lire un nombre, nous ajoutons `temp_number` à `total` et réinitialisons `temp_number` et `in_number`.
- « `if in_number` » : à la fin de la boucle, si nous étions en train de lire un nombre, nous ajoutons `temp_number` à `total`.

```
def somme(chaîne):
    # Initialiser la somme totale et la variable
    temporaire
    total = 0
    temp_number = 0
    # Variable booléenne qui nous dira si on est sur un
     nombre ou pas
    in_number = False

    # Parcourir chaque caractère de la chaîne
    for caractère in chaîne:
        if caractère.isdigit():
            # Si le caractère est un chiffre, ajouter à
     la variable temporaire
            temp_number = temp_number * 10 + int(caract
    ère)
            in_number = True
        else:
            # Si le caractère n'est pas un chiffre et
    que nous étions en train de lire un nombre
            if in_number:
                total += temp_number
                temp_number = 0
                in_number = False

    # Ajouter le dernier nombre temporaire à la somme
    totale si nécessaire
    if in_number:
        total += temp_number

    # Retourner la somme totale
    return total
```

27 Black Jack

Énoncé p. 57

Lycée Sophie Germain, Paris

Nous précisons que l'ensemble des fonctions proposées dans ce corrigé sera téléchargeable par le dernier QRcode.

1 *Fonction pour tirer une carte.*

Une fonction pour tirer une carte aléatoire pourrait ressembler à celle proposée page suivante.

CORRIGÉS

```
import random

def tirer_carte():
    valeur = random.randint(1, 13)
    if valeur == 1: # l'As vaut 11
        return 11
    else:
        return valeur
```

2 *Fonction pour initialiser le jeu.*

Une fonction pour tirer une carte pour le joueur humain et une pour l'ordinateur :

```
def initialiser_jeu():
    joueur = tirer_carte()
    ordinateur = tirer_carte()
    return joueur, ordinateur
```

3 *Fonction pour faire jouer l'utilisateur :*

Cette fonction permet au joueur humain de continuer à tirer des cartes jusqu'à ce qu'il le décide ou qu'il dépasse 21 :

```
def jouer_humain():
    points = 0
    while points <= 21:
        points += tirer_carte()
        print(f"Points actuels : {points}")
        if points > 21:
            print("Vous avez dépassé 21, vous avez
    perdu !")
            break
        continuer = input("Voulez-vous tirer une
    autre carte ? (o/n) : ")
        if continuer.lower() != 'o':
            break
    return points
```

4 *Fonction pour faire jouer l'ordinateur.*

Une stratégie rudimentaire pourrait consister à tirer des cartes tant que le total est inférieur à 17. Voir la proposition page suivante.

```
def jouer_ordinateur():
    points = 0
    while points < 17:  # L'ordinateur tire jusqu'à
     ce qu'il atteigne 17 points ou plus
        points += tirer_carte()
        if points > 21:
            print("L'ordinateur a dépassé 21, il a
    perdu !")
            break
    return points
```

5 *Programme principal.*

Voici comment organiser la partie et déterminer le vainqueur :

```
def partie():
    joueur, ordinateur = initialiser_jeu()
    print(f"Vous avez tiré une carte valant {joueur}
     points.")
    print(f"L'ordinateur a tiré une carte valant {
    ordinateur} points.")

    joueur = jouer_humain()
    if joueur > 21:
        print("L'ordinateur gagne !")
        return

    ordinateur = jouer_ordinateur()
    if ordinateur > 21 or joueur > ordinateur:
        print("Vous gagnez !")
    elif joueur == ordinateur:
        print("Égalité !")
    else:
        print("L'ordinateur gagne !")
```

Le QRcode ci-dessus vous donne accès à toutes les fonctions de ce corrigé.

Chapitre 3

Types construits

Plan du chapitre

1. Les listes
2. Les tuples (appelés aussi p-uplets)
3. Les dictionnaires
4. Traitement de données en table

1 Les listes

Exercice type 1 — Lycée Turgot, Paris

On considère la matrice :

$$A = \begin{pmatrix} -3 & 1 & 0 \\ 1 & -1 & 2 \\ 0 & -3 & 5 \end{pmatrix}.$$

1. Créer une liste nommée `a` représentant la matrice A.
2. Écrire une fonction `valabs(liste)` qui retourne une liste composée des items de la liste `liste` sans leur signe.
   Aide : on pourra utiliser la fonction `abs(x)` *qui donne la valeur absolue du nombre x (c'est-à-dire la valeur de x sans son signe).*
3. À l'aide de la fonction `valabs(liste)`, construire une liste `b` par compréhension représentant une matrice dont tous les coefficients sont égaux à ceux de A sans leur signe.
4. Créer une fonction `somme(matrice)` qui calcule la somme des coefficients d'une matrice.

Voir corrigé page 87

1.1 Définitions

Définitions 1

Une *liste* est une structure qui permet de regrouper plusieurs éléments, appelés aussi *items*.

Remarque : le mot *item* est masculin.

COURS

Exemples

- On souhaite que la variable `villes` contiennent plusieurs villes de France. En Python, on peut la définir par exemple ainsi :

```
villes = ["Bordeaux","Paris","Marseille","Lyon","Nantes"]
```

- On souhaite que la variable `ages` contiennent plusieurs valeurs numériques. On peut alors procéder ainsi :

```
ages = [10,11,12,13,14,15,16,17,18]
```

1.2 Opérations sur les items d'une liste

1.2.1 - Créer une liste à items identiques

Pour créer une liste ne comprenant que des items prenant la même valeur, on utilise la syntaxe :

```
liste = [<valeur>]*<nombre d'items souhaités>
```

Exemples

- L'instruction suivante créée une liste de 10 items prenant tous la valeur 0.

```
liste = [0]*10
```

- L'instruction suivante créée la liste `[1,2,1,2,1,2,1,2,1,2]`.

```
[1,2]*5
```

1.2.2 - Afficher un item

Comme pour les chaînes de caractères, la numérotation (ou l'*indexation*) des items commence à 0.

Exemple :

```
villes = ["Bordeaux","Paris","Marseille","Lyon","Nantes"]
print(villes[0])
```

Ce programme affiche le premier item, à savoir ici : « Bordeaux ».

1.2.3 - Modifier un item

Si on souhaite modifier la valeur d'un item, il suffit de le remplacer directement à l'aide de son index.

Exemple :

```
villes[4] = 'Bourg-en-Bresse'
```

Ici, le cinquième item de la liste `villes`, qui était « Nantes », a été remplacé par une autre valeur, à savoir ici : « Bourg-en-Bresse ».

1.2.4 - Supprimer un item

Il existe deux manières de supprimer un item en Python, comme le montrent les exemples suivants.

Exemples

- Avec la commande `del` (avec l'index de l'item) :

```
del villes[4] (efface l'item d'index 5)
```

- Avec la méthode `remove` (avec la valeur de l'item) :

```
villes.remove("Paris")
```

1.2.5 - Inverser les items

On peut inverser l'ordre des items dans une liste à l'aide de la méthode `reverse`.

Exemple :

```
villes = ["Bordeaux","Paris","Marseille","Lyon","Nantes"]
villes.reverse()
print(villes)
```

Le programme affiche alors :

```
['Nantes', 'Lyon', 'Marseille', 'Paris', 'Bordeaux']
```

1.2.6 - Compter le nombre d'items

On peut compter le nombre d'items dans une liste à l'aide de la fonction `len`, déjà rencontrée dans le chapitre précédent, page 41.

Exemple :

```
villes = ['Grenoble', 'Nantes', 'Lyon', 'Marseille',
'Paris']
print(len(villes))
```

Ce programme nous renvoie la valeur `5`, car il y a 5 items dans la liste.

1.2.7 - Compter les occurrences d'une valeur

Si on souhaite savoir combien de fois un item apparaît dans une liste, on peut utiliser la méthode `count`.

Exemple :

```
maliste = ['Jean', 'Pierre', 'Jean', 'Paul', 'Lucie']
maliste.count('Jean')
```

Ce programme affiche « 2 » car « Jean » apparaît deux fois dans la liste.

COURS

1.2.8 - Trouver l'index d'un item

Dans une liste volumineuse, il est souvent utile de faire appel à la méthode `index` pour trouver un item bien particulier.

Exemple :

```
valeurs = ['120','48','75','84','145','121','78','49']
valeurs.index('121')
```

Ce programme affiche l'index de l'item « 121 », donc il affiche ici : « 5 ».

Si on ne souhaite pas connaître l'index de l'item, mais que l'on souhaite savoir si l'index est bien présent dans la liste, on peut utiliser le mot clé `in`, dont nous avions parlé à la page 38.

Exemple :

```
valeurs = ['120','48','75','84','145','121','78','49']
if ('121' in valeurs):
     print('Ok !')
```

Ce programme retourne « `Ok !` » car la valeur a été trouvée dans la liste.

1.3 Manipulation des listes

1.3.1 - Vider une liste

On vide une liste ainsi :

```
villes = []
```

1.3.2 - Trier une liste

On trie dans l'ordre alphabétique une liste avec la méthode `sort` :

```
maliste = ['Jean', 'Pierre', 'Alexine', 'Paul', 'Lucie']
maliste.sort()
print(maliste)
```

Le programme affiche :

```
['Alexine', 'Jean', 'Lucie', 'Paul', 'Pierre']
```

1.3.3 - Afficher une partie d'une liste (non exigible)

On peut afficher un certain nombre d'items depuis le début d'une liste ainsi :

```
maliste = ['Jean', 'Pierre', 'Alexine', 'Paul', 'Lucie']
print(maliste[:2])
```

Le programme affiche :

```
['Jean', 'Pierre']
```

Mais on peut aussi afficher les valeurs des items d'une liste à partir de n'importe quel rang :

```
maliste = ['Jean', 'Pierre', 'Alexine', 'Paul', 'Lucie']
print(maliste[2:4])
```

Le programme affiche :

```
['Alexine', 'Paul']
```

ATTENTION

L'instruction `maliste[n:p]` affiche les items depuis l'index `n` jusqu'à l'item d'index `p-1`.

1.3.4 - Dupliquer une liste

Si on souhaite dupliquer une liste `villes` dans une autre liste `towns`, un simple « = » suffit. Mais dans ce cas, les deux listes seront dépendantes, c'est-à-dire que si on modifie l'une d'elles (en y ajoutant par exemple un item), l'autre changera de la même façon.

```
villes = ['Bordeaux','Paris']
towns = villes
towns.append('Lens')
print(villes)
```

Ce programme affiche :

```
['Bordeaux', 'Paris', 'Lens']
```

contre toute attente car nous avons modifié la liste `towns`, et non la liste `villes` affichée.

Pour créer deux listes indépendantes, on utilisera la syntaxe suivante :

```
villes = ['Bordeaux','Paris']
towns = villes[:]
towns[1]('Lens')
print(villes)
print(towns)
```

Ce programme affiche :

```
['Bordeaux', 'Paris']
['Bordeaux', 'Lens']
```

preuve que les deux listes sont indépendantes.

COURS

1.3.5 - Assembler une liste en une chaîne de caractères

La méthode `join` permet de transformer une liste en chaîne de caractères, en insérant (si besoin est) un caractère entre chaque item.

Exemple :

```
villes = ['Bordeaux','Paris','Lyon']
itineraire = " - ".join(villes)
print(itineraire)
```

Le programme affiche :

```
'Bordeaux - Paris - Lyon'
```

1.3.6 - Découper une chaîne de caractères en liste

Ceci est possible avec la méthode `split`.

Exemple :

```
villes = 'Bordeaux,Paris,Lens,Grenoble,Biarritz'
liste = villes.split(',')
print(liste)
```

Le programme affiche :

```
['Bordeaux', 'Paris', 'Lens', 'Grenoble', 'Biarritz']
```

1.3.7 - Concaténer deux listes

On peut concaténer deux listes avec les méthodes `append` ou `extend`.

Avec extend

Exemple :

```
villes = ['Bordeaux', 'Paris', 'Lens', 'Grenoble']
villes_espagne = ['Barcelone','Pampelune','Ibiza']
villes.extend(villes_espagne)
villes
```

Le programme affiche :

```
['Bordeaux', 'Paris', 'Lens', 'Grenoble', 'Barcelone',
    'Pampelune', 'Ibiza']
```

COURS
INTERROS
CORRIGÉS

Avec append

Exemple :

```
villes = ['Bordeaux', 'Paris', 'Lens', 'Grenoble',
'Biarritz']
villes_espagne = ['Barcelone','Pampelune','Ibiza']
villes.append(villes_espagne)
villes
```

Le programme affiche :

```
['Bordeaux', 'Paris', 'Lens', 'Grenoble',
  ['Barcelone', 'Pampelune', 'Ibiza']]
```

Différence

Alors que la méthode `append` ajoute exactement ce que l'on veut (dans le deuxième exemple, on ajoute une liste et c'est bel et bien une liste qui est ajoutée comme item dans la liste `villes`), la méthode `extend` *étend* la liste (comme son nom l'indique d'ailleurs).

Mais les deux méthodes n'admettent qu'un seul argument.

Ainsi, `liste.append('bidule','truc')` est impossible ; pour ajouter deux items, il faut écrire deux instructions :

```
liste.append('bidule')
liste.append('truc')
```

1.4 La liste des premiers entiers : la fonction range

Il est très souvent nécessaire d'avoir une liste de la forme :

```
[0,1,2,3,4,5,...,100]
```

Plutôt que de la rentrer au clavier (ce qui peut être très long si on va jusqu'à 1 million), le fonction `range` s'offre à nous.

Exemple :

```
list( range(10) )
```

constitue la liste :

```
[0,1,2,3,4,5,6,7,8,9]
```

Elle est constituée de 10 items, mais ne va pas jusqu'à 10.

COURS

Remarque : cette fonction est souvent utilisée dans les boucles. Par exemple, le programme :

```
s = 0
for n in range(10):
  s += n

print(s)
```

affiche « 45 » car $0+1+2+3+4+5+6+7+8+9=45$.

1.5 Écrire une matrice

Définition 2

Une *matrice* est un tableau contenant des nombres. Elle est souvent entourée de parenthèses ou de crochets.

Exemple : on considère le système linéaire suivant :

$$\begin{cases} 2x-5y+3=0 \\ 7x+4y-1=0 \end{cases}$$

On peut représenter ce système par un tableau :

Variables	x	y	constantes
Ligne 1	2	-5	3
Ligne 2	7	4	-1

La matrice représentant ce système peut alors être :

$$\begin{pmatrix} 2 & -5 & 3 \\ 7 & 4 & -1 \end{pmatrix}$$

Pour représenter une matrice en informatique, on utilise une liste dont tous les items sont des listes.

La matrice de l'exemple peut alors s'écrire :

```
matrice = [[2,-5,3],[7,4,-1]]
```

Chaque item est une liste contenant tous les coefficients d'une ligne.

1.6 Construire une liste par compréhension

Définition 3

Construire une liste par compréhension signifie simplifier le code pour le rendre plus lisible et plus rapide. On utilise alors la syntaxe suivante :

```
ma_liste = [<fonction sur item> for <item> in <liste> if
<condition>]
```

Voyons sur un exemple ce que cela peut donner.

Exemple : on considère la liste :

```
liste = [1,4,2,7,1,9,0,3,4,6,6,6,8,3]
```

On souhaite créer une autre liste à partir de celle-ci en ne gardant que les valeurs strictement supérieures à 5.
On peut bien entendu utiliser le code suivant :

```
new_list = [ ]
for i in liste:
  if i > 5:
    new_list.append(i)
```

Construire la liste par compréhension donne ceci :

```
new_list = [i for i in liste if i > 5]
```

Dans cet exemple,

- `<fonction item>` est représenté par `i`. Autrement dit, la fonction utilisée est celle qui prend pour valeur celle de l'item ;
- `<condition>` est ici : `i > 5`. On ne prend que les valeurs qui satisfont cette condition.

Solution de l'exercice type 1

Lycée Turgot, Paris

1. `a = [[-3,1,0],[1,-1,2],[0,-3,5]]`
2. La fonction `valabs(liste)` est :

```
def valabs(liste):
    for i in range(len(liste)):
        liste[i] = abs(liste[i])
    return liste
```

...

COURS INTERROS CORRIGÉS

Solution de l'exercice type 1 (suite)

Lycée Turgot, Paris

Explications :

- l'instruction `for i in range(len(liste))` fait en sorte que l'on prenne les entiers de 0 à `len(liste)-1` ;
- dans cette boucle, l'instruction `liste[i]=abs(liste[i])` remplace l'item de rang `i` par sa valeur absolue, c'est-à-dire par la même valeur sans le signe « — » quand il y en a un ;
- une fois la boucle terminée, une fois la liste parcourue, on retourne la valeur de la nouvelle liste ainsi obtenue.

3 `b = [valabs(i) for i in a]`
Cette instruction permet de construire une liste `b` contenant toutes les valeurs de la liste `a` sans leur signe.

4 La fonction suivante calcule la somme des coefficients d'une matrice :

```
def somme(matrice):
    s = 0
    for ligne in matrice:
        for c in ligne:
            s += c
    return s
```

Voir énoncé page 79

2 Les tuples (appelés aussi p-uplets)

Exercice type 2

Lycée Camille Claudel, Blois

Compléter le code suivant afin qu'il affiche les différentes valeurs que prend la variable `d`, différence (positive) entre les deux arguments du tuple `t`, tant que cette différence est inférieure ou égale à 40 000.

```
def f(x,y):
    a, b = x+y, x-y
    return (a,b)
t = (1,2)
d = ...
while d ... :
    t = f(...)
    d = ...
    print(d)
```

Voir corrigé page 90

2.1 Définition et affectation

Définition 4

Un *tuple* est une structure non modifiable (à la différence d'une liste) qui permet de regrouper plusieurs éléments.

On déclare un tuple comme une liste, si ce n'est que les crochets sont remplacés par des parenthèses.

Exemples

- On peut imaginer que le code suivant définit un point A de coordonnées $(1\ ;\ -2)$ dans un repère cartésien.

  ```
  point1 = ('A',1,-2)
  ```

- On peut aussi définir un tuple vide :

  ```
  t = tuple()
  ```

- On peut aussi définir un tuple à partir d'une chaîne de caractères :

  ```
  t1, t2 = tuple('python'), tuple('maths')
  ```

  Dans ce cas, :

  `t1 = ('p','y','t','h','o','n')` et `t2 = ('m','a','t','h','s')`.

2.2 Pourquoi utiliser un tuple ?

Il est très souvent question de tuples comme résultat d'une fonction.

Exemple :

```
q,r = divmod(7,3)
```

La fonction `divmod` renvoie un tuple (plus précisément, un couple) dont le premier argument est le quotient de la division euclidienne de 7 par 3, et le second, son reste.

2.3 Disperser un tuple

Disperser un tuple, c'est le « casser » en plusieurs variables (autant que d'arguments du tuple).

Exemple :

```
t = divmod(56,15)
print(*t)
```

Ce code nous renvoie les deux nombres du tuple (à savoir ici 3 et 11).

COURS

Solution de l'exercice type 2 — Lycée Camille Claudel, Blois

```python
def f(x,y):
    a, b = x+y, x-y
    return (a,b)

t = (1,2)
d = abs(t[0]-t[1])

while d<=40000:
    t = f(*t)
    d = abs(t[0]-t[1])
    print(d)
```

Pour calculer la différence (positive) entre deux nombres, on utilise la fonction native `abs` (valeur absolue) car on ne sait pas à l'avance si le premier argument du tuple est plus grand ou plus petit que le second.
De plus, la fonction `f(x,y)` admet deux arguments, et non un tuple, donc il est nécessaire de disperser le tuple dans la boucle `while` (`t = f(*t)`).

Voir énoncé page 88

3 Les dictionnaires

3.1 Définition

Définitions 5

Un *dictionnaire* est une sorte de liste si ce n'est qu'à la place des index (des entiers), on utilise des *clés* (c'est-à-dire des valeurs autres que des entiers).

Pour définir un dictionnaire, on procède de la même manière que pour les listes, en remplaçant les crochets par des accolades « { » et « } ».

Exemple :

```python
infos = {'nom':'Dupont' , 'prenom':'Gilbert' , 'age':'54'}
```

On peut aussi utiliser des tuples comme clés.

Exemple :

```python
p = { }
p[(-1,1)] = 'A'
p[(2,5)] = 'B'
print(p)
```

Ce code affiche :

```
{(-1, 1): 'A', (2, 5): 'B'}
```

3.2 Opérations sur les dictionnaires

3.2.1 - Récupérer une valeur

La méthode `get` permet de récupérer une valeur connaissant sa clé.

Exemple :

```
infos = {
      'nom':'Dupont' ,
      'prenom':'Gilbert' ,
      'age':'54'
      }
infos.get('prenom')
```

Ce programme affiche : « `Gilbert` ».

Si la clé n'est pas trouvée, rien ne s'affiche. Si vous le souhaitez, vous pouvez afficher un message par défaut quand la clé n'est pas trouvée.

Exemple :

```
infos = {
      'nom':'Dupont' ,
      'prenom':'Gilbert' ,
      'age':'54'
      }
infos.get('adresse','clé introuvable')
```

On peut aussi accéder à la valeur stockée dans le dictionnaire en plaçant la clé entre crochets :

```
infos['prenom']
```

Cette syntaxe est plus concise mais, dans le cas où la clé n'est pas trouvée, cela conduit à une erreur.

3.2.2 - Vérifier l'existence d'une clé

Le mot-clé `in` renvoie un booléen dont la valeur est « True » si la clé est présente.

Exemple :

```
infos = {'nom':'Dupont' , 'prenom':'Gilbert' ,
         'email':'gilbert@dupont.fr' }
if 'adresse' in infos:
   print(infos.get('adresse'))
else:
   print(infos.get('email'))
```

COURS

INTERROS

CORRIGÉS

COURS

3.2.3 - Récupérer les clés et valeurs par une boucle

Quand un dictionnaire est très long, on peut utiliser une boucle pour récupérer toutes les clés et valeurs. Ceci est possible à l'aide de la méthode `items`.

Exemple :

```
infos = {
    'nom' : 'Dupont' ,
    'prénom' : 'Gilbert' ,
    'email' : 'gilbert@dupont.fr' ,
    'age' : '54' ,
    'genre' : 'masculin' }

for cle,valeur in infos.items():
    print cle,valeur
```

3.3 Recherches avec conditions

Imaginons un tableau où chaque ligne représente un individu.
Dans chaque colonne, nous enregistrons des informations sur cet individu.
Il est très souvent utile d'afficher ou de sélectionner uniquement quelques lignes en fonction de ce que l'on recherche.

Exemple : le programme suivant créé un tableau réunissant quelques informations sur les clients d'un magasin.

```
clients = [
    {'nom':'Gilbert','prenom':'Jean','age':'54','genre':'m'
    },
    {'nom':'Dupont','prenom':'Pierre','age':'34','genre':'m
    '},
    {'nom':'Sylla','prenom':'Anne','age':'48','genre':'f'}
]
```

Nous avons créé un tableau où chaque ligne est un dictionnaire (pour plus de lisibilité et de compréhension), soit une liste de dictionnaires.

- Nous souhaitons dans un premier temps afficher tous les clients dont l'âge est inférieur à 50 ans. On peut alors utiliser le programme suivant :

```
for i in range(len(clients)):
    if int(clients[i]['age'])<50:
        print(clients[i])
```

On obtient :

```
{'nom':'Dupont','prenom':'Pierre','age':'34','genre':'m'}
{'nom':'Sylla','prenom':'Anne','age':'48','genre':'f'}
```

- Nous souhaitons à présent afficher les clients de genre « m » (pour « masculin ») :

```
for i in range(len(clients)):
    if clients[i]['genre'] == 'm':
        print( clients[i] )
```

On obtient :

```
{'nom':'Gilbert','prenom':'Jean','age':'54','genre':'m'}
{'nom':'Dupont','prenom':'Pierre','age':'34','genre':'m'}
```

La première ligne « `for i in range(len(clients)):` » parcourt toutes les lignes du tableau `clients` (ici, `len(clients))` est égal à 3 car il y a 3 lignes, 3 clients).

Ensuite, on insère notre condition et l'instruction à exécuter en cas de succès. À noter que pour le premier algorithme, il a fallu convertir `clients[i]['age']` en entier afin de pouvoir tester sa valeur.

3.3.1 - Copier un dictionnaire

Comme nous l'avons vu précédemment pour les listes, écrire « `dic2 = dic1` » ne fait que créer une copie dépendante de l'original. Il est préférable de passer par la méthode `copy`.

Exemple :

```
infos = {'nom':'Dupont' , 'prenom':'Gilbert' ,
'email':'gilbert@dupont.fr' }
backup = infos.copy()
```

Ainsi, on peut modifier le dictionnaire `infos` tout en ne changeant pas le dictionnaire `backup`.

3.3.2 - Supprimer une clé et sa valeur

On supprime une clé et sa valeur avec la fonction `del`.

Exemple :

```
infos = {'nom':'Dupont' , 'prenom':'Gilbert' ,
'email':'gilbert@dupont.fr' }
del infos['email']
print(infos)
```

Le programme affiche :

```
{'nom': 'dupont', 'prenom': 'Gilbert'}
```

COURS

4 Traitement de données en table

Exercice type 3 — Lycée Raoul Follereau, Nevers

Sur un site Internet, Eugénie a téléchargé une feuille de calcul dans laquelle sont respectivement enregistrés par colonne les noms, prénoms, adresses, codes postaux, villes, départements, numéro de téléphone et courriel des membres de son association. Elle sauvegarde ces données dans le fichier CSV nommé *feuille.csv*. Écrire un programme Python qui lui permet :

1. d'importer les données de son fichier CSV dans une table nommée `data` ;
2. de rechercher les doublons et de ne garder que les entrées uniques dans une table nommée `membres` ;
3. de trier les données dans l'ordre alphabétique suivant les noms et d'afficher toutes les valeurs de la table ;
4. de trier les données dans l'ordre alphabétique suivant les départements et d'afficher toutes les valeurs de la table ;
5. et n'afficher que les données des membres habitant la ville de Toulouse.

Voir corrigé page 101

4.1 Création d'une table

4.1.1 - Définition

Définition 6

On peut définir une table comme étant une liste où les items sont des listes, des tuples ou des dictionnaires.

Exemple :

```
tableau = [ [1,2,3] , [4,5,6] ]
```

Nous avons ici construit le tableau suivant :

1	2	3
4	5	6

Remarque : nous avons déjà rencontré des tables dans le chapitre précédent quand nous avons parlé de matrices.

4.1.2 - Table à nombre de lignes et colonnes variable

Pour construire un tableau dont les dimensions sont entrées au clavier, on pourra procéder ainsi :

```
lignes = int(input("Nombre de lignes : "))
colonnes = int(input("Nombre de colonnes : "))
tableau = [[0] * colonnes for i in range(lignes)]
```

Ici, on construit un tableau rempli de « 0 ».

4.1.3 - À partir d'un fichier CSV

Définition 7

Un fichier CSV (*Comma Separated Values*) est un fichier texte contenant plusieurs données séparées par une virgule.

Exemple : le fichier `contact.csv` suivant est un fichier CSV :

```
Prénom,Nom,courriel,Âge,Ville
Robert,Lepingre,bobby@exemple.com,41,Paris
Jeanne,Ducoux,jeanne@exemple.com,32,Marseille
Pierre,Lenfant,pierre@exemple.com,23,Rennes
```

Pour importer les données d'un fichier CSV et les mettre dans une table, on utilise le module `csv` de Python.

```
import csv
with open('contacts.csv', newline=") as csvfile:
     data = list(csv.reader(csvfile))
```

On construit ici une table `data` où chaque item est une liste contenant les informations du fichier CSV séparées par une virgule.

4.2 Recherche dans une table

Pour cette section, nous allons nous appuyer sur l'exemple suivant :

Exemple : on souhaite construire la table d'addition des nombres de 1 à 9. Pour cela, on utilise le programme suivant :

```
add = [[0] * 9 for i in range(9)]
for c in range(9):
   for l in range(9):
      add[c][l]=l+c+2
```

COURS

- La 1re ligne construit une table remplie de « 0 » , de dimension 9 × 9.
- La ligne « `for c in range(9):` » créé une boucle pour `c` allant de 0 à 9 (`c` représentant ici le nombre de colonnes souhaitées).
- La ligne « `for l in range(9):` » créé une boucle pour `l` allant de 0 à 9 (`l` représentant ici le nombre de lignes souhaitées).
- « `add[c][l]` » représente la somme de `c` et `l`, mais comme il y a un décalage au niveau des variables de boucles, on doit leur ajouter 1 à chacune, d'où le « `l+c+2` »).

4.2.1 - Rechercher la présence d'une valeur dans une table

Nous souhaitons savoir si le nombre « 10 » apparaît dans le tableau `add` créé dans l'exemple précédent.
Pour cela, nous allons le parcourir à l'aide d'une boucle.

```
test = False
for valeur in add:
   if 10 in valeur:
      test = True
      break

print(test)
```

- On commence par déclarer un booléen `test` qui, par défaut, vaut *False*. Il nous servira à signifier la présence du « 10 » dans le tableau.
- « `for valeur in add:` » signifie que l'on parcourt le tableau `add` et que chaque ligne est représentée par la variable `valeur`.
- « `if 10 in valeur` » signifie que l'on teste si « 10 » figure dans la variable `valeur`; si tel est le cas, on change la valeur du booléen `test` en *True* et on arrête la boucle (avec `break`).
- On termine en affichant le booléen `test`.

Remarque : la présence de `break` n'est pas réellement utile pour un tableau aussi petit que le nôtre, mais pour un tableau plus volumineux, il est fortement conseillé de l'utiliser afin de ne pas continuer la boucle pour rien.

4.2.2 - Afficher toutes les instances d'une valeur

Maintenant, nous souhaitons afficher toutes les valeurs « 10 » qui sont contenues dans notre tableau `add`. Pour cela, nous allons légèrement modifier le programme précédent.

```
for valeur in add:
   if 10 in valeur:
      print('10 est dans add[' + str(add.index(valeur)) + '
    ]['
            + str(valeur.index(10)) + ']')
```

Le programme renvoie alors :

```
10 est dans add[0][8]
10 est dans add[1][7]
10 est dans add[2][6]
10 est dans add[3][5]
10 est dans add[4][4]
10 est dans add[5][3]
10 est dans add[6][2]
10 est dans add[7][1]
10 est dans add[8][0]
```

Expliquons les instructions de ce programme :

- « `for valeur in add:` » est la boucle qui nous permet de parcourir le tableau `add`, et chaque item sera alors désigné par `valeur`. Mais attention : comme il s'agit d'un tableau, cet item est une liste.
- « `if 10 in valeur` » a pour but de vérifier si « 10 » est dans la liste `valeur`. Si tel est le cas, on exécute l'instruction suivante.
- « `add.index(valeur)` » désigne l'index (ici, le numéro de ligne du tableau en quelque sorte) de l'item (la liste « ligne ») où a été trouvé « 10 ».
  Donc en tapant « `str(add.index(valeur))` », on convertit ce numéro en chaîne de caractères pour l'afficher à l'aide de `print`.
  Quant à « `valeur.index(10)` », cela représente l'index du nombre « 10 » dans la liste `valeur`. On a ensuite converti en chaîne de caractères, comme précédemment.

Si l'aspect de la sortie de ce programme ne convient pas, si l'on souhaite afficher directement la somme qui donne « 10 », on pourra utiliser le programme suivant :

```
for valeur in add:
 if 10 in valeur:
  print('10 = ' + str(add.index(valeur)+1) + ' + ' +
  str(valeur.index(10)+1))
```

COURS
INTERROS
CORRIGÉS

COURS

La sortie est alors :

```
10 = 1 + 9
10 = 2 + 8
10 = 3 + 7
10 = 4 + 6
10 = 5 + 5
10 = 6 + 4
10 = 7 + 3
10 = 8 + 2
10 = 9 + 1
```

4.2.3 - Recherche de doublons dans une table

Reprenons l'exemple précédent, où la table `infos` admet pour items des dictionnaires.

Pour chasser les doublons, c'est-à-dire les dictionnaires identiques, on peut procéder ainsi :

```
data = [ ]

for i in infos:
    p = False
    for j in data:
        if j == i:
            p = True
    if p == False:
        data.append(i)
```

Explications :

- on crée d'abord une table vide `data` dans laquelle nous avons l'intention de mettre tous les dictionnaires différents ;
- on parcourt ensuite la table `infos` ;
- on initialise un booléen `p` (pour « présence ») à *False* pour signifier que l'item de `infos` sur lequel on est n'est, par défaut, pas présent dans la table `data` ;
- on parcourt ensuite `data` dans le but de voir si `i` y est. Si tel est le cas (`if j==i`, c'est-à-dire si l'item de `data` coïncide avec celui de `infos` sur lequel on est) alors le booléen passe à *True* ;
- une fois `data` parcourue, on regarde si le booléen `p` est toujours à *False*. Si tel est le cas, c'est que nous n'avons pas rencontré d'items de `data` qui soit identique à l'item de `infos` sur lequel on se trouve. Dans ce cas, on ajoute cet item à `data`.

On ajoute ainsi chaque item de `infos` tant que celui-ci ne coïncide pas avec un item déjà présent dans `data`.

Au final, `data` sera constitué d'items tous différents.

4.3 Tri sur une colonne

4.3.1 - Tri suivant la première colonne d'une table

Nous allons partir de la table `data` suivante :

```
data = [
        ("Clément", 14, 16),
        ("Charles", 12, 15),
        ("Ariane", 14, 18),
        ("Thomas", 11, 12),
        ("Damien", 12, 15),
]
```

Pour trier les items suivant la première colonne, on utilisera la fonction `sorted` si on ne souhaite pas modifier la table. Le code :

```
print(sorted(data))
```

renvoie :

```
[
('Ariane', 14, 18),
('Charles', 12, 15),
('Clément', 14, 16),
('Damien', 12, 15),
('Thomas', 11, 12)
]
```

sans affecter la table initiale.

Si on souhaite modifier l'ordre non plus à l'affichage mais dans la table elle-même, on utilisera la méthode `sort()` :

```
data.sort()
```

4.3.2 - Trier suivant une colonne quelconque

Les fonctions « lambda »

Imaginons que nous souhaitions créer simplement une fonction qui calcule le carré d'un nombre.

On peut bien sûr écrire :

```
def carre(x):
        return x**2
```

Mais on peut aussi faire appel au mot-clé `lambda` :

```
carre = lambda x : x**2
```

On dit ici que la fonction définie, la fonction `carre`, est une fonction *lambda*.

COURS

La syntaxe d'une fonction lambda est la suivante :

- on commence par lui donner un nom, et on fait suivre le signe « = » ;
- ensuite, on écrit le mot-clé « lambda » suivi de la variable. S'il y a plusieurs variables, on les sépare par une virgule ;
- ensuite, on écrit « : » suivi de l'instruction à exécuter.

Exemple :

```
somme = lambda x,y : x+y
```

La fonction `somme` calcule ici la somme des deux arguments. Ainsi, `somme(3,5)` renvoie le nombre « 8 ».

Le tri

`sorted` et `sort()` admettent un paramètre optionnel appelé `key`. C'est grâce à cet argument que l'on va pouvoir sélectionner la colonne sur laquelle on va effectuer le tri, et ce à l'aide d'une fonction lambda.

Reprenons l'exemple précédent de la table `data`. Pour trier suivant la colonne 2, on va écrire :

```
data.sort(key = lambda colonnes: colonnes[1])
```

ATTENTION

N'oubliez pas que la 1re colonne est désignée par `colonne[0]`, et donc la colonne 2 est `colonne[1]`.

Remarque : le mot « colonne » peut être remplacé par tout autre mot, comme par exemple : `data.sort(key=lambda col: col[1])`.

4.4 Fusion de deux tables

La fusion de deux tables a déjà été traitée précédemment : elle est possible à l'aide de la fonction `extend`.

Exemple :

```
import csv
with open('contacts.csv', newline='') as csvfile:
    data = list(csv.reader(csvfile))
with open('contacts2.csv', newline='') as csvfile:
    data2 = list(csv.reader(csvfile))
data.extend(data2)
print(data)
```

Ici, nous avons pris deux fichiers CSV, nous les avons converti en deux tables, puis nous les avons fusionné : la table `data` contient au final toutes les données.

Solution de l'exercice type 3

Lycée Raoul Follereau, Nevers

```
import csv # on importe les données CSV
with open('feuille.csv', newline='') as csvfile:
    data = list(csv.reader(csvfile))
membres = [ ] # Recherche de doublons
for i in data:
    p = False
    for j in membres:
        if j == i:
            p = True
    if p == False:
        membres.append(i)
print(membres.sort()) # Tri alphabétique sur la colonne
     1
print(membres.sort(key=lambda col: col[4])) # Tri col.
    5
for ligne in membres:
  if 'Toulouse' in ligne:
  print(ligne) # Afficher les membres de Toulouse
```

Voir énoncé page 94

INTERROS

1 QCM Listes, tuples et dictionnaires

5 min Corrigé p. 115

Pour chacune des questions suivantes, une des propositions est exacte. Laquelle ?

1 Si on définit la variable `lieux` par :

`lieux = ["Ville" , "Campagne" , "Forêt"],`

alors :

(a) « Ville » est :

- a un index ;
- b un item.

(b) la variable `lieux` contient :

- a une liste ;
- b un tuple.
- c un dictionnaire

2 Si on définit la variable `couleur` par :

`couleur = [ "bleu" , "vert" , "rouge" , "marron" ],`

alors :

- a l'index de `couleur` est 5 ;
- b l'index de « vert » est « 2 » ;
- c l'index de « marron » est « 3 ».

3 Si `liste = ['0','0','1','1','1','2','0','1','2','3']` alors `liste.count(1)` vaut :

- a 0
- b 3
- c 4

4 Si `lettres = ["a","b","c","c","d"]` et si on écrit ensuite « `lettres.append('c')` », que vaut `len(lettres)` ?

- a 3
- b 4
- c 5
- d 6

2 QCM Traitement de données

5 min

Corrigé p. 115

Pour chacune des questions suivantes, une seule des propositions faites est exacte. Trouvez-la.

1 Pour construire une table de 7 lignes et 7 valeurs remplie de 1, on utilise la syntaxe :

- a `[1]*7 for i in range(7)`
- b `for i in range(7) [1]*7`
- c `[[1]*7 for i in range(7)]`
- d `[for i in range(7) [1]*7]`

2 Après avoir appelé le module `csv` de Python, pour importer un fichier CSV dans une table nommée `table`, on utilise la syntaxe :

a
```
open('fichier.csv'):
        data=list(csv.reader('contact.csv'))
```
b
```
open('fichier.csv') as csvfile:
        data=list(csv.reader(csvfile))
```
c
```
with open('fichier.csv'):
        data=list(csv.reader('contact.csv'))
```
d
```
with open('fichier.csv') as fic:
        data=list(csv.reader(fic))
```

3 Disposant d'une table nommée `table` composée uniquement de nombres entiers, rechercher la présence de la valeur 7 peut s'écrire :

- a `if 7 in table: <instructions>`
- b `for i in table: if 7 in i: <instructions>`
- c `for i in table: if 7 in table: <instructions>`
- d `for i in table: for j in i: if 7 in j: <instructions>`

4 Une fonction lambda peut être définie par :

- a `f = lambda x : x*x+x+1`
- b `f : lambda x = x*x+x+1`
- c `f : lambda x : x*x+x+1`
- d `f = lambda x = x*x+x+1`

INTERROS

3 V/F Listes, tuples et dictionnaires

5 min Corrigé p. 116

Les affirmations suivantes sont-elles vraies ou fausses ?

1. Les données de la variable `individu = ('Mélusine','Enfaillite')` sont modifiables.
2. Si `phrase = "Un être humain est une machine sophistiquée."` alors `phrase.split(' ')` est une liste à sept items.
3. L'instruction « `mots.tri()` » permet de mettre les items de la liste `mots` dans l'ordre alphabétique.

Listes

4 Un air de déjà-vu

★ 5 min Corrigé p. 116

Lycée polyvalent Emile Zola, Aix-en-Provence

Quel rôle a le programme suivant ?

```
nom_prenom = input("Entrez vos noms et prénoms séparés
    par un espace : ")
nom = nom_prenom.split(' ')[0]
a = nom[0].upper()
prenom = nom_prenom.split(' ')[1]
b = prenom[0].upper()

print(a,b)
```

5 Stocker des valeurs

★ 10 min Corrigé p. 117

Lycée Camille Jullian, Bordeaux

On considère le nombre :

$$A = n^2 - 2n + 3, \qquad n \in \mathbb{N}.$$

Écrire un programme qui stocke dans une liste `valeurs` les différentes valeurs de A pour n allant de 0 à 10, et qui affiche la liste.

6 Manipulations simples

★ 10 min Corrigé p. 117

Lycée François Mauriac, Bordeaux

Écrire un programme Python qui permet de :

- définir une liste (nommée `liste`) contenant les nombres : 45, 17, 89, 38, 10 et 74 dans cet ordre ;
- trier et afficher la liste ainsi obtenue ;
- ajouter l'élément « 12 » à la liste et afficher la liste ;
- renverser et afficher la liste ;
- afficher l'indice de l'élément « 10 » ;
- enlever l'élément « 38 » et afficher la liste ;
- afficher la sous-liste du 2ᵉ au 3ᵉ élément ;
- afficher la sous-liste du début au 2ᵉ élément ;
- afficher la sous-liste du 3ᵉ élément à la fin de la liste ;
- afficher le dernier élément en utilisant un indiçage négatif.

7 Construction d'une liste spéciale

★ 10 min Corrigé p. 119

Lycée Sophie Germain, Paris

Soit n une valeur entière saisie par l'utilisateur.

1 Écrire un programme qui permet de construire une liste de la forme :

`[1,1,1,2,1,3,1,4,1,5,1,6,...,1,n]`

2 Écrire un programme qui permet de construire une liste de la forme :

`[1,1,2,1,2,3,1,2,3,4,1,2,3,4,5,...,1,2,3,...,n]`

8 La fonction range

★ 15 min Corrigé p. 119

Lycée Turgot, Paris

1 Écrire un programme qui :

- définit une liste (nommée `liste`) contenant tous les entiers de 1 à 10 à l'aide de la fonction `range` ;
- calcule et affiche la somme des éléments de `liste` et stocke le résultat dans la variable `somme` ;
- calcule et affiche la moyenne de ces nombres, en utilisant la variable `somme`.

2 Écrire un programme qui permet de calculer et d'afficher, à l'aide de la fonction `range` et d'une boucle, les sommes :

$$1+\frac{1}{2^2}+\frac{1}{3^2}+\cdots+\frac{1}{100^2},\quad 1+\frac{1}{2^2}+\frac{1}{3^2}+\cdots+\frac{1}{1\,000^2}$$

$$\text{et}\quad 1+\frac{1}{2^2}+\frac{1}{3^2}+\frac{1}{4^2}+\cdots+\frac{1}{10\,000^2}.$$

INTERROS

3 En utilisant la fonction range, écrire un programme qui calcule la somme :

$$50 \times 1 + 52 \times 2 + 54 \times 3 + 56 \times 4 + 58 \times 5 + \cdots + 100 \times 25.$$

9 Créer une liste à partir de deux

★★ 10 min Corrigé p. 120

Lycée François Villon, Les Mureaux

On donne les chaînes de caractères :

`chaine1 = "abc"` et `chaine2 = "de"`.

Écrire un programme permettant de créer la liste :

`['ad', 'ae', 'bd', 'be', 'cd', 'ce']`

à partir des chaînes `chaine1` et `chaine2`.

Indication : on pourra utiliser deux boucles imbriquées.

10 Compréhensions de liste

★★ 15 min Corrigé p. 121

Lycée Augustin Fresnel, Caen

1 Construire par compréhension une liste contenant les codes ASCII de la chaîne `"Hello World!"` en utilisant la fonction native `ord`.

2 Construire par compréhension une liste contenant les nombres de 0 à 90 (inclus), de dix en dix, soit : `[0, 10, 20, ..., 90]`.

3 Construire par compréhension une liste contenant 10 nombres entiers, pairs, tirés au sort, compris entre 0 et 20 (inclus). On utilisera la fonction `randint(a, b)` du module `random` qui retourne un nombre entier tiré aléatoirement dans l'intervalle $[a,b]$.

4 Construire par compréhension une matrice 10×10 (10 lignes, 10 colonnes) contenant tous les nombres entre 0 et 99. Le résultat doit être le suivant : `[[0, 1, 2, 3, 4, 5, 6, 7, 8, 9],`
`[10, 11, 12, 13, 14, 15, 16, 17, 18, 19],`
...
`[90, 91, 92, 93, 94, 95, 96, 97, 98, 99]]`

5 À partir d'une liste `mots` contenant des mots (`str`), construire une liste par compréhension contenant les chaînes de caractères de `mots` dont le nombre de lettres est pair. Exemple : à partir de `mots = ['goéland', 'mouette', 'sterne', 'cormoran']`
le résultat sera : `['sterne', 'cormoran']`.

6 À partir d'une liste `nb` contenant des nombres entiers, construire une liste par compréhension contenant uniquement les nombres de `nb` qui contiennent le chiffre 3. Exemple : à partir de `nb = [1, 301, 102, 13, 59, 33]`, la liste construite doit être : `[301, 13, 33]`.

Tuples

11 Pythagore à la rescousse

★ 5 min Corrigé p. 121

Lycée André Theuriet, Civray

On souhaite définir une fonction `is_square(t)` admettant un triplet `t` pour argument, ce triplet contenant la longueur des trois côtés d'un triangle.

1 Compléter le programme suivant afin qu'il vérifie si le triangle est rectangle ou non.

```
def is_square(t):
    if ... or ... or ...:
        print("Ce triangle est rectangle.")
    else:
        print("Ce triangle n'est pas rectangle.")
```

2 L'utilisation d'un triplet pour argument de cette fonction vous semble-t-elle judicieuse ? Dans la négative, proposer un autre type d'argument et les modifications à apporter au programme précédent.

12 Racines d'un trinôme

★ 10 min Corrigé p. 122

Lycée Henri Alain-Fournier, Bourges

On considère un trinôme $P(x) = ax^2 + bx + c$, où $a \neq 0$.

Compléter le programme suivant afin que la fonction `racines(coefs)` admette un triplet `coefs = (a, b, c)` pour argument, et retourne un tuple dont le premier argument est le nombre de racines de P et où les autres arguments sont les valeurs des éventuelles racines.

```
def racines(coef):
    a, b, c, delta = coef[0], coef[1], coef[2], ...
    if delta<0:
        result = ...
    else:
        if delta == 0:
            result = ...
        else:
            result = ...
    return result
```

INTERROS

13 Inventaire et tri

★★ 15 min Corrigé p. 122

Lycée Raoul Follereau, Nevers

Voici un début de programme :

```
epicerie = [ ("tomates",20) , ("pommes",10) , ("
    carottes",5) , ("poires",3) , ("bananes",4), ("
    ananas",1) ]
tri = int(input("Tri alphanumérique (1) ou tri alphabé
    tique (2) ? "))
```

Compléter ce programme afin qu'il affiche le tri souhaité, c'est-à-dire une liste où les noms de fruits sont triés dans l'ordre alphabétique pour le choix 2, ou une liste où les quantités sont triées dans l'ordre croissant pour le choix 1.

Indications : pour le tri numérique, on pourra construire une liste `epicerie_inverse` où les items sont des tuples de la forme (quantité , fruit).

14 Minimum, maximum et moyenne

★★ 15 min Corrigé p. 123

Lycée Sainte-Clotilde, Strasbourg

Écrire une fonction `min_max_moy` admettant une liste comme argument et renvoyant la valeur minimum, la valeur maximum et la moyenne des valeurs de la liste sous forme de tuple.

Dictionnaires

15 Le Scrabble

★ 5 min Corrigé p. 123

Lycée Sophie Germain, Paris

Le Scrabble est une jeu de société où l'on doit former des mots avec un tirage aléatoire de lettres, chaque lettre valant un certain nombre de points.

Le tableau ci-dessous donne les points de certaines lettres :

Lettres	A	B	C	D	E	F	G	H	I	J	K
Points	1	3	3	2	1	4	2	4	1	8	5

En considérant le programme donné page suivante, répondre aux questions :

1. Quel est le type de la variable `scrabble`.
2. Expliquez l'affectation de cette variable.
3. Qu'affiche ce programme quand on entre le mot « KAKI » au clavier ?
   On pourra s'aider d'un tableau d'exécution montrant chaque étape de la boucle.

4 Quel est le rôle de ce programme ?

```
scrabble = {'A':1, 'B':3, 'C':3, 'D':2, 'E':1, 'F':4,
'G':2, 'H':4, 'I':1, 'J':8, 'K':5}
somme = 0
mot = input("Entrez un mot : ")

for i in mot:
    somme += scrabble.get(i)

print(somme)
```

16 Tableau physico-chimique

★ 10 min Corrigé p. 124

Lycée Camille Jullian, Bordeaux

Le tableau suivant représente des informations physico-chimiques sur des éléments simples : température d'ébullition (T_e) et de fusion (T_f), numéro (Z) et masse (M) atomique.

	T_e	T_f	Z	A
Au	2 970	1 063	79	196,967
Ga	2 237	29,8	31	69,72

Affectez les données de ce tableau à un dictionnaire `dico` Python de façon à pouvoir écrire par exemple :

`print(dico["Au"]["Z"])` (affiche : 79)

17 Base de données

★★ 15 min Corrigé p. 124

Lycée Sainte-Clotilde, Strasbourg

L'entreprise *KiStokTout* possède une base de données dans laquelle elle enregistre les informations personnelles de ses clients :

nom, prénom, adresse, âge, taille (en cm), poids (en kg).

Elle initialise cette base de données par la ligne suivante :

```
registre = [1,{'nom':'HILUM' , 'prenom':'Mohan' ,
'adresse':'224 avenue des fleurs 75001 Paris', 'age':67,
'taille':178, 'poids':81}]
```

où « 1 » est le numéro de ligne dans la base de données.

1 **(a)** Quel est le type de la variable `registre` ?
(b) Quel est le type de `registre[0]` ?
(c) Quel est le type de `registre[1]` ?

2 Que retourne l'instruction `registre[1]['nom']` ?
Quel est le type de `registre[1]['nom']` ?

3 D'après le cours, quelle instruction devrait-on utiliser pour ajouter les informations de Kévin Dupont, 24 ans, habitant au 5 allée des cailloux 33000 Bordeaux, mesurant 1m87 pour 95 kg, à la ligne 2 de la base de données ?

4 Compléter le programme suivant (compléter les points de suspension) afin qu'il permette à une personne d'entrer au clavier les informations tant qu'il y en a.

```
registre = []
ligne = 0
rep = "..."

while rep == "O":
    nom = input("Entrez un nom : ")
    prenom = input("Entrez un prénom : ")
    adresse = input("Entrez une adresse : ")
    age = ...(input("Entrez un âge : "))
    taille = ...(input("Entrez une taille (en cm) :
     "))
    poids = ...(input("Entrez un poids (en kg) : "))

    ligne ...
    registre....([ligne,{’nom’:nom,’prenom’:prenom,’
    adresse’:adresse,’age’:age,’taille’:taille,’
    poids’:poids}])
    rep = input("Voulez-vous poursuivre la saisie (
    O = oui) ? ")
    rep....()
```

18 Création d'un dictionnaire

★★★ 15 min Corrigé p. 125

Lycée Turgot, Paris

Écrire une fonction `compte_lettres` qui admet pour argument une chaîne de caractères et qui construit un dictionnaire contenant le nombre d'occurrences de tous les caractères utilisés dans la chaîne.

Le programme ressemblera à ceci :

```
def compte_lettre(chaine):
  ...
  return dico
```

où `dico` est le dictionnaire attendu.

19 De l'hexadécimal au binaire ★★★ 20 min Corrigé p. 126

Lycée polyvalent Émile Zola, Aix-en-Provence

Écrire un programme Python permettant de convertir un nombre hexadécimal en binaire à l'aide d'un dictionnaire.

Indication : on pourra utiliser le programme de l'exercice 12 du chapitre 1.

Traitement de données en table

20 Théorie des graphes ★ 5 min Corrigé p. 127

Lycée Jean Macé, Niort

En théorie des graphes, une des branches des mathématiques, une matrice d'adjacence est une matrice ne contenant que des 0 et des 1. Nous la noterons (a_{ij}), où a_{ij} représente le coefficient de la i-ième ligne et j-ième colonne, avec i et j deux entiers compris entre 1 et n.

Supposons que cette matrice soit représentée par la table `adj[i][j]`. On dit que les sommets i et j sont connectés si `adj[i][j]` est égal à 1.

Écrire un code permettant d'afficher les sommets qui sont connectés.

21 Présence dans une base de données ★ 5 min Corrigé p. 127

Lycée Jean-Joseph Fourier, Auxerre

Considérons une table `data` dans laquelle chaque ligne renferme des données sur les membres abonnés d'un site internet : nom, prénom, courriel, pseudonyme, mot de passe et d'autres informations diverses.

Lors de son enregistrement, l'abonné(e) est obligé(e) d'informer son courriel et son mot de passe, mais pas nécessairement son pseudo.

Pour s'identifier sur le site, une personne a besoin d'entrer son mot de passe ainsi que son pseudonyme.

Une personne contacte l'administrateur du site et prétend qu'elle est abonnée et ne peut plus se connecter avec son pseudo « BigArnaudDesMers » ni même avec son courriel « bigarnaudofsea@gmail.com ».

1. Écrire un code permettant à l'administrateur de voir rapidement si l'une des deux informations communiquées par la personne est présente dans la table.
2. Modifier ce code afin qu'il prenne en compte la casse des informations.

22 Fichier INSEE

★ 10 min Corrigé p. 128

Lycée Henri Bergson, Angers

Retrouvez le corrigé de cet exercice en vidéo.

Sur le site Internet de l'INSEE, il est possible de télécharger des feuilles au format XLS (feuilles de calcul), qu'il est ensuite possible de convertir en fichier CSV.

Nous avons téléchargé une feuille nommée « insee2018.xls », et copié les 100 premières lignes qui m'intéressaient dans une nouvelle feuille afin de pouvoir la sauvegarder au format CSV. Nous avons au préalable changé le format des nombres afin de le rendre standard (le format initial des nombres, avec séparateur de milliers, n'est pas souhaitable).

1. Écrire les lignes de code qui permettent d'importer les données de ce fichier dans une table nommée `data`.
2. Par défaut, les données étaient classées par ordre alphabétique sur le nom des régions, puis sur le nom des communes.

   Écrire le code qui permet d'afficher toutes les communes, suivies de leur région et du nombre total d'habitants, dans l'ordre croissant de leur nombre total d'habitants (1 commune par ligne), tant que ce nombre total ne dépasse pas 1 500.
3. Écrire le code permettant d'afficher les communes dont le nombre total d'habitants est strictement compris entre 2 000 et 4 000.
4. En supposant que le fichier original n'ait pas été tronqué, écrire un code permettant d'afficher toutes les communes dont le nom commence par « Pont- », ainsi que leur région et le nombre total d'habitants.

Tous les types construits

23 Gestion d'une bibliothèque

★★★ 30 min Corrigé p. 129

Lycée François Mauriac, Bordeaux

Vous êtes responsable de la gestion d'une bibliothèque. Vous devez créer un programme Python pour gérer les livres et les emprunteurs.

- **Livres :** chaque livre est représenté par un dictionnaire contenant les informations suivantes.
  - → `titre (str)` : titre du livre
  - → `auteur (str)` : auteur du livre
  - → `annee (int)` : année de publication
  - → `disponible (bool)` : indique si le livre est disponible ou emprunté
- **Emprunteurs :** chaque emprunteur est représenté par un dictionnaire contenant les informations suivantes.

→ `nom (str)` : nom de l'emprunteur

→ `livres_empruntes (list)` : liste des titres des livres empruntés

Exemple de données :

```
livres = [
    {"titre": "1984", "auteur": "George Orwell", "annee
    ": 1949, "disponible": True},
    {"titre": "Le Petit Prince", "auteur": "Antoine de
    Saint-Exupéry", "annee": 1943, "disponible": True},
    {"titre": "Le Hobbit", "auteur": "J.R.R. Tolkien",
    "annee": 1937, "disponible": True}
]

emprunteurs = [
    {"nom": "Alice", "livres_empruntes": []},
    {"nom": "Bob", "livres_empruntes": []}
]
```

1 Écrire une fonction pour ajouter un livre à la bibliothèque. On la notera : `ajouter_livre(livres, titre, auteur, annee)`.

2 Écrire une fonction `ajouter_emprunteur(emprunteurs, nom)` pour ajouter un emprunteur.

3 Écrire une fonction pour emprunter un livre. On la notera : `emprunter_livre(livres, emprunteurs, nom_emprunteur, titre_livre)`.

4 Écrire une fonction pour rendre un livre. On la notera : `rendre_livre(livres, emprunteurs, nom_emprunteur, titre_livre)`.

5 Écrire une fonction `afficher_livres_disponibles(livres)` pour afficher tous les livres disponibles.

6 Écrire une fonction `afficher_emprunteurs(emprunteurs)` pour afficher tous les emprunteurs et les livres qu'ils ont empruntés.

24 Gestion d'un annuaire

★★ 15 min Corrigé p. 131

Lycée Lycée Camille Claudel, Blois

Vous devez créer un programme Python pour gérer un annuaire téléphonique. Chaque entrée de l'annuaire contient le nom, le prénom, et le numéro de téléphone d'une personne.

L'annuaire est représente par un dictionnaire nommé `annuaire`.

1. Écrire une fonction pour ajouter une nouvelle entrée à l'annuaire. On la notera : `ajouter_entree(annuaire, nom, prenom, telephone)`.
2. Écrire une fonction pour supprimer une entrée de l'annuaire. On la notera : `supprimer_entree(annuaire, nom, prenom)`.
3. Écrire une fonction pour rechercher une entrée par nom ou prénom. On la notera : `rechercher_entree(annuaire, nom=None, prenom=None)`.
4. Écrire une fonction pour afficher toutes les entrées de l'annuaire. On la notera : `afficher_annuaire(annuaire)`.

1 QCM Listes, tuples et dictionnaires

Énoncé p. 102

1 (a) Réponse [b]. Il s'agit ici d'un item. L'index de « Ville » est « 0 ».
(b) Réponse [a]. Il s'agit d'une liste.

2 Réponse [c]. En effet, l'indexation commence à « 0 » donc l'index de « vert » est « 1 » (et non « 2 »). Quant à la première proposition (« l'index de `couleur` est 5 »), elle n'a pas de sens car une liste n'a pas d'index. Seuls les items ont un index.

3 Réponse [a]. En effet, `liste` est une liste de chaînes de caractères. Or, `liste.count(1)` demande le nombre d'occurrences du nombre 1 (et non de la chaîne « 1 »). Comme il n'y a pas de nombres dans `liste`, l'instruction renvoie la valeur : 0.

4 Réponse [d]. En effet, en tapant « `lettres.append('c')` », on ajoute un item à la liste `lettres`. Comme il y en avait 5, il y en a désormais 6.

De plus, l'instruction `len(lettre)` donne la longueur de la liste, c'est-à-dire le nombres d'items de la liste.

2 QCM Traitement de données

Énoncé p. 103

1 Réponse [c]. Nous souhaitons construire une table donc il doit y avoir deux paires de crochets ; ainsi, les réponses [a] et [b] sont impossibles.

De plus, la construction des lignes s'écrit avant la boucle « for ». La réponse [d] est alors incorrecte.

2 Réponse [d]. La syntaxe commence nécessairement par « with open » donc les réponses [a] et [b] sont incorrectes. De plus, après avoir ouvert le fichier CSV, on lui donne un alias avec la syntaxe « as ... » ; la réponse [c] est alors incorrecte car il n'y a pas d'alias après l'ouverture du fichier. Cet alias est utilisé ensuite dans la fonction `csv.reader(<mon alias>)`. Dans la réponse correcte, l'alias utilisé est « fic » (dans le cours, l'alias utilisé est « csvfile », mais on peut prendre l'alias que l'on veut).

3 Réponse [b]. En effet, la réponse [a] ne peut pas être correcte car les items de `table` sont des listes et non des nombres ; par conséquent, si on souhaite tester les items de la table, ce test doit concerner des listes et non des nombres. Par exemple, « `if [1,2,3] in table` » est correct (si on sait que les items de `table` sont des listes à 3 colonnes).

Pour les mêmes raisons, la réponse [c] n'a pas de sens (car il y a toujours le test « `if 7 in table` »).

Quant à la réponse [d], elle aurait pu avoir un sens si elle avait été écrite ainsi :

```
for i in table: for j in i: if 7 == j: <instructions>
```

Mais avec « `in` » à la place de « == », elle n'a plus de sens.

COURS INTERROS CORRIGÉS

4 Réponse $\boxed{\text{a}}$. On définit une fonction `f` en la nommant puis en disant à quoi elle est égale (on met donc le signe « = » juste après son nom, suivi de « lambda » et de la (ou des) variable(s). Ensuite, on définit l'expression de l'image après avoir écrit « : ».

3 V/F Listes, tuples et dictionnaires

Énoncé p. 104

1 L'affirmation est *fausse*. Telle que définie, la variable `individu` est un tuple, donc par définition, on ne peut pas modifier les données stockées.

2 L'affirmation est *vraie*. En effet, la fonction `split` sépare une chaîne de caractères selon le caractère informé entre les parenthèses et chaque partie est stockée dans une liste.

Le caractère selon lequel on souhaite couper la variable `phrase` étant l'espace, on compte le nombre de mots séparés par un espace : il y en a sept. Donc la liste créée a sept items.

3 L'affirmation est *fausse*. La fonction `tri()` n'existe pas. En revanche, `mots.sort()` range les items dans l'ordre alphabétique.

4 Un air de déjà-vu

Énoncé p. 104

Lycée polyvalent Emile Zola, Aix-en-Provence

Voyons étape par étape ce que fait le programme :

- la ligne 1 demande d'entrer un nom et un prénom, que l'on stocke dans la variable `nom_prenom` ;
- `nom_prenom.split(' ')` sépare la variable `nom_prenom` de part et d'autre du caractère « espace » – à l'aide de la fonction `split` – et donne une liste de deux items (le nom et le prénom). Donc `nom_prenom.split(' ')[0]` représente le premier item, donc le nom, stocké dans la variable `nom` ;
- `nom[0]` représente le premier caractère de la chaîne `nom` donc `nom[0].upper()` représente l'initiale de `nom` en majuscule ;
- les deux lignes suivantes définissent de la même façon l'initiale en majuscule du prénom.

Ainsi, les variables `a` et `b` renferment les initiales en majuscules du nom et du prénom, variables affichées par le programme.

COURS
INTERROS
CORRIGÉS

5 Stocker des valeurs

Énoncé p. 104

Lycée Camille Jullian, Bordeaux

Un programme possible est le suivant :

```
valeurs = []

for n in range(11):
    valeurs.extend([n*n-2*n+3])

print(valeurs)
```

Explications :

- on commence par définir la variable `valeurs` comme une liste (vide) ;
- on parcourt ensuite les nombres entiers de 0 à 10 à l'aide de `range(11)` : ne pas oublier que la fonction `range(a)` commence à 0 et finit à $(a-1)$. C'est la raison pour laquelle il faut aller jusqu'à 11 pour afficher tous les nombres A jusqu'à $n = 10$;
- on utilise ensuite la méthode `extend` pour ajouter les différentes valeurs de A en fonction de n.

ATTENTION

On a ici utilisé des crochets à l'intérieur de la méthode `extend` car `n*n-2*n+3` est considéré comme un nombre de classe *int*, et non comme un item de liste. Donc sans crochets, Python renvoie l'erreur : « `TypeError: 'int' object is not iterable` ».

6 Manipulations simples

Énoncé p. 105

Lycée François Mauriac, Bordeaux

On commence par initialiser la liste :

```
liste = [45 , 17 , 89 , 38 , 10 , 74]
```

Ensuite, on trie la liste et on affiche le résultat :

```
Retourne: [10, 17, 38, 45, 74, 89]
liste.sort()
print(liste)
```

CORRIGÉS

On ajoute l'élément « 12 » à la liste et on affiche la nouvelle liste :

```
Retourne : [10, 17, 38, 45, 74, 89, 12]
liste.append(12)
print(liste)
```

On inverse la liste et on affiche le résultat :

```
Retourne : [12, 89, 74, 45, 38, 17, 10]
liste.reverse()
print(liste)
```

On affiche l'indice de l'élément « 10 » :

```
Retourne : 6
print(liste.index(10))
```

On enlève l'élément « 38 » on on affiche le résultat :

```
Retourne : [12, 89, 74, 45, 17, 10]
liste.remove(38)
print(liste)
```

On affiche maintenant la sous-liste du 2e au 3e élément :

```
Retourne : [89, 74]
print(liste[1:3])
```

On affiche la sous-liste du début au 2e élément :

```
Retourne : [12, 89]
print(liste[:2])
```

On affiche la sous-liste du 3e élément au dernier :

```
[74, 45, 17, 10]
print(liste[2:len(liste)])
```

On affiche le dernier élément en utilisant un indice négatif :

```
Retourne : 10
print(liste[-1])
```

7 Construction d'une liste spéciale

Énoncé p. 105

Lycée Sophie Germain, Paris

1

```
n = int(input("Entrez un nombre entier : "))
liste = [ ]

for i in range(1,n+1):
    liste.extend([1,i])

print(liste)
```

2

```
n = int(input("Entrez un nombre entier : "))
liste = [ ]

for i in range(1,n+1):
  liste_tmp = [ ]
  for j in range(1,i+1):
    liste_tmp.extend([j])
  liste.extend(liste_tmp)
print(liste)
```

8 La fonction range

Énoncé p. 105

Lycée Turgot, Paris

1

```
liste = [ ]
for i in range(1,11):
    liste.append(i)

somme = 0
for i in liste:
    somme += i

print(somme)

moyenne = somme/len(liste)

print(moyenne)
```

CORRIGÉS

2

```
somme = 0
for n in [100,1000,10000]:
    somme = 0
    for i in range(1,n):
        somme += 1/(i*i)
    print("Pour n =",n,"la somme est :",somme)
```

L'utilisation de la boucle « `for n in [100,1000,10000]` » nous a permis d'écrire moins de lignes qu'en utilisant « `for i in range(1,100)` », « `for i in range(1,1000)` » et « `for i in range(1,10000)` ».

3

```
somme = 0
c = 0
for i in range(50,100,2):
    c += 1
    somme += i*c
print(somme)
```

9 Créer une liste à partir de deux

Énoncé p. 106

Lycée François Villon, Les Mureaux

```
chaine1 = "abc"
chaine2 = "de"
liste = []

for i in chaine1:
    for j in chaine2:
        liste.append(i+j)
print(liste)
```

Explications :

- on commence par définir les chaînes de caractères, et on définit la liste comme étant vide afin de pouvoir y ajouter des items ;
- on parcourt d'abord la chaîne `chaine1` avec la boucle `for i in chaine1` ;
- pour chaque lettre de `chaine1`, on parcourt la chaîne `chaine2` avec la boucle `for j in chaine2` ;
- à ce stade, `i` représente un caractère de `chaine1` et `j`, un caractère de `chaine2`. On les concatène avec l'instruction `i+j` et on insère le résultat dans `liste` avec l'instruction `liste.append(i+j)`.

10 Compréhensions de liste

Énoncé p. 106

Lycée Augustin Fresnel, Caen

1. `[ord(c) for c in "Hello World!"]`
2. `[i for i in range(0,91)]`
3. `from random import randint`
   `[randint(0, 10)*2 for i in range(10)]`
4. `t = [[i*10+j for j in range(10)] for i in range(10)]`
5. On utilise une compréhension de liste avec `if` :
   `[mot for mot in mots if len(mot) % 2 == 0]`
6. On utilise une compréhension de liste avec `if` et la fonction de conversion de nombre en chaîne : `str` :
   `[n for n in nb if '3' in str(n)]`

11 Pythagore à la rescousse

Énoncé p. 107

Lycée André Theuriet, Civray

1.

```
def is_square(t):
    if t[0]**2+t[1]**2 == t[2]**2 or t[0]**2+t
    [2]**2 == t[1]**2 or t[2]**2+t[1]**2 == t[0]**2:
        print("Ce triangle est rectangle.")
    else:
        print("Ce triangle n'est pas rectangle.")
```

Remarque : dans la mesure où la fonction `sort()` n'est pas disponible pour les tuples, on ne peut pas trier les items et on est donc obligé de tester les trois égalités (au cas où les côtés ne seraient pas dans l'ordre croissant).

2. À la vue de la remarque faite à la question précédente, l'utilisation d'un triplet n'est pas judicieuse ici. Il serait préférable d'utiliser une liste afin de pouvoir trier les mesures dans l'ordre croissant et minimiser les conditions :

```
def is_square(liste):
    liste.sort()
    if liste[0]**2+liste[1]**2 == liste[2]**2:
        print("Ce triangle est rectangle.")
    else:
        print("Ce triangle n'est pas rectangle.")
```

CORRIGÉS

12 Racines d'un trinôme

Énoncé p. 107

Lycée Henri Alain-Fournier, Bourges

```
def racines(coef):
    a, b, c, delta = coef[0], coef[1], coef[2], b*b-4*a
    *c
    if delta<0:
        result = 0
    else:
        if delta == 0:
            result = 1, -b/(2*a)
        else:
            result = 2, (-b-delta**0.5)/(2*a), (-b+
    delta**0.5)/(2*a)
    return result
```

13 Inventaire et tri

Énoncé p. 108

Lycée Raoul Follereau, Nevers

```
epicerie = [
  ("tomates",20),
  ("pommes",10),
  ("carottes",5),
  ("poires",3),
  ("bananes",4),
  ("ananas",1)
  ]

tri = 0

while tri !=1 and tri !=2:
    tri = int(input("Tri alphanumérique (1) ou tri
    alphabétique (2) ? "))

if tri == 2:
    epicerie.sort()
    print(epicerie)
else:
    epicerie_inverse = []
    for i in range(len(epicerie)-1):
        epicerie_inverse.extend([(epicerie[i][1],
    epicerie[i][0])])
    epicerie_inverse.sort()
    print(epicerie_inverse)
```

Pour le tri alphanumérique, il n'y a aucun problème : la fonction `sort()` fait le travail.

En revanche, pour trier suivant le deuxième argument des tuples, c'est plus compliqué, d'où l'idée de construire une autre liste en inversant les arguments des tuples.

14 Minimum, maximum et moyenne

Énoncé p. 108

Lycée Sainte-Clotilde, Strasbourg

```
def min_max_moy(liste):
    liste.sort()
    somme = 0
    for i in liste:
        somme += i
    t = (liste[0],liste[len(liste)-1],somme/len(liste))
    return t
```

Explications :

- `liste.sort()` commence part trier la liste dans l'ordre croissant ; ainsi, la plus petite valeur de la liste sera `liste[0]` et la plus grande, `liste[len(liste)-1]`. Il ne faut pas oublier que `len(liste)` donne le nombre d'items de la liste mais il y a un décalage d'index car on commence à l'index 0. Par exemple, pour une liste à 10 items, le dernier sera `liste[9]` ;
- on calcule ensuite la somme des valeurs stockées dans la liste en définissant d'abord la variable `somme` à 0, puis en lui ajoutant toutes les valeurs avec la boucle `for i in liste`. La moyenne sera calculée en divisant `somme` par `len(liste)`.

15 Le Scrabble

Énoncé p. 108

Lycée Sophie Germain, Paris

1. La variable `scrabble` étant définie avec des accolades, et contenant des données de la forme « A : B » séparées par une virgule, elle est du type *dictionnaire*.
2. La variable `scrabble` contient les lettres données dans le tableau de l'énoncé ainsi que les points associés à chacune des lettres.
3. La boucle « `for i in mot` » signifie que l'on parcourt le mot entré au clavier. Rappelons que `scrabble.get(i)` donne la valeur attribuée à la variable `i` (donc ici, le nombre de points pour la lettre stockée dans `i`).

COURS
INTERROS
CORRIGÉS

CORRIGÉS

Le tableau suivant donne pas à pas les valeurs successives de la variable `somme` :

Valeur de `i`		K	A	K	I
Valeur de `scrabble.get(i)`		5	1	5	1
Valeur de `somme`	0	5	6	11	12

4 À travers l'exemple précédent, on se rend compte que ce programme calcule le total des points d'un mot.

16 Tableau physico-chimique

Énoncé p. 109

Lycée Camille Jullian, Bordeaux

Il suffit de taper :

```
dico = {
  "Au" : {
    'Te':2970,
    'Tf':1063,
    'Z':79,
    'A':196.967
  } ,
  "Ga" : {
    'Te':2237,
    'Tf':29.8,
    'Z':31,
    'A':69.72
  }
}
```

17 Base de données

Énoncé p. 109

Lycée Sainte-Clotilde, Strasbourg

1 **(a)** La variable `registre` étant définie par des crochets, elle est du type *liste*.

(b) La valeur `registre[0]` étant « 1 », son type est *int*.

(c) La valeur `registre[1]` est : « {'nom' :'HILUM' , 'prenom' :'Mohan' , 'adresse' :'224 avenue des fleurs 75001 Paris', 'age' :67, 'taille' :178, 'poids' :81} », donc son type est *dictionnaire*.

2 L'instruction `registre[1]['nom']` retourne : « 'HILUM' ».

Le type de `registre[1]['nom']` est donc *chaîne de caractères*.

3 D'après le cours, pour ajouter un item à une liste, il faut utiliser l'instruction :

```
registre.extend([2,{'nom':'DUPONT' , 'prenom':'Kévin' ,
'adresse':'5 allée des ponts 33000 Bordeaux', 'age':24,
'taille':187, 'poids':95}])
```

⚠ ATTENTION

L'énoncé précise bien que l'on doit utiliser une seule instruction, et non deux. Donc la méthode append ne peut pas fonctionner ici car elle ajouterait un item de type *liste*, ce qui n'est pas souhaité.

4

```
registre, ligne, rep = [], 0, "O"
while rep == "O":
    nom = input("Nom : ")
    prenom = input("Prénom : ")
    adresse = input("Adresse : ")
    age = int(input("Âge : "))
    taille = int(input("Taille (en cm) : "))
    poids = int(input("Poids (en kg) : "))
    ligne += 1
    registre.extend([ligne,{'nom':nom,'prenom':
    prenom,'adresse':adresse,'age':age,'taille':
    taille,
    'poids':poids}])
    rep = input("Voulez-vous poursuivre la saisie (
    O = oui) ? ")
    rep.upper()
```

18 Création d'un dictionnaire

Énoncé p. 110

Lycée Turgot, Paris

```
def compte_lettres(chaine):
    dico = {}
    for i in chaine:
        if i in dico: # si la lettre figure déjà
            dico[i] = dico.get(i)+1
        else: # sinon, on la créé
            dico[i] = 1
    return dico
```

Explications :

- on commence par définir la variable `dico` comme étant un dictionnaire (vide) ;
- on parcourt ensuite la chaîne de caractères `chaine` avec la boucle `for i in chaine` ;
- on teste avant tout si le caractère sur lequel on est (représenté par la variable `i`) figure dans le dictionnaire avec l’instruction `if i in dico`. Si tel est le cas, on incrémente le nombre attribué à cette lettre avec l’instruction `dico[i] = dico.get(i)+1` ;
- sinon, on créé un item correspondant à cette lettre dans le dictionnaire et on lui attribue la valeur « 1 ».

19 De l’hexadécimal au binaire

Énoncé p. 111

Lycée polyvalent Émile Zola, Aix-en-Provence

```
dico = {’0’:0, ’1’:1, ’2’:2, ’3’:3, ’4’:4, ’5’:5, ’6’:6,
    ’7’:7, ’8’:8, ’9’:9, ’A’:10, ’B’:11, ’C’:12,’D’:13,
     ’E’:14, ’F’:15}

def convertdecbin(n):
    b = ’’
    while n != 0:
        q = int(n/2)
        b = str(n-q*2)+b
        n = q
    return b


def converthexabin(n):
    hexa = ’’
    for i in n:
        h = convertdecbin(dico[i])
        if len(h)<4:
            if len(h) == 3:
                h = ’0’+h
            else:
                if len(h) == 2:
                    h = ’00’+h
                else:
                    h = ’000’+h
        hexa += h
    return hexa
```

Afin de rendre plus compréhensible le programme, deux fonctions ont été créées : `convertdecbin(n)` convertit le nombre n (exprimé en décimal) en binaire et retourne sous forme de chaîne de caractères le résultat (c'est la partie issue de l'exercice 12 du chapitre 1).

Ensuite, la fonction `converthexabin(n)` a été définie, qui convertit le nombre n (exprimé en décimal) en binaire et retourne le résultat sous la forme d'une chaîne de caractères. Dans cette fonction, on parcourt la variable `n` grâce à la boucle `for i in n:` et pour chaque caractère `i` rencontré, on commence d'abord par récupérer la clé correspondant à la valeur `i` dans le dictionnaire `dico`, puis on convertit cette valeur (qui est décimale) en binaire à l'aide de la fonction `convertdecbin`. Ensuite, on teste si la valeur `h` obtenue a une longueur de 4 bits ; si tel n'est pas le cas (`if len(h)<4`), on complète par des « 0 » afin que `h` ait une longueur de 4 bits. À la fin de ce test, une fois que l'on est assuré que `h` tient sur 4 bits, on concatène le résultat à la valeur de `hexa` précédente (`hexa += h`). Enfin, la variable `n` étant parcourue en entier, on retourne la valeur de la variable `hexa`, qui est la conversion demandée.

20 Théorie des graphes

Énoncé p. 111

Lycée Jean Macé, Niort

On doit afficher toutes les instances de la valeur 1, comme dans le paragraphe 4.2.2 du cours.

On utilise alors le même modèle de code :

```
for ligne in adj:
    if 1 in ligne:
  print('Les sommets',adj.index(ligne)+1,'et',ligne.
    index(1)+1,'sont connectés.')
```

21 Présence dans une base de données

Énoncé p. 111

Lycée Jean-Joseph Fourier, Auxerre

1 Le code suivant permet d'afficher la phrase « Informations présentes. » dans le cas où l'une des deux informations (pseudo ou courriel) est dans la table :

```
for ligne in data:
    if 'BigArnaudDesMers' in ligne or '
    bigarnaudofsea@gmail.com' in ligne:
        print('Informations présentes.')
```

2 Le pseudo peut comporter des majuscules et des minuscules. Il est donc intéressant de comparer les pseudos uniquement en minuscules.

Il y a deux possibilités.

- 1re possibilité : le pseudo et le courriel sont enregistrés en minuscules dans la table. Dans ce cas, une légère modification du code précédent est à apporter :

```
for ligne in data:
    if 'bigarnauddesmers' in ligne or '
    bigarnaudofsea@gmail.com' in ligne:
        print('Informations présentes.')
```

- 2e possibilité : les pseudo et courriel sont enregistrés sans modification par rapport à la saisie initiale.

  Dans ce cas, il faut parcourir tous les items et les convertir en minuscules, comme dans le programme page ci-dessous.

```
presence = False
for ligne in data:
    for col in ligne:
        if col.lower() == 'bigarnauddesmers' or
    col.lower() == 'bigarnaudofsea@gmail.com':
            presence = True
if presence == True:
    print('Informations présentes.')
```

  Il est nécessaire d'utiliser un booléen afin de ne pas afficher plusieurs fois la phrase dans l'éventualité où le courriel et le pseudo figurent dans la table.

Dans le doute (comme aucune information n'est connue sur la table et la casse de ses items), il est préférable de choisir le second code. Mais quand on est administrateur d'un site internet, on pense à ce genre de choses et on enregistre les données au format le plus intéressant pour les manipulations futures.

22 Fichier INSEE

Énoncé p. 112

Lycée Henri Bergson, Angers

1 Il est possible d'importer les données à l'aide des lignes suivantes :

```
import csv

with open('insee2018.csv',newline='') as file:
    data = list(csv.reader(file))
```

Remarque : le fichier XLS initial comporte 3 540 lignes de données, ce qui est bien trop pour Python (essayez et vous verrez que votre ordinateur va prendre très longtemps avant de finir l'importation). C'est pourquoi nous n'avons pris que 100 lignes lors de l'exportation en CSV.

2 On trie d'abord selon la dernière colonne, puis on parcourt la table en affichant ce que l'on souhaite uniquement si la valeur de la dernière colonne est inférieure ou égale à 1 500 (voir programme page suivate).

Il est important de retenir que les données importées sont au format texte ; ainsi, quand on souhaite faire un test numérique sur une valeur, on doit convertir cette dernière en nombre en utilisant la fonction `int`.

```
data.sort(key=lambda col : col[9])

for ligne in data:
    if int(ligne[9]) <= 1500:
        print(ligne[6],'(',ligne[1],') -',ligne[9],'
    habitants.')
```

3 Le code ressemble au précédent ; seul le test change :

```
for ligne in data:
    if int(ligne[9])<4000 and int(ligne[9])>2000:
        print(ligne[6],'(',ligne[1],') -',ligne[9],'
    habitants.')
```

4 Le code suivant permet d'afficher les communes dont le nom commence par « Pont- » :

```
for ligne in data:
    if ligne[6][:5] == 'Pont-':
        print(ligne[6],'(',ligne[1],') -',ligne[9],'
    habitants.')
```

On aurait aussi pu écrire en test : `if 'Pont-' in ligne[6]:`.

23 Gestion d'une bibliothèque

Énoncé p. 112

Lycée François Mauriac, Bordeaux

Télécharger l'ensemble des réponses dans un programme.

1

```
def ajouter_livre(livres, titre, auteur, annee):
    livre = {"titre": titre, "auteur": auteur, "
    annee": annee, "disponible": True}
    livres.append(livre)
```

CORRIGÉS

2

```
def ajouter_emprunteur(emprunteurs, nom):
    emprunteur = {"nom": nom, "livres_empruntes":
    []}
    emprunteurs.append(emprunteur)
```

3

```
def emprunter_livre(livres, emprunteurs,
    nom_emprunteur, titre_livre):
    for emprunteur in emprunteurs:
        if emprunteur["nom"] == nom_emprunteur:
            for livre in livres:
                if livre["titre"] == titre_livre
    and livre["disponible"]:
                    livre["disponible"] = False
                    emprunteur["livres_empruntes"].
    append(titre_livre)
                    return True
    return False
```

4

```
def rendre_livre(livres, emprunteurs, nom_emprunteur
    , titre_livre):
    for emprunteur in emprunteurs:
        if emprunteur["nom"] == nom_emprunteur:
            if titre_livre in emprunteur["
    livres_empruntes"]:
                emprunteur["livres_empruntes"].
    remove(titre_livre)
                for livre in livres:
                    if livre["titre"] ==
    titre_livre:
                        livre["disponible"] = True
                        return True
    return False
```

5

```
def afficher_livres_disponibles(livres):
    for livre in livres:
        if livre["disponible"]:
            print(f"{livre['titre']} par {livre['
    auteur']} ({livre['annee']})")
```

6
```
def afficher_emprunteurs(emprunteurs):
    for emprunteur in emprunteurs:
        print(f"{emprunteur['nom']} a emprunté : {',
 '.join(emprunteur['livres_empruntes']) if
    emprunteur['livres_empruntes'] else 'aucun livre
    '}")
```

24 Gestion d'un annuaire

Énoncé p. 113

Lycée Lycée Camille Claudel, Blois

Télécharger l'ensemble des réponses dans un programme.

1
```
def ajouter_entree(annuaire, nom, prenom, telephone)
:
    clé = (nom, prenom)
    annuaire[clé] = telephone
```

2
```
def supprimer_entree(annuaire, nom, prenom):
    clé = (nom, prenom)
    if clé in annuaire:
        del annuaire[clé]
        return True
    return False
```

3
```
def rechercher_entree(annuaire, nom=None, prenom=
    None):
    for clé, telephone in annuaire.items():
        if (nom is None or clé[0] == nom) and (
    prenom is None or clé[1] == prenom):
            return clé, telephone
    return None
```

4
```
def afficher_annuaire(annuaire):
    for clé, telephone in annuaire.items():
        print(f"Nom: {clé[0]}, Prénom: {clé[1]}, Té
    léphone: {telephone}")
```

COURS INTERROS CORRIGÉS

Chapitre 4

Langages et programmation

Plan du chapitre

1. Constructions élémentaires
2. Points communs et différences des langages
3. Spécifications et tests
4. Tests

Exercice type

Lycée Jean-Joseph Fourier, Auxerre

On donne le programme suivant, écrit en Fortran :

```
program ranger
  implicit none
  character(len=20) :: mot1, mot2, mot
  print *,'Donnez un premier mot'
  read *, mot1
  print *,'Donnez un second mot'
  read *, mot2
  if (mot1.gt.mot2) then
    mot = mot1
    mot1 = mot2
    mot2 = mot
  end if
  print *,'Les deux mots rangés sont :'
  print *, mot1, mot2
end program ranger
```

1 En vous référant aux langages que vous connaissez, que peut signifier « `len=20` » ?

2 Que représente la commande « `gt` » présente dans l'instruction conditionnelle ?

3 Que fait ce programme ?

Voir corrigé page 140

À RETENIR

Pour faire tourner en ligne un programme (C, PHP, HTML, etc.), vous pouvez vous rendre sur le site :

`https://www.onlinegdb.com`

COURS

Nous avons vu dans les premiers chapitres des programmes écrits en langage Python.

Python est un langage parmi d'autres. Nous allons voir dans ce chapitre quels sont les points communs à tous les langages, et quelles sont aussi les différences.

Citons quelques langages dans l'ordre alphabétique :

Basic, C, C++, Caml, Fortran, Java, Pascal, Perl, PHP Python, TEX...

Sur la page :

`https://fr.wikipedia.org/wiki/Liste_de_langages_de_programmation`,

on peut visualiser l'ensemble des langages informatiques qui existent.

1 Constructions élémentaires

1.1 Séquences et affectations

Définitions 1

- Une *séquence* est une suite d'instructions.
- Une *affectation* est le fait de donner une valeur à une variable.

En algorithmique, une affectation est symbolisée par une flèche de la valeur (à droite) vers la variable (à gauche).

Exemple :

```
a ← 3
```

En revanche, dans un programme, l'affectation est symbolisée par une égalité.

Exemples

- En Python, le code suivant :
  ```
  a, texte = 32, "Bonjour"
  ```
  affecte la valeur 32 à la variable `a` et la valeur « Bonjour » à la variable `texte`.
- En C, les mêmes affectations s'écrivent :
  ```
  int a = 32;
  char texte[] = "Bonjour";
  ```
- En PHP, cela donne :
  ```
  $a = 32;
  $texte = "Bonjour";
  ```

Il y a certes quelques différences dans la syntaxe, mais on arrive à comprendre, même si l'on ne connaît pas parfaitement le langage.

1.2 Instructions conditionnelles

1.2.1 - Le modèle « if ... then ... else ... »

Les instructions conditionnelles sont traduites en algorithmique par :

```
SI <condition> ALORS <instructions> SINON <instructions>
```

En programmation, la <condition> prend la forme :

- d'un test d'égalité (« SI a == b »),
- d'un test d'inégalité (« SI a <=b »),
- d'un test booléen (« SI b ALORS ... » ou « SI !b ALORS ... »).

Exemples

- En Python, nous avons vu que la syntaxe était la suivante :

```
if a==b:
    ... instructions 1 ...
else:
    ... instructions2 ...
```

- En C, ou en PHP, cela donne :

```
if (a==b) {
    ... instructions1 ...
}
else {
    ... instructions2 ...
}
```

Notez la présente de délimiteurs (les accolades) contrairement à Python (où un « : » et une indentation automatique suffisent).

1.2.2 - Le modèle « switch »

Ce modèle est une façon plus condensée d'écrire certaines instructions conditionnelles avec « if ».

Exemples

- En PHP, on peut avoir par exemple :

```
switch ($i) {
    case 0: instructions si $i=0 ; break;
    case 1: instructions si $i=1 ; break;
    default : instructions si $i est différent de 0 et 1;
}
```

COURS

Exemples

- En C, cela donne :

```
switch(i) {
    case 0: instructions si i=0; break;
    case 1: instructions si i=1; break;
    default: instructions si i est différent de 0 et 1;
}
```

- Avec Python, c'est moins simple :

```
if i==0:
    instructions si i=0
elif i==1:
    instructions si i=1
else:
    instructions si i est différent de 0 et 1
```

1.3 Les boucles

1.3.1 - Les boucles bornées

Les boucles bornées sont traduites en algorithmique par :

```
POUR i ALLANT DE ... à ... FAIRE
 ... instructions ...
FIN DU POUR
```

En programmation, on utilise les mots anglais adéquats (comme toujours).

Exemples

- En Python, cela donne :

```
for i in range(50): (exécute 50 fois les instructions)
    ... instructions ...
```

- En C, cela donne :

```
int i;
for (i=0 ; i<50 ; i++) {
    ... instructions ...
}
```

- En PHP, cela donne :

```
for ($i=1 ; $i<=50 ; $i++) {
    ... instructions ...
}
```

1.3.2 - Les boucles non bornées

En algorithmique, ce sont les boucles du type :

```
TANT QUE <condition> FAIRE
 ... instructions ...
FIN DU TANT QUE
```

En programmation, le « TANT QUE » est traduit par « while ».

Exemples

- En Python, cela donne :

```
while a<50:
    ... instructions ...
```

- En C, cela donne :

```
while (a<50) {
    ... instructions ...
}
```

ou :

```
do {
    ... instructions ...
} while (a<50)
```

La différence est qu'avec la 1re boucle « while », si la condition « a<50 » n'est pas satisfaite dès le début, les instructions ne seront pas exécutées alors qu'avec la 2e boucle « do ... while », les instructions sont exécutées au moins une fois (car le test est effectué après le premier passage dans la boucle).

- En PHP, cela donne :

```
while ($i<=50) {
    ... instructions ...
}
```

ATTENTION

Les boucles « while » sont très souvent les victimes d'une mauvaise programmation quand on débute : elles peuvent être infinies si l'on ne fait pas attention aux instructions. Par exemple, en C :

```
int i = 1;
int s = 0;
while (i < 10) {
    s = s + i;
}
```

est une boucle infinie car la valeur de la variable `i` n'est pas modifiée dans la boucle.

COURS INTERROS CORRIGÉS

COURS

ATTENTION

Ainsi, `i` vaut toujours 1 et la condition « i<10 » est toujours réalisée. Il faut écrire par exemple :

```
int i = 1;
int s = 0;
while (i < 10) {
    s = s + i;
    i++;
}
```

Dans ce cas, `i` est incrémentée à chaque passage dans la boucle et prendra tôt ou tard une valeur supérieure ou égale à 10 ; la boucle s'arrêtera alors.

1.4 Les fonctions

1.4.1 - Définition

Les fonctions sont des blocs d'instructions admettant aucun, un ou plusieurs arguments. Elles sont très utiles pour éviter d'écrire un grand nombre de fois des instructions qui doivent être exécutées à de multiples reprises.

Exemples

- En Python, on peut avoir par exemple :

  ```
  def toto(a,b):
      return a*a+b*b-a*b
  ```

  qui définit la fonction $f(a,b) = a^2 + b^2 - ab$.
- En C, cela donne :

  ```
  double toto(double a, double b) {
      double r = a*a+b*b-a*b;
      return r;
  }
  ```

  Le type « double » devant la fonction (ainsi que devant ses arguments) signifie que ce qui suit est un nombre « réel » (ou presque... voir chapitre 2). Le résultat retourné par la fonction est donc un « réel ».
  Si la fonction ne retourne rien, le type « void » est attendu.
- En PHP, cela donne :

  ```
  function toto($a,$b) {
      return $a*$a+$b*$b-$a*$b;
  }
  ```

Remarque : en langage C, un programme est toujours contenu dans au moins une fonction ; par défaut, c'est la fonction `main()`.

1.4.2 - Prototyper une fonction

Définition 2

Le *prototype* d'une fonction est la donnée :
- de son type (entier, flottant...) ;
- de son nom ;
- du type de ses arguments.

Les prototypes sont en général utilisés sur les gros projets (contenant beaucoup de fichiers). Ils permettent de présenter dans un fichier une fonction définie dans un autre fichier.

En C, le prototypage d'une fonction se fait en général (même si cela n'est pas obligatoire pour un programme court) en début de code.

Pour un programme long, le prototypage est écrit dans un fichier externe (avec l'extension `.h`) appelé en début de code :

```
#include <monfichier.h>
```

2 Points communs et différences des langages

Avant tout chose, il ne vous est pas demandé de connaître *tous* les langages de programmation (existe-t-il une personne qui les connaisse tous ?).

En revanche, il est conseillé de connaître au moins un langage car la structure d'un programme, quel que soit son langage, restera la même.

2.1 Les instructions conditionnelles

Elles comportent toujours le mot « if » (pour les plus simples) ou « switch » (pour les instructions plus conséquentes).

Elles se terminent parfois par « end if » ou « endif » quand il n'y a pas d'accolades. Quelques fois, elles se terminent par « fi » (comme en TEX), mais c'est très rare.

En revanche, selon les langages, la syntaxe des conditions qui suivent « if » peut varier :
- très souvent, les comparaisons utilisent « == », « > », « >= », etc., mais dans certains langages, les comparaisons sont écrites avec des lettres (« gt » pour « > » ou encore « lt » pour « < » par exemple) ;
- en Python, les opérateurs booléens s'écrivent « or », « and », ou « not », mais dans de nombreux langages, ils s'écrivent respectivement « || », « && » et « ! ».

COURS
INTERROS
CORRIGÉS

2.2 Les boucles (itérations)

Elles comportent souvent les mots « for » ou « while », mais quand ce n'est pas le cas, il peut y avoir un « do ».

Quelques fois, le mot « break » ou « exit » apparaît pour « casser » la boucle, c'est-à-dire sortir prématurément de la boucle sans aller jusqu'au bout de l'itération.

2.3 Les fonctions

Là encore, un mot apparaît presque toujours : « function ». Si tel n'est pas le cas, le mot « def » peut être utilisé (comme en Python).

2.4 Une différence importante

L'une des difficultés majeures pour un débutant en programmation réside dans la déclaration des variables et la gestion des types. Certains langages, comme Python ou PHP, utilisent un typage dynamique : il n'est pas nécessaire de préciser le type d'une variable lors de sa création, car le langage le détermine automatiquement en fonction de la valeur affectée. En revanche, d'autres langages, comme C ou Java, utilisent un typage statique, où le type de chaque variable doit être déclaré explicitement dès sa création. Ces langages proposent souvent plusieurs types numériques avec des tailles et des précisions différentes (`int`, `long`, `short`, `double`, `float`, etc.), ce qui peut complexifier la tâche pour un programmeur débutant. Consulter la documentation propre au langage est donc primordial.

Par exemple, en *Java* :

- le type « int » désigne un entier entre -2^{31} et $2^{31} - 1$;

Mais en C :

- le type « int » désigne un entier entre $-2^{15} + 1$ et $2^{15} - 1$;

Solution de l'exercice type — **Lycée Jean-Joseph Fourier, Auxerre**

1 Nous avons vu qu'en Python, « len » était en rapport avec la longueur d'une chaîne de caractères (abréviation de *length* = longueur).
Ainsi, « `len=20` » pourrait désigner la longueur maximale des chaînes `mot1`, `mot2` et `mot`.

2 « `gt` » est l'équivalent du test « > ».

3 L'instruction conditionnelle `if (mot1.gt.mot2)` laisse à supposer que les instructions suivantes s'exécutent quand la longueur de `mot1` est strictement plus grande que celle de `mot2`. Dans ce cas, les instructions intervertissent les deux variables.
Le programme semble donc trier les chaînes de caractères selon l'ordre croissant de leur longueur.

Voir énoncé page 133

3 Spécifications et tests

3.1 Documentation

Le code seul ne suffit pas. En effet, le développement et la maintenance d'un programme peuvent impliquer plusieurs personnes et s'étaler sur plusieurs années. Documenter le code est donc essentiel, en utilisant des commentaires pour expliciter certaines lignes ou parties. Bien sûr, cela commence par utiliser des noms explicites pour les variables et fonctions.

Voici deux exemples de la même fonction, la première avec des noms quelconques :

```
def f(a):
    b = []
    for c in a:
        if c == ''.join(reversed(c)):
            b.append(c)
    return b
```

et la seconde avec des noms explicites, beaucoup plus lisible :

```
def extraire_palindromes(liste_mots):
    liste_palindromes = []
    for mot in liste_mots:
        if mot == ''.join(reversed(mot)):
            liste_palindromes.append(mot)
    return liste_palindromes
```

3.2 Documenter une fonction : docstring

Python permet de documenter une fonction en insérant une chaîne de caractères, appelée *docstring*, encadrée par trois guillemets, immédiatement après l'instruction `def`. Cette chaîne peut s'étendre sur plusieurs lignes. En voici un exemple simple :

```
def carre(x):
    """Calcule le carré d'un nombre.
    Entrées :
        le nombre x à mettre au carré (int ou float)

    Sortie :
        le carré de x (int ou float)
    """
    return x * x
```

Cette documentation est alors accessible depuis la console : `help(carre)`.

COURS

3.3 Préconditions et postconditions

Définition 3 : Spécification

La docstring d'une fonction précise les spécifications attendues :
- précondition : conditions sur les arguments (type, valeur attendue, etc.).
- postcondition : ce que la fonction renvoie.

Spécifier une fonction consiste donc à décrire clairement ces deux éléments. Pour garantir ces spécifications, on peut insérer des assertions dans le code. Une assertion vérifie qu'une condition est vraie durant l'exécution : si ce n'est pas le cas, le programme s'arrête et affiche un message d'erreur, que l'on peut personnaliser.

La syntaxe est : `assert <condition>` ou `assert <condition>, "message d'erreur"` où `<condition>` est une expression (test) dont l'évaluation retourne un booléen.

Exemple :

```
def kelvin_to_fahrenheit(temperature):
    """ Convertit une température en Kelvin en degrés
    Fahrenheit.
    Entrée : température en Kelvin (int ou float)
    Sortie : température en degré Fahrenheit (float)."""
    assert type(temperature) == float or type(temperature)
    == int
    assert temperature >= 0, "Une température absolue ne
    peut être négative."
    return ((temperature - 273.15) * 1.8) + 32
```

Remarque : Les assertions sont utiles pendant le développement et le débogage pour détecter les erreurs, mais elles ne sont généralement pas utilisées dans le code final en production.

4 Tests

Même bien documenté et spécifié, un code n'est pas à l'abri d'erreurs. Une bonne pratique consiste à tester rigoureusement une fonction en l'exécutant avec divers exemples de paramètres et en comparant les résultats obtenus aux résultats attendus. Pour cela, il est essentiel de bien connaître les spécifications et d'anticiper les cas particuliers susceptibles de provoquer des erreurs.

Il est courant, dans l'industrie, de rédiger les tests après avoir spécifié la fonction, mais avant de l'implémenter. Cette tâche peut même être confiée à des personnes différentes de celles qui écrivent le code.

Trouver un jeu de tests pertinent est une démarche cruciale : un ensemble de tests bien conçu permet de révéler un grand nombre d'erreurs potentielles, tandis qu'un jeu incomplet risque de laisser passer des erreurs non détectées.

Cependant, un jeu de tests, même étendu, n'est jamais exhaustif car il est impossible de couvrir tous les cas possibles. Ainsi, la réussite à un ensemble de tests ne garantit pas qu'un programme soit exempt de défauts. Néanmoins, plus les tests sont variés et nombreux, plus les erreurs sont susceptibles d'être détectées et corrigées en amont.

Exemple : la fonction `indice_min` reçoit une liste de nombres et retourne l'indice de sa valeur minimale. Elle vérifie également que la liste n'est pas vide et retourne `None` dans ce cas. Un bon jeu de tests pourrait être :

```python
assert indice_min([]) == None
assert indice_min([4, 2, 5]) == 1
assert indice_min([1, 9, 4, 2, 0]) == 4
assert indice_min([1, 2, 3, 4]) == 0
assert indice_min([-2, -4, -9, 0]) == 2
```

Ce jeu de tests couvre différents cas : liste vide, valeurs positives, négatives et différentes positions de la valeur minimale. Il illustre bien l'importance de tester divers scénarios pour minimiser les erreurs.

INTERROS

1 QCM Savoir traduire un programme

5 min

Corrigé p. 151

Pour chacune des questions suivantes, quatre propositions sont faites, mais une seule est correcte. Laquelle ?

1 En Fortran

```
program main
  real :: degreesfahrenheit, degreescelsius
  read *, degreesfahrenheit
  degreescelsius = 5*(degreesfahrenheit-32)/9
  print *, degreescelsius
end program main
```

Ce programme :

a convertit une température réelle exprimée en degrés Fahrenheit en degrés Celsius ;

b convertit une température réelle exprimée en degrés Celsius en Fahrenheit ;

c convertit une température entière exprimée en degrés Fahrenheit en degrés Celsius ;

d convertit une température entière exprimée en degrés Celsius en Fahrenheit.

2 En Java

```
static void main(String[ ] args)
{
  int x,y;
  x = 4;
  while (x <= 10){
    y = 3*x+5;
    System.out.println("f("+x+")="+y);
    x = x+1
  }
}
```

Ce programme :

a affiche l'antécédent de 4 par la fonction $x \mapsto 3x + 5$;

b affiche l'image de 4 par la fonction $x \mapsto 3x + 5$;

c affiche les antécédents de tous les entiers de l'intervalle [4 ; 10] par la fonction $x \mapsto 3x + 5$;

d affiche les images de tous les entiers de l'intervalle [4 ; 10] par la fonction $x \mapsto 3x + 5$.

2 V/F Rédaction de programmes

5 min — Corrigé p. 151

Retrouvez le corrigé de cet exercice en vidéo.

En C

```c
#include <stdio.h>

int convert(double a)
{ return 5*(a-32)/9; }

void main()
{
  double x, y;
  printf("Entrez le premier nombre : ");
  scanf("%lf",&x); /* Entre un nombre. */
  y = convert(x);
  printf("%lf", y);
}
```

Ce programme est-il correctement écrit ?

Interpréter un langage

3 Programme PHP

★ — 10 min — Corrigé p. 151

Lycée Camille Claudel, Blois

En PHP

```php
<?php
$image_path = '/images';
$arr = array(4, 7, 19, 23);
foreach ($arr as $value) {
  echo '<img src="'.$image_path.'/'.$value.'.png"/>'; }
?>
```

Que fait ce programme ?

4 Programme en Java

★ — 10 min — Corrigé p. 152

Lycée Henri Bergson, Angers

Que fait ce programme écrit en Java proposé page suivante ?

INTERROS

En Java

```
public class Main {
    public static void main(String[] args) {
        int[] table = {12, -5, 7, 8, -6, 6, 4, 78, 2};
        long elt = 4;
        int i;
        for (i = 0; i < 8; i++) {
            if (elt == table[i])
                break;
        }
        afficher(i, elt);
    }

    static void afficher(int rang, long val) {
        if (rang == 8)
            System.out.println("valeur : " + val + "
    pas trouvée.");
        else
            System.out.println("valeur : " + val + "
    trouvée au rang : " + rang);
    }
}
```

5 Programme en C

★ 10 min Corrigé p. 152

Lycée Sophie Germain, Paris

Retrouvez le corrigé de cet exercice en vidéo.

Qu'affiche le programme ci-dessous, écrit en C ?

En C

```
#include <stdio.h>
void affiche(int n,int* table);
void tri(int n,int* table);

void main() {
  int tableau[3] = {8,4,6};
  tri(3,tableau);
  affiche(3,tableau);
}
```

(suite du programme page suivante)

suite du programme en C

```
void tri(int n, int* table) {
  int j,temp;
  for (j=0; j<n-1 ; j++) {
    if (table[j]>table[j+1]){
      temp = table[j];
      table[j] = table[j+1];
      table[j+1] = temp;
    }
  }
}
void affiche(int n, int* table) {
  int j;
  for (j=0 ; j<n ; j++) {
    printf("%i ",table[j]);
  }
}
```

Écriture de programmes

6 Du Javascript au Python

★ 10 min Corrigé p. 153

Lycée Lycée Henri Alain-Fournier, Bourges

Convertir le programme suivant, écrit en Javascript, en Python :

```
// Fonction pour calculer la moyenne de trois nombres
function calculerMoyenne(a, b, c) {
    return (a + b + c) / 3;
}
// Saisie des trois nombres
let nombre1 = parseFloat(prompt("Entrez le premier
    nombre:"));
let nombre2 = parseFloat(prompt("Entrez le second
    nombre:"));
let nombre3 = parseFloat(prompt("Entrez le troisième
    nombre:"));
// Calcul de la moyenne
let moyenne = calculerMoyenne(nombre1, nombre2, nombre3
    );
// Affichage du résultat
console.log("La moyenne des trois nombres est: " +
    moyenne);
```

7 Prototyper une fonction

★ 10 min Corrigé p. 153

Lycée Camille Jullian, Bordeaux

Dans chacun des cas suivants, prototyper les fonctions demandées.

1. La fonction `convert` convertit une durée exprimée en minutes en une durée exprimée en heures et minutes. Elle affiche le résultat sous la forme « ... h ... min ».
2. La fonction `FarToCel` convertit une température en degré Fahrenheit en degré Celsius. Elle renvoie le résultat sous forme de nombre.

8 Convertisseur de temps

★ 15 min Corrigé p. 153

Lycée Sainte-Anne, Brest

Retrouvez le corrigé de cet exercice en vidéo.

Écrire un programme en Python ou en C comprenant :

- une partie principale qui demande un temps (exprimé en minutes) ;
- une fonction `convert1` qui convertit le temps saisi en temps exprimé en heures et minutes ;
- une fonction `convert2` qui convertit le temps saisi en temps exprimé en heure décimale (par exemple, 90 minutes est converti en 1,5 heure).

9 Conversions de programmes

★★★ 2 min Corrigé p. 154

Lycée François Mauriac, Bordeaux

1. Traduire le programme Fortran suivant vers Python.

```
program somme
    implicit none
    integer :: a, b, somme
    print *, 'Entrez le premier nombre:'
    read *, a
    print *, 'Entrez le second nombre:'
    read *, b
    somme = a + b
    print *, 'La somme est:', somme
end program somme
```

2. Traduire le programme PHP suivant vers Python.

```
<?php
$nom = readline("Entrez votre nom: ");
echo "Bonjour, " . $nom . "!\n";
?>
```

3 Traduire le programme C suivant vers Python.

```
#include <stdio.h>

int main() {
    int nombre;
    printf("Entrez un nombre: ");
    scanf("%d", &nombre);
    if (nombre % 2 == 0) {
        printf("Le nombre est pair.\n");
    } else {
        printf("Le nombre est impair.\n");
    }
    return 0;
}
```

Spécifications et tests

10 Prototyper une fonction

★ 5 min Corrigé p. 155

Lycée Richelieu, Versailles

Dans chacun des cas suivants, prototyper les fonctions demandées.

1 La fonction `convert` convertit une durée exprimée en minutes en une durée exprimée en heures et minutes. Elle affiche le résultat sous la forme « ... h ... min ».

2 La fonction `FarToCel` convertit une température en degré Fahrenheit en degré Celsius. Elle renvoie le résultat sous forme de nombre.

11 Documenter une fonction

★★ 10 min Corrigé p. 155

Lycée Jean Macé, Niort

Voici la définition d'une fonction :

```
def produit_scalaire(u, v):
  """ votre commentaire ...
  ...
  ... """
    return u[0]*v[0] + u[1]*v[1]
```

1 Écrire une docstring pour cette fonction, en détaillant les pré- et post-conditions.

2 Proposer des assertions pour vérifier les préconditions.

COURS
INTERROS
CORRIGÉS

INTERROS

12 Écrire les assertions et le jeu de test ★★

10 min

Corrigé p. 156

Lycée André Theuriet, Civray

La fonction suivante teste la validité d'une adresse mail :

```python
def est_email_valide(email):
    """ Vérifie si l'adresse e-mail fournie est valide.
    Préconditions : email est une chaîne de caractères
    non vide.
    Postcondition : retourne True si l'e-mail est
    valide, False sinon.
    """
    # Ajouter ici les assertions pour vérifier les pré
    conditions
    # ...

    # vérifie la présence d'un seul '@'
    if email.count('@') != 1:
        return False
    # vérifie la présence d'un '.' après le '@'
    partie_domaine = email.split('@')[1]
    if '.' not in partie_domaine:
        return False
    return True
```

1. Écrire les instructions `assert` vérifiant les préconditions.
2. Écrire un jeu de test sous la forme d'une suite d'assertions.

1 QCM Savoir traduire un programme

Énoncé p. 144

1 Réponse $\boxed{\text{c}}$. La variable `degreesfahrenheit` est *a priori* lue, et le programme retourne la variable `degreescelsius`.

2 Réponse $\boxed{\text{d}}$. La boucle « while » montre que les instructions suivantes sont répétées tant que $x \leqslant 10$ et dans la boucle, on voit que x reçoit la valeur $x+1$ à chaque fois ; donc les valeurs prises par x sont : 4, 5, 6, 7, 8, 9 et 10. De plus, dans cette boucle, $y = 3x+5$ donc y représente l'image de x par la fonction $x \mapsto 3x + 5$. Le programme affiche donc bien les images de toutes les valeurs entières de x dans l'intervalle [4 ; 10].

2 V/F Rédaction de programmes

Énoncé p. 145

On peut remarquer que les variables x et y sont déclarées avec le type « double ». Or, y prend la valeur `convert(x)`, qui est une fonction de type « int » ; il y a donc ici une confusion de types.

Pour que le programme fonctionne, il faudrait déclarer la fonction ainsi : « `double convert(x)` ».

3 Programme PHP

Énoncé p. 145

Lycée Camille Claudel, Blois

La 1re ligne du programme affecte une valeur à une variable texte ; d'après le nom donné à la variable, il s'agit du chemin vers une image, donc d'un répertoire dans lequel se trouve une image.

La seconde ligne définit un tableau (« array ») de 4 valeurs.

Ensuite, l'instruction `foreach` ressemble à « for » : il s'agit alors d'une boucle. Son argument (`$arr as $value`) fait référence au tableau précédemment défini. On peut donc traduire cela par le fait que l'on va parcourir ce tableau et que chaque valeur sera désignée par la variable `$value`.

L'instruction `echo` signifie que l'on affiche quelque chose. La suite ressemble à du HTML et la balise `<img src='...'>` nous dit que l'on affiche une image, donc le nom est le répertoire suivi de la valeur sur laquelle on est. Concrètement, le code HTML engendré par le programme PHP est :

```
<img src='/images/4.png'/>
<img src='/images/7.png'/>
<img src='/images/19.png'/>
<img src='/images/23.png'/>
```

Ce programme affiche donc 4 images.

COURS
INTERROS
CORRIGÉS

4 Programme en Java

Énoncé p. 145

Lycée Henri Bergson, Angers

Ce programme est composé de deux parties :

- la première partie est *a priori* la partie principale (le terme « main » confirme que c'est le cas). Sont déclarées dans cette partie une table de valeurs (12, −5, 7, ...) ainsi qu'une variable `elt` de type *long*. La boucle « for » parcourt le tableau de valeurs et teste si `elt` est égale à la valeur sur laquelle on est, auquel cas la boucle « for » est rompue (« break ») et la fonction `afficher` est appelée (fonction qui est hors de la boucle !) ;
- la seconde partie est justement la définition de la fonction `afficher`, qui prend deux arguments : le premier est un entier de type `int`, le second un entier de type `long`. Cette fonction teste si le premier argument (`rang`) vaut 8 (dernier rang de la table), auquel cas cela signifie que l'argument `elt` n'est pas dans la table, et un message est affiché. Sinon, l'argument `elt` est trouvé et un message est affiché.

Finalement, ce programme affiche si un élément informé à l'avance est trouvé ou pas dans la table.

5 Programme en C

Énoncé p. 146

Lycée Sophie Germain, Paris

Ce programme est composé de 3 parties principales (hors en-tête) :

- *la partie main*, qui va exécuter le programme principal. On y déclare un tableau à 3 valeurs (8,4,6). On appelle ensuite les fonctions `tri` et `affiche` ;
- *la fonction `tri`*, qui prend pour arguments un entier n et un tableau de valeurs. Dans cette fonction, on parcourt le tableau et on inverse l'ordre de deux valeurs consécutives si la première est supérieure à la seconde. Ainsi, cette fonction trie dans l'ordre croissant les valeurs du tableau ;
- *la fonction `affiche`*, qui prend les mêmes arguments que la fonction précédente, et dans laquelle on parcourt le tableau pour en afficher les valeurs une à une.

Ce programme affiche donc le tableau de valeurs après avoir trié ces dernières dans l'ordre croissant.

Remarque : la présence d'un astérisque après `int` signifie que la variable `table` « pointe » vers un emplacement en mémoire contenant un entier (`int`) ou un tableau d'entiers. On appelle cela un *pointeur*.

6 Du Javascript au Python

Énoncé p. 147

Lycée Lycée Henri Alain-Fournier, Bourges

Voici le programme Python correspondant au programme écrit en Javascript proposé :

```python
# Fonction pour calculer la moyenne de trois nombres
def calculer_moyenne(a, b, c):
    return (a + b + c) / 3

# Saisie des trois nombres
nombre1 = float(input("Entrez le premier nombre: "))
nombre2 = float(input("Entrez le second nombre: "))
nombre3 = float(input("Entrez le troisième nombre: "))

# Calcul de la moyenne
moyenne = calculer_moyenne(nombre1, nombre2, nombre3)

# Affichage du résultat
print("La moyenne des trois nombres est: " + str(
    moyenne))
```

7 Prototyper une fonction

Énoncé p. 148

Lycée Camille Jullian, Bordeaux

1. La fonction `convert` admet un argument : un entier (la durée en minutes). Le résultat est un texte.
2. La fonction `FarToCel` admet un argument : un nombre réel (un flottant), et renvoie un nombre réel (un flottant).

   En C, le prototype sera :

```c
double FarToCel(double temp);
```

8 Convertisseur de temps

Énoncé p. 148

Lycée Sainte-Anne, Brest

En C

```c
#include <stdio.h>

void convert1(int t){
    int h,m;
    h=t/60;
    m=t-h*60;
    printf("%d min = %d h %d min.\n",t,h,m);
}
```

COURS INTERROS CORRIGÉS

CORRIGÉS

suite du programme en C

```
void convert2(int t){
    double h;
    h=(double)t/60;
    printf("%d min = %.2f h.\n",t,h); // affichage avec
     2 décimales
}

void main()
{
    int t;
    printf("Entrez un temps (en min) :");
    scanf("%d", &t);
    convert1(t);
    convert2(t);
}
```

En Python

```
t = int(input('Entrez le temps (en minutes) : '))

def convert1(t):
    h = int(t/60)
    m = t-h*60
    rep = str(h)+'h'+str(m)+'min.'
    return rep

def convert2(t):
    h = t/60
    rep = str(h)+' heures.'
    return rep

print(t,'minutes =',convert1(t))
print(t,'minutes =',convert2(t))
```

9 Conversions de programmes

Énoncé p. 148

Lycée François Mauriac, Bordeaux

1 **De Fortan à Python**

```
# Programme Python pour demander deux nombres et
    afficher leur somme
a = int(input('Entrez le premier nombre: '))
b = int(input('Entrez le second nombre: '))
somme = a + b
print('La somme est:', somme)
```

2 **De C à Python**

```python
# Programme Python pour demander un nombre et
    afficher si ce nombre est pair ou impair
nombre = int(input("Entrez un nombre: "))
if nombre % 2 == 0:
    print("Le nombre est pair.")
else:
    print("Le nombre est impair.")
```

3 **De PHP à Python**

```python
# Programme Python pour demander le nom de l'
    utilisateur et afficher un message de bienvenue
nom = input("Entrez votre nom: ")
print("Bonjour, " + nom + "!")
```

10 Prototyper une fonction

Énoncé p. 149

Lycée Richelieu, Versailles

1 La fonction `convert` admet un argument : un entier (la durée en minutes). Le résultat est un texte.

2 La fonction `FarToCel` admet un argument : un nombre réel (un flottant), et renvoie un nombre réel (un flottant).

En C, le prototype sera :

```c
double FarToCel(double temp);
```

11 Documenter une fonction

Énoncé p. 149

Lycée Jean Macé, Niort

1 La fonction calcule un produit scalaire (en plus, c'est son nom, cela facilite les choses). Elle admet deux paramètres, u et v. On sait qu'un produit scalaire est une manière de multiplier deux vecteurs entre eux et dont le résultat est un scalaire (c'est-à-dire un nombre). Ici, les vecteurs sont de dimension 2 : $\vec{u} \cdot \vec{v} = u_x \cdot v_x + u_y \cdot v_y$. La ligne 5 utilise clairement des indices (`0` et `1`) pour u et v. On peut donc supposer que u et v sont des *tuples*. Voici donc une proposition de docstring :

```
""" Calcule le produit scalaire de deux vecteurs.
Entrées : u et v sont des tuples de deux nombres,
   par exemple \texttt{(3, -4)}
Sortie : la fonction retourne un nombre (float)
  qui est le produit scalaire uv. """
```

COURS INTERROS CORRIGÉS

CORRIGÉS

2 On vérifie que u et v sont des tuples, qu'ils sont de dimension 2 et que chacun des nombres qu'ils contiennent sont soit de type `float` soit de type `int` :

```
assert type(u) == tuple and type(v) == tuple
assert len(u) == 2 and len(v) == 2
assert isinstance(u[0], (int, float))
assert isinstance(u[1], (int, float))
assert isinstance(v[0], (int, float))
assert isinstance(v[1], (int, float))
```

12 Écrire les assertions et le jeu de test

Énoncé p. 150

Lycée André Theuriet, Civray

1 Les assertions doivent garantir que `email` est de type `str` et que ce n'est pas une chaîne vide :

```
assert type(email) == str, "email doit être une cha
    îne de caractères"
assert email != '', "email ne doit pas être une cha
    îne vide"
```

2 Le jeu de test est une suite d'assertions sur le résultat renvoyé par la fonction. Il faut envisager différents cas d'emails valides ou invalides :

```
assert est_email_valide(1234) == False
assert est_email_valide("") == False
assert est_email_valide("exemple@domaine.fr") ==
    True
assert est_email_valide("exempledomaine.fr") ==
    False
assert est_email_valide("exemple@domainefr") ==
    False
```

Chapitre 5

Interactions humain-machine sur le Web

Plan du chapitre

1. Modalités de l'interaction
2. Interaction client-serveur
3. Formulaire d'une page web

Exercice type

Lycée Sainte-Clotilde, Strasbourg

Retrouvez le corrigé de cet exercice en vidéo.

Cassandre souhaite créer une page web contenant le formulaire suivant :

Entre ton nom : [   ]
Entre ton prénom : [   ]
Entre un mot de passe : [   ]
Tu es : ☐ un garçon ☐ une fille
[Valide le formulaire]

1 Quelle méthode de transmission des données va-t-elle a priori utiliser ?

2 Elle souhaite vérifier si le mot de passe est entré avant de transmettre les informations au serveur.
Quel événement peut-elle utiliser ?

3 Proposer un code HTML (sans le détail de la fonction Javascript). On nommera `traitement.php` le fichier recevant les informations et *formulaire* le formulaire.

Voir corrigé page 172

COURS

1 Modalités de l'interaction

1.1 Définition

Définition 1 : IHM

Les *Interactions Humain-Machine* (IHM) sur le web définissent les moyens et les outils mis en œuvre afin qu'une personne physique puisse agir (contrôler, communiquer) à distance, par l'intermédiaire d'Internet par exemple.

Parmi ces moyens, on trouve le langage HTML (HyperText Markup Language).

1.2 Structure principale d'un fichier HTML

Un document HTML est composé de deux parties séparées par des *balises* :

- son *en-tête*, entre les balises `<head>` et `</head>` ;
- son *corps*, entre les balises `<body>` et `</body>`.

Un document HTML ressemble à ceci :

```
<!DOCTYPE html>
<html>
<head>
...
</head>
<body>
...
</body>
</html>
```

Remarquez que la première ligne indique le langage utilisé : elle est à destination des navigateurs Internet afin qu'ils sachent interpréter correctement le code du document.

Dans l'en-tête d'un document HTML, on retrouve par exemple :

- l'encodage, indiqué comme ceci (par exemple) dans le cas d'un encodage UTF-8 :

```
<meta charset="utf-8">
```

- le titre de la page, indiqué comme ceci :

```
<title>Mon titre</title>
```

- l'éventuel style de la page (définition des couleurs, des polices de caractères, etc.), indiqué par :

```
<link rel="stylesheet" href="style.css">
```

  où `style.css` est un fichier dans lequel est contenu le style de tous les éléments de la page ;
- les éventuelles fonctions qui gèrent certains événements, notamment en Javascript, indiqué ainsi :

```
<script src="script.js"></script>
```

  où `script.js` est le fichier contenant toutes les fonctions en Javascript.

Dans le corps, il est mis tout ce qui doit être proposé aux personnes accédant à cette page : du texte, des images, des liens, etc.

1.3 Les éléments d'une page HTML

1.3.1 - Les textes

Tout document HTML doit être structuré. C'est la raison pour laquelle chaque élément devrait être *balisé*.

Les titres

- Un titre *niveau 1* s'écrit :

```
<h1>Mon titre niveau 1</h1>
```

- Un titre *niveau 2* s'écrit :

```
<h2>Mon titre niveau 2</h2>
```

  Il sera écrit par défaut à l'aide d'une taille de caractères plus petite que le titre de niveau 1.

On peut aller ainsi jusqu'au 6e niveau.

Paragraphes

Chaque paragraphe s'écrit :

```
<p>Mon paragraphe.</p>
```

paragraphe à reformuler :

Accentuation et importance du texte

- Un mot ou groupe de mots « mis en avant » s'écrit :

```
<em>Texte mis en avant.</em>
```

  Par défaut, les navigateurs affichent ce texte en italique, mais cette mise en forme peut être modifiée par une feuille de style (CSS).

COURS

- Un mot ou groupe de mot « important » s'écrit :

  ```
  <strong>Texte important.</strong>
  ```

  Par défaut, les navigateurs affichent ce texte en gras, mais là encore, la mise en forme dépend du CSS.

- Les balises `<i>` et `<b>` permettent respectivement d'afficher un texte en italique et en gras sans signification sémantique. En vertu du principe de séparation entre le contenu (HTML) et la mise en forme (CSS), il est préférable d'utiliser `<em>` et `<strong>` pour donner du sens au contenu, mais `<i>` (texte distinct du reste, par exemple mot en langue étrangère) et `<b>` (mot-clé ou texte mis en évidence) restent utilisables.

1.3.2 - Les images

Les images sont indiquées ainsi :

```
<img src="monimage.jpg" alt="titre">
```

1.3.3 - Les blocs

Les blocs sont très utilisés en HTML. Ils s'écrivent ainsi :

```
<div>
Contenu du bloc
</div>
```

Ils servent à diviser de façon structurée une page web.

On peut imaginer, par exemple, qu'un menu est dans un bloc, que le contenu principal est dans un autre bloc, que le bas de chaque page est dans un autre, etc.

1.3.4 - Les liens

C'est l'une des choses les plus importantes sur le web. Ils se présentent sous cette forme :

```
<a href="cible du lien"> texte ou image </a>
```

1.3.5 - Les cadres

Les cadres sont des moyens d'afficher une autre page ou un autre objet dans la page courante. Elles sont très utilisées pour partager des vidéos par exemple.

```
<iframe src="adresse de ce que nous voulons afficher"></iframe>
```

1.3.6 - Tableaux

Les tableaux sont indiqués comme ceci :

```
<table>
  <thead>
    <tr>
      <td>Titre Colonne 1</td>
      <td>Titre colonne 2</td>
    </tr>
  </thead>
  <tbody>
    <tr>
      <td>Contenu cellule 1, ligne 1</td>
      <td>Contenu cellule 2, ligne 1</td>
    </tr>
    <tr>
      <td>Contenu cellule 1, ligne 2</td>
      <td>Contenu cellule 2, ligne 2</td>
    </tr>
  <tbody>
  <tfoot>
    <tr>
      <td>Fin Colonne 1</td>
      <td>Fin colonne 2</td>
    </tr>
  </tfoot>
</table>
```

Les balises `<thead></thead>`, `<tbody></tbody>` et `<tfoot></tfoot>` ne sont pas obligatoires. Elles indiquent respectivement l'en-tête, le corps et le pied du tableau.

1.3.7 - Les formulaires et les entrées

Les formulaires sont aussi très répandus. Ils se déclarent ainsi :

```
<form>
Contenu du formulaire
</form>
```

Dans les formulaires, on peut voir :

- un champ de saisie pour du texte ;

```
<label for="txt">Mon texte : </label>
<input type="text" name="txt" id="txt">
```

COURS

- un champ de saisie pour les mots de passe :

```
<label for="mdp">Mot de passe : </label>
<input type="password" name="mdp" id="mdp">
```

- un champ de sélection unique :

```
<label for="choix">Mon choix : </label>
<input type="radio" name="choix" id="choix">
```

- une case à cocher :

```
<label for="choix">Mon choix : </label>
<input type="checkbox" name="choix" id="choix">
```

Remarque : la balise `<label>` sert à étiqueter un élément. On peut aussi adopter la syntaxe imbriquée pour ne pas spécifier le nom de l'élément :

```
<label>Mon choix :
<input type="radio" name="choix"></label>
```

- un menu déroulant :

```
<label>
<select name="choix">
    <option>Option 1</option>
    <option>Option 2</option>
    ...
</select>
</label>
```

- un menu à sélections multiples :

```
<label>
<select multiple name="liste[]">
    <option>Option 1</option>
    <option>Option 2</option>
    ...
</select>
</label>
```

- un bouton :

```
<label><input type="button" name="monbouton"></label>
```

1.4 Description d'un élément

Chaque élément présenté précédemment accepte les attributs principaux réunis dans le tableau suivant :

Attribut	Raccourci
Identifiant	id
Nom	name
Classe	class
Titre	title

L'attribut `class` sert à définir le style CSS de l'élément (plusieurs éléments peuvent avoir la même classe), mais il peut aussi servir à déterminer le comportement des éléments ayant cette classe.

ATTENTION

Plusieurs éléments peuvent avoir une même classe ; en revanche, plusieurs éléments ne peuvent pas avoir le même nom ou le même identifiant.

Exemple :

```
<form>
 <label for="pseudo">Identifiant :</label>
 <input type="text" id="pseudo" class="champ">
 <label for="motdepasse">Mot de passe :</label>
 <input type="password" id="motdepasse" class="champ">
 <input type="button" value="Valider" class="bouton">
</form>
```

Dans cet exemple, les deux champs de saisie ont la même classe pour signifier qu'ils doivent avoir le même aspect. Le dernier élément a une classe différente pour traduire le fait que son apparence est différente des deux autres.

Remarque : remarquez que l'élément `label` est lié à son champ de saisie à l'aide de l'identifiant de ce dernier, par son attribut « for ».

À ces attributs s'ajoutent ceux qui sont spécifiques aux images :

Texte alternatif	alt
Hauteur	height
Largeur	width

Exemple :

```
<img src="monimage.jpg" id="cousin-evariste"
alt="Cousin Évariste à la plage" width="600">
```

COURS

Remarques

- Si seule l'une des dimensions est indiquée, l'autre sera calculée proportionnellement à celle informée.
- Les dimensions peuvent être exprimées en pourcentage (contrairement à l'exemple ci-dessus où elles sont par défaut exprimées en pixels).
  Dans ce cas, c'est par rapport à la largeur (`width`) ou la hauteur (`height`) de l'élément contenant l'image (la fenêtre du navigateur, un bloc, une cellule de tableau, etc.).
- Dans une logique de séparation entre contenu et mise en forme, il est conseillé de fixer ces attributs dans la feuille de style plutôt que dans la page HTML.

1.5 Événements

Un lien en HTML peut être utilisé pour effectuer principalement deux types d'actions :

1.5.1 - Clic sur un lien

Il existe deux types d'événements :

- redirection :

  ```
  <a href="autrepage.html">Clique ici</a>
  ```

  Dans ce cas, quand on clique sur le texte « Clique ici », la page *autrepage.html* se charge.
- exécution d'une fonction :

  ```
  <a href="javascript:fonction()">Clique ici</a>
  ```

  Dans ce cas, la fonction javascript `fonction()` s'exécute, cette fonction étant définie dans le document (très souvent dans l'en-tête).

Exemple :

```
<!DOCTYPE html>
<html>
  <head>
    <script type="text/javascript" >
      function popup() { alert('Tu as cliqué !') }
    </script>
  </head>
  <body>
    <a href="javascript:popup()">Clique ici.</a>
  </body>
</html>
```

Dans cet exemple, une fenêtre apparaît quand on clique sur le lien.

1.5.2 - Événements de pointeur

Ces événements sont déclenchés lorsque le pointeur de la souris fait quelque chose sur un élément.
Voici quelques événements principaux :

Événement	Description
onclick	le script se déclenche quand on clique.
ondblclick	le script se déclenche quand on double-clique.
onmouseover	le script se déclenche quand la souris est sur l'élément.

Exemple :

```
<!DOCTYPE html>
<html>
 <head>
  <script type="text/javascript" >
   function popup() { alert('Tu as survolé le texte !') }
  </script>
 </head>
 <body>
  <span onmouseover="javascript:popup()">Clique ici.</span>
 </body>
</html>
```

Ici, la fenêtre s'affiche quand on survole le texte.

1.5.3 - Événements de formulaire

Les événements suivants (les principaux) sont propres aux formulaires.

Événement	Description
onblur	le script se déclenche quand le champ perd le focus.
onchange	le script se déclenche quand on change la valeur du champ.
onfocus	le script se déclenche quand le champ gagne le focus.
onformchange	le script se déclenche quand le formulaire est modifié.
oninput	le script se déclenche quand le champ est informé.
oninvalid	le script se déclenche quand la valeur saisie est invalide.
onselect	le script se déclenche quand on sélectionne un champ dans un menu.
onsubmit	le script se déclenche quand on valide le formulaire.

COURS
INTERROS
CORRIGÉS

COURS

Exemple :

```
<!DOCTYPE html>
<html>
 <head>
  <script type="text/javascript" >
   function controle(val) {
  if (val.length<8)
   alert('Trop petit ce mot de passe !')
   }
  </script>
 </head>
 <body>
  <form>
   <label for='nom'>Nom :</label>
   <input type='text' id='nom'>

   <label for='mdp'>Mot de passe :</label>
   <input type='password' id='mdp' onblur='javascript:
    controle(this.value)'>

   <input type='submit' value='Valider'>
  </form>
 </body>
</html>
```

Dans cet exemple, dès que l'on clique dans le champ correspondant au mot de passe et qu'on le quitte (on perd le focus), la fonction `controle` s'exécute et vérifie si la longueur de la variable est plus petite que 8 caractères ; dans un tel cas, un message apparaît.

Remarque : dans cet exemple, on aurait aussi pu écrire :

```
<form onsubmit="javascript:controle(mdp.value)">
...
<input type="password" id="mdp">
<form>
```

Dans ce cas, la fonction est appelée uniquement quand on appuie sur le bouton de validation, ce qui peut être ici un inconvénient : on a tout intérêt à vérifier la longueur du mot de passe dès qu'il est saisi.

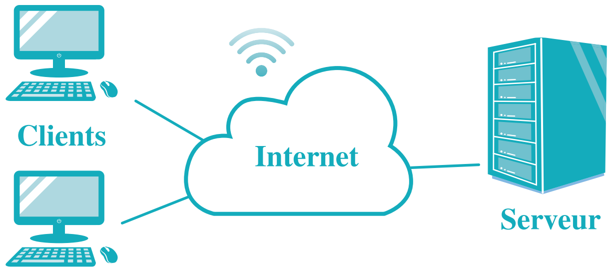
COURS
INTERROS
CORRIGÉS

2 Interaction client-serveur

2.1 L'environnement client-serveur

Définition 2

Une *requête* est un message envoyé par un client à un serveur afin de demander une action.



Dans un environnement client-serveur,

- le *client* est généralement l'ordinateur sur lequel on se connecte à Internet : c'est lui qui envoie les requêtes ;
- le *serveur* est la machine sur laquelle est la page web : c'est elle qui répond aux requêtes.

2.2 Interaction client-serveur

Les interactions exécutées au niveau du client sont en général prises en charge par JavaScript. Dans les exemples précédents, où l'on traitait des événements, tout se passait au niveau du client. Par conséquent, le serveur (où est hébergée la page web) n'a aucune connaissance de l'issue des événements.

Ainsi, lorsqu'une vérification est effectuée directement côté client, la réponse est immédiate, car il n'y a pas de communication avec le serveur.

On comprend alors aisément que plus les traitements sont réalisés côté client, plus ils sont rapides, puisqu'ils ne nécessitent pas d'attendre une réponse du serveur.

En revanche, quand on valide un formulaire par exemple, la requête est envoyée au serveur, qui traite les informations et renvoie une réponse au client. Cette réponse peut contenir une page web mise à jour, des données ou simplement un code de statut HTTP indiquant si l'opération a réussi ou échoué.

COURS

2.3 Les requêtes HTTP

Définition 3 : syntaxe générale

La syntaxe d'une requête est toujours de la forme :

```
Méthode Url Version
En-tête de requête
<saut de ligne>
Corps de requête
```

2.3.1 - La méthode

Il existe plusieurs méthodes pour une requête HTTP. Les plus importantes sont :

- GET : c'est la méthode la plus courante pour demander une ressource au client. Cette requête ne modifie pas la ressource, et il doit être possible de la répéter sans effet. De plus, les paramètres peuvent être passés dans l'URL (ex. : /recherche ?query=nsi).
- POST : cette méthode doit être utilisée lorsqu'une requête modifie la ressource.

2.3.2 - L'en-tête

C'est un peu le même principe que les balises « <meta> » de l'HTML : elles peuvent prendre beaucoup de valeurs. Par exemple :

- HOST : <nom du site>
- Cookie : <nom du cookie>=<valeur du cookie>
- Connection : Close ou Keep-Alive
- Content-type : <type de média du corps de la requête>
- Content-Length : <longueur du corps de la requête>

Remarque : un cookie est un fichier texte qui est sauvegardé côté client et dans lequel sont stockées certaines informations (par exemple, votre identifiant au site afin que vous n'ayez pas à l'écrire à chaque fois).

Pour ce qui est de l'en-tête « Connection » :

- « Close » ferme la connexion après la réponse ;
- « Keep-Alive » conserve la connexion après la réponse.

2.3.3 - Le corps

Il peut contenir par exemple le contenu d'un formulaire HTML.

2.3.4 - Exemples de requêtes

Avec la méthode GET

C'est la méthode permettant de récupérer le contenu d'un fichier.

Exemple :
```
GET /fichier.html HTTP/1.1
Host: www.site.com
Connection: Close
<saut de ligne>
```

Avec la méthode POST

C'est la méthode permettant de transmettre des données à un fichier. Très souvent, le fichier est codé en PHP (encore un autre langage !) afin qu'il puisse traiter les informations reçues.

Exemple :
```
POST /fichier.php HTTP/1.1
Host: www.site.com
Connection: Close
Content-type: application/x-www-form-urlencoded
Content-Length: 22
<saut de ligne>
nom=DUPONT&prenom=Jean
```

2.4 Les réponses

Définition 4 : syntaxe générale

Les réponses du serveur ont la syntaxe générale suivante :
```
Ligne de statut
En-tête de réponse
<saut de ligne>
Corps de réponse
```

2.4.1 - Ligne de statut

La ligne de statut est composée :

- de la version HTTP du serveur ;
- du code d'erreur retourné (par exemple 200, 403, 404, 500) ;
- du texte associé à l'erreur (respectivement pour les exemples précédents : « OK », « FORBIDDEN », « NOT FOUND », « INTERNAL ERROR »).

Exemple : `HTTP/1.1 200 OK`

COURS INTERROS CORRIGÉS

COURS

2.4.2 - L'en-tête

Les en-têtes peuvent contenir ici aussi beaucoup d'informations, comme :

- l'horodatage de génération de la réponse ;
- le modèle du serveur HTTP ;
- le type de média ;
- la longueur du corps ;
- le cookie ;
- etc.

Exemple : `Date: Thu, 15 Nov 2018 09:38:57 GMT`

```
Server: Apache/2.5.12 (Debian GNU/Linux) DAV/2 SVN/1.1.4
Connection: close
Transfer-Encoding: chunked
Content-Type: text/html; charset=UTF-8
```

Dans cet exemple, la dernière ligne nous informe que la réponse est au format HTML et encodée en UTF-8.

Quant à l'avant-dernière, elle nous informe du type d'encodage du transfert (ici, « chunked » signifie que les données sont transférées par blocs, ce qui permet de ne pas connaître à l'avance la taille du corps).

2.5 Transmissions chiffrées

Certaines données sont sensibles (par exemple, le numéro de carte bancaire). Il est donc important, quand on souhaite les transmettre via le protocole HTTP, que ces données soient chiffrées, c'est-à-dire transformées pour ne pas être comprises par d'éventuel.le.s pirates informatiques qui les intercepteraient entre le client et le serveur.

Les pages chiffrées sont reconnaissables en regardant la barre d'adresse ; l'URL commence toujours par :

`https://...`

« https » signifie *HyperText Transfert Protocol Secure* ; c'est la combinaison du HTTP et d'une couche de chiffrement (comme le SSL ou le TLS, deux protocoles de sécurisation des échanges sur Internet).

3 Formulaire d'une page web

Dans ce paragraphe, nous allons voir concrètement comment fonctionnent les formulaires que vous pouvez remplir sur le web, et notamment la façon dont sont transmises les informations.

3.1 Structure générale

Nous allons prendre l'exemple simple d'un formulaire demandant à l'utilisateur ou l'utilisatrice son identifiant et son mot de passe, et proposant un bouton poussoir de validation.

La structure du code HTML est la suivante :

```
<!DOCTYPE html>
<html>
 <head>
  <meta charset="utf-8">
 </head>
 <body> <!-- <br> est pour aller à la ligne -->
  <form name='connexion'>
   <label for="identifiant">Identifiant :</label>
   <input type="text" name="identifiant"><br>
   <label for="motdepasse">Mot de passe :</label>
   <input type="password" name="motdepasse"><br>
   <input type="submit" value="Valider">
  </form>
 </body>
</html>
```

Ce code permet d'afficher le formulaire. Il ne reste plus maintenant qu'à transmettre les informations au serveur. Pour cela, il existe deux méthodes.

3.2 Avec la méthode GET

La première chose à faire est de choisir le fichier destination des données. Nous allons nommer ce fichier `traitement.php` (c'est en effet très souvent un fichier codé en PHP qui traite les informations).

Pour utiliser la méthode GET, nous allons remplacer la ligne :

```
<form name='connexion'>
```

par :

```
<form name='connexion' action='traitement.php' method='get'>
```

Supposons que l'on complète le champ *identifiant* par « toto » et le champ *motdepasse* avec « moi ». En cliquant sur le bouton, le fichier PHP est appelé. Mais si vous essayez, vous verrez que dans la barre d'adresse s'affiche :

```
traitement.php?identifiant=toto&motdepasse=moi
```

La méthode GET fait passer les variables en toute transparence... ce qui est fâcheux dans notre cas.

COURS
INTERROS
CORRIGÉS

COURS

3.3 Avec la méthode POST

Nous allons utiliser maintenant la méthode POST. La syntaxe ne change pas ; seule la méthode change :

```
<form name='connexion' action='traitement.php' method='post'>
```

La différence est toutefois majeure ici car, quand on valide le formulaire, on voit dans la barre d'adresse :

`traitement.php`

Les données n'apparaissent pas, et c'est beaucoup mieux ainsi dans notre situation.

Ainsi, on préfèrera :

- la méthode GET pour des transmissions non sensibles, non confidentielles, où les variables transmises peuvent être visibles côté client ;
- la méthode POST pour des transmissions de données confidentielles, comme des mots de passe, ou des numéros de cartes bancaires.

Remarque : la méthode POST ne suffit pas à sécuriser les transmissions. Il faut pour cela la coupler avec d'autres techniques de sécurisation.

Solution de l'exercice type

Lycée Sainte-Clotilde, Strasbourg

1 Nous voyons qu'il y a un mot de passe à transmettre. Par conséquent, il ne faut pas que les données transmises apparaissent dans la barre d'adresse du navigateur. La méthode GET n'est donc pas recommandée ici.

Il faut donc utiliser la méthode POST pour transmettre les données de ce formulaire au serveur.

2 Elle pourrait utiliser l'événement `onsubmit` quand le bouton de validation est cliqué, événement qui déclenchera l'appel d'une fonction Javascript par exemple, mais le formulaire est validé, quelle que soit l'issue de la fonction Javascript utilisée.

Il est donc préférable d'insérer un bouton poussoir basique qui, lorsque l'on clique dessus, va déclencher l'événement souhaité, à savoir vérifier si le mot de passe est écrit et si tel est le cas, envoyer le formulaire.

Solution de l'exercice type (suite)

Lycée Sainte-Clotilde, Strasbourg

3

```
<!DOCTYPE html>
<html>
 <head>
  <meta charset="utf-8">
   <script type="text/javascript">
    function controle(mot) {...} </script>
  </head>
  <body>
   <form name='formulaire' method="post" action="
    traitement.php">
   <label for="nom">Entre ton nom : </label>
   <input type="text" name="nom"><br>
   <label for="prenom">Entre ton prénom : </label>
   <input type="text" name="prenom"><br>
   <label for="mdp">Mot de passe : </label>
   <input type="password" name="mdp"><br>
   <label for="genre">Tu es :</label>
   <input type="radio" name="genre" value="garçon">
    Un garçon
   <input type="radio" name="fille" value="fille">
    Une fille<br>
   <input type="button" value="Valider" onclick="
    javascript:controle(mdp.value)">
   </form>
  </body>
</html>
```

Voir énoncé page 157

INTERROS

1 QCM Savoir le vocabulaire du cours

5 min Corrigé p. 179

À chaque question ne correspond qu'une seule bonne réponse parmi les trois proposées. Laquelle ?

1. La partie du code HTML qui s'exécute quand on clique sur un lien ou sur un bouton est :
   a un inattendu b un événement c un élément
2. Les délimiteurs HTML, tels que `<head>...</head>`, s'appellent :
   a les balises b les valises c les bornes
3. Une requête est traitée plus rapidement quand elle est du côté :
   a client b serveur c web
4. La méthode qui permet de transmettre les variables en les faisant apparaître dans la barre d'adresse du navigateur est :
   a HEAD b POST c GET

2 V/F Connaître le cours

5 min Corrigé p. 179

Les affirmations suivantes sont-elles vraies ou fausses ?

1. Si l'adresse d'un site Internet commence par « `https://` » alors on peut être assurés que les données transmises sont chiffrées.
2. Les événements peuvent être gérés aussi bien dans l'en-tête que dans le corps d'un fichier HTML.
3. La méthode la plus sûre pour transmettre des données sensibles est la méthode POST.
4. Plusieurs serveurs peuvent se connecter à un client.

3 Interpréter un code

★ 5 min Corrigé p. 179

Lycée André Theuriet, Civray

Voici le code HTML du corps d'un fichier :

```
<h1>L'ingénu</h1>
<p>Un homme <i>ingénu</i> est un homme qui a une sincé
    rité innocente, à la limite de la naïveté.</p>
<p>Cet adjectif vient du latin <b>ingenuus</b>, qui
    peut se traduire par <b>inné</b>.</p>
```

1. De combien de paragraphes cette page est-elle composée ?
2. Que représente la première ligne ?

3 Comment apparaît le mot « ingenuus » sur la page web ? Y a-t-il un autre mot qui apparaît de la même façon ?

4 Comment apparaît le mot « ingénu » sur la page web ?

4 Liens hypertextes

★ 5 min Corrigé p. 180

Lycée Jean-Joseph Fourier, Auxerre

Voici un bout de code HTML :

```
<a href="rene.html"><img src="images/papi-rene.jpg"
alt="Papi René" /></a>
```

1 Qu'affiche ce code ?

2 Où se trouve l'objet affiché par ce code ?

3 Y a-t-il un événement lié à ce code ? Lequel ?

5 Un menu

★ 5 min Corrigé p. 180

Lycée Camille Claudel, Blois

Voici une partie de code HTML :

```
<form method="get" action="redirect.php">
 <label for="page">Page :</label>
 <select name="page">
  <option value="1.html">1.html</option>
  <option value="2.html">2.html</option>
 </select>
 <input type="submit" value="Valider">
</form>
```

1 Expliquer ce qu'il affiche dans une page web.

2 La méthode utilisée vous semble-t-elle judicieuse ?

3 Un visiteur écrit directement dans la barre d'adresse du navigateur :

`redirect.php?2.html`

Est-ce équivalent à l'événement qui est déclenché en sélectionnant la deuxième valeur du menu ?

COURS INTERROS CORRIGÉS

INTERROS

6 Un événement sur formulaire

★ 10 min Corrigé p. 180

Lycée Sophie Germain, Paris

1 Que fait le code HTML suivant ?

```
<!DOCTYPE html>
<html>
 <head>
  <meta charset="utf-8">
   <script type="text/javascript">
    function controle(val) { alert(val) }
   </script>
  </head>
  <body>
  <form>
   <label for="page">Page :</label>
    <select name='page'
  onchange="javascript:controle(this.value)">
      <option value=''></option>
      <option value='1.html'>1.html</option>
      <option value='2.html'>2.html</option>
    </select>
   </form>
   </body>
</html>
```

2 Ce code peut-il fonctionner si la connexion Internet est interrompue ?

7 Mot de passe

★★ 10 min Corrigé p. 181

Lycée polyvalent Émile Zola, Aix-en-Provence

Paul administre un site Internet où les visiteurs peuvent créer un compte en informant un identifiant et un mot de passe.

Il compte créer un formulaire dans lequel il demandera l'identifiant et le mot de passe, informations qu'il enverra par la méthode POST à son serveur qui les enregistrera dans une base de données sans les modifier.

Expliquez pourquoi ce qu'il compte faire n'est pas recommandé.

8 Un champ bien rempli

★★ 15 min Corrigé p. 181

Lycée François Villon, Les Mureaux

Afin d'être sûr que le champ `pseudo` de son formulaire soit bien rempli, Luc utilise le code suivant :

```
<!DOCTYPE html>
<html>
 <head>
  <meta charset="utf-8">
  <script type="text/javascript">
   window.onload = function() {
    document.getElementById('nom').focus(); }
   function controle(val) {
    if (val == '')
     alert('Entrez votre nom.') }
   </script>
  </head>
  <body>
   <form name='formulaire' action='record.php'>
    <label for="nom">Nom :</label>
    <input type="text" id="nom" onblur="controle(this.
    value)">
    <input type="submit" value="Valider">
   </form>
  </body>
</html>
```

Expliquer le fonctionnement de ce code.

9 Formulaire d'inscription

★★★ 30 min Corrigé p. 182

Lycée Raoul Follereau, Nevers

Vous devez créer un formulaire d'inscription pour un site web. Ce formulaire doit permettre aux utilisateurs de s'inscrire en fournissant leur nom, prénom, adresse e-mail et mot de passe. Le formulaire doit également inclure une validation côté client pour s'assurer que tous les champs sont remplis avant l'envoi des données au serveur.

1 Créer le formulaire HTML.

- Le formulaire doit inclure les champs suivants : nom, prénom, adresse e-mail, et mot de passe.
- Utiliser les balises `<label>` pour étiqueter chaque champ.
- Utiliser les balises `<input>` pour les champs de saisie.
- Ajouter un bouton de validation pour soumettre le formulaire.

2. Ajouter la validation côté client.
   - Utiliser JavaScript pour vérifier que tous les champs sont remplis avant l'envoi du formulaire.
   - Afficher un message d'erreur si un champ est vide.
3. Utiliser la méthode POST.
   - Configurer le formulaire pour utiliser la méthode POST pour envoyer les données au serveur.
   - Spécifier l'action du formulaire (par exemple, `action="inscription.php"`).

Questions complémentaires :

1. Pourquoi utilise-t-on la méthode POST pour envoyer les données du formulaire au serveur ?
2. Quelle est l'importance de la validation côté client avant l'envoi des données au serveur ?

10 Calculatrice interactive

★★★ 30 min Corrigé p. 183

Lycée Sainte-Anne, Brest

L'objectif de cet exercice est de créer une mini-calculatrice web capable d'effectuer des opérations simples (addition, soustraction, multiplication, division) entre deux nombres. Les calculs devront se faire *côté client* grâce à JavaScript.

1. Créez un formulaire HTML avec les champs suivants :
   - deux champs pour entrer des nombres ;
   - un menu déroulant pour choisir l'opération à effectuer (`+`, `-`, `*`, `/`) ;
   - un bouton pour effectuer le calcul.
2. Lorsque l'utilisateur clique sur le bouton :
   - une fonction JavaScript est déclenchée ;
   - le résultat est affiché dynamiquement sur la page *sans recharger* celle-ci.
3. Vérifiez côté client que :
   - les deux champs contiennent des nombres ;
   - une erreur s'affiche si un champ est vide ou si une division par zéro est tentée.

Exemple d'affichage attendu :

Champs pour les nombres : `3` et `4`.
Opération sélectionnée : `*`.
Résultat affiché : `12`.

1 QCM Savoir le vocabulaire du cours

Énoncé p. 174

1. Réponse b. Quand on clique sur un lien s'exécute un événement.
2. Réponse a. Les délimiteurs s'appellent des balises (voir page 158).
3. Réponse a. Une requête est plus rapidement traitée du côté client, c'est-à-dire sur votre ordinateur, car il n'y a pas de délai de transmission jusqu'au serveur.
4. Réponse c. La méthode GET permet en effet de transmettre des informations, vers un fichier PHP par exemple, et fait apparaître les données dans la barre d'adresse du navigateur sous la forme :

```
fichier.php?variable1=valeur1&variable2=valeur2
```

2 V/F Connaître le cours

Énoncé p. 174

1. La proposition est *vraie*. Le protocole `https` est la combinaison du protocole `HTTP` (pour transmettre des données) et d'un certificat de sécurité, `SSL` par exemple, qui chiffre les données.
2. La proposition est *vraie*. Les événements peuvent être mis :
   - dans le corps, sous forme de liens hypertextes comme :

   ```
   <a href='monfichier.html'>Mon lien</a>;
   ```

   - dans l'en-tête, sous forme de fonction Javascript, et dans le corps (pour faire appel aux fonctions Javascript créées).
3. La proposition est *vraie*. En effet, avec la méthode POST, les données ne paraissent pas dans la barre d'adresse du navigateur Internet.
4. La proposition est *fausse*. En effet, il n'y a qu'un serveur qui héberge une page donnée disponible sur le web. En revanche, plusieurs clients peuvent se connecter au même serveur (c'est d'ailleurs le principe d'Internet).

3 Interpréter un code

Énoncé p. 174

Lycée André Theuriet, Civray

1. Les paragraphes sont délimités par les balises `<p>` et `</p>`. Il y a donc ici deux paragraphes.
2. La première ligne comporte les balises `<h1>` et `</h1>` ; par conséquent, la première ligne représente un titre (de niveau 1, donc le titre principal).
3. Le mot « ingenuus » est entre les balises `<b>` et `</b>` ; par conséquent, il apparaît en gras sur la page web.

   Il y a aussi le mot « inné » qui est entre les mêmes balises, donc il apparaît aussi en gras.
4. Le mot « ingénu » est entre les balises `<i>` et `</i>` ; par conséquent, il apparaît en italique sur la page web.

CORRIGÉS

4 Liens hypertextes

Énoncé p. 175

Lycée Jean-Joseph Fourier, Auxerre

1 Ce code affiche une image intitulée « papi-rene.jpg ». Si l'image n'est pas disponible alors le texte « Papi René » est affiché.

2 L'objet affiché, donc l'image, se trouve apparemment dans le répertoire nommé « images » (relativement au répertoire dans lequel se trouve le fichier HTML de la page web).

3 L'image se trouve entre les balises `<a href='rene.html'>` et `</a>` donc quand on clique dessus, il y a effectivement un événement : nous sommes redirigés vers la page « rene.html ».

5 Un menu

Énoncé p. 175

Lycée Camille Claudel, Blois

1 Ce code affiche un formulaire dans lequel il y a un unique champ de sélection. Les deux valeurs que peut prendre ce champ sont « 1.html » et « 2.html ». Il y a aussi un bouton poussoir (de validation du formulaire).

2 La méthode utilisée ici pour envoyer les données vers le serveur est la méthode GET. A priori, il n'y a pas de données sensibles : la variable envoyée n'est que le nom d'une page Internet.

La méthode GET est donc judicieuse ici... du moins, elle n'est pas gênante.

3 Quand on choisit la seconde option du menu, à savoir « 2.html », et en validant le formulaire, nous sommes redirigés vers la page `redirect.php`, et la valeur de la variable transmise s'affiche à la suite, séparée par un point-virgule, sous la forme :

```
redirect.php?page=2.html
```

Ainsi, quand on entre directement `redirect.php?2.html`, il manque la variable qui est égale à « 2.html ». Ainsi, cela n'aura pas la même conséquence, et il est fort à parier que cela engendrera une erreur.

6 Un événement sur formulaire

Énoncé p. 176

Lycée Sophie Germain, Paris

1 Ce code est composé :

- d'un en-tête dans lequel il y a une fonction javascript qui affiche la variable mise en argument (`val`) ;
- d'un corps qui affiche un formulaire où il y a un unique champ de sélection. Par défaut, le champ affiché est vide (la première option est vide).

Il n'y a pas de bouton poussoir de validation, mais un événement est prévu quand on change la valeur de la variable `page` : l'appel de la fonction `controle`.

2 Dans la mesure où ce code ne fait appel qu'à un événement javascript, il est exécuté uniquement côté client, donc il peut être exécuté sans connexion internet.

7 Mot de passe

Énoncé p. 176

Lycée polyvalent Émile Zola, Aix-en-Provence

L'envoi des données par la méthode POST est une bonne idée. Cependant, les enregistrer en brut dans la base de données, c'est-à-dire sans modification, est déconseillé car des pirates pourraient avoir accès à cette base de données et ainsi connaître le mot de passe de chaque personne inscrite.

Dans la pratique, le mot de passe est chiffré avant d'être enregistré.

Ensuite, quand la personne veut s'identifier, le mot de passe en brut est envoyé au serveur, puis chiffré de la même façon, et comparé à ce qui est enregistré dans la base de données. Si cela coïncide, cela signifie que les deux mots de passe sont identiques.

8 Un champ bien rempli

Énoncé p. 177

Lycée François Villon, Les Mureaux

1 Dans l'en-tête, outre l'encodage, on voit qu'il y a une partie en javascript.

- `window.onload = function() { document.getElementById('nom').focus(); }` signifie que l'on donne le focus au champ dont l'identifiant est « nom » dès que la fenêtre est lue (« window.onload »). Ainsi, nous sommes assurés que le champ a bien le focus dès le début.
- On définit ensuite une fonction `controle` qui teste si l'argument est vide, auquel cas elle affiche une popup avec un texte nous incitant à remplir le champ.

2 Dans le corps :

- il s'affiche un formulaire avec un champ unique, et un bouton de validation ;
- sur le champ, dont l'identifiant est « nom », il est mis un événement `onblur`, qui déclenche l'appel de la fonction `controle`, définie en en-tête, dès que le champ perd le focus. Cet événement est réalisé même si on ne quitte pas le champ et que l'on clique directement sur le bouton de validation (car dès qu'on clique sur un bouton, les champs perdent le focus).

CORRIGÉS

9 Formulaire d'inscription

Énoncé p. 177

Lycée Raoul Follereau, Nevers

1 La méthode POST est utilisée pour envoyer les données du formulaire au serveur car elle permet de transmettre des données de manière *sécurisée* et *sans les afficher dans l'URL.*

2 La validation côté client est importante pour plusieurs raisons :

- *Rapidité :* elle permet de vérifier les données avant qu'elles ne soient envoyées au serveur, ce qui réduit le temps de traitement et améliore l'expérience utilisateur en fournissant une rétroaction immédiate.
- *Réduction de la charge serveur :* en validant les données côté client, on évite d'envoyer des requêtes inutiles au serveur, ce qui réduit la charge sur celui-ci.
- *Sécurité :* bien que la validation côté client ne soit pas suffisante à elle seule pour garantir la sécurité (car elle peut être contournée), elle constitue une première ligne de défense contre les entrées incorrectes ou malveillantes.

L'en-tête du code HTML peut-être le suivant :

```
<!DOCTYPE html>
<html>
<head>
    <meta charset="utf-8">
    <title>Formulaire d'inscription</title>
    <script type="text/javascript">
        function validerFormulaire() {
            var nom = document.getElementById('nom').
    value;
            var prenom = document.getElementById('
    prenom').value;
            var email = document.getElementById('email
    ').value;
            var motdepasse = document.getElementById('
    motdepasse').value;

            if (nom == "" || prenom == "" || email == "
    " || motdepasse == "") {
                alert("Veuillez remplir tous les champs.
    ");
                return false;
            }
            return true;
        }
    </script>
</head>
```

Le reste du code est le suivant. Vous pouvez télécharger le code entier à l'aide du QR code.

```
<body>
    <form name="formulaire" action="inscription.php"
    method="post" onsubmit="return validerFormulaire()">

        <label for="nom">Nom : </label>
        <input type="text" id="nom" name="nom"><br>
        <label for="prenom">Prénom : </label>
        <input type="text" id="prenom" name="prenom"><
    br>
        <label for="email">Adresse e-mail : </label>
        <input type="email" id="email" name="email"><br
    >
        <label for="motdepasse">Mot de passe : </label>
        <input type="password" id="motdepasse" name="
    motdepasse"><br>
        <input type="submit" value="S’inscrire">
    </form>
</body>
</html>
```

10 Calculatrice interactive

Énoncé p. 178

Lycée Sainte-Anne, Brest

Voici un exemple de code HTML :

```
<!DOCTYPE html>
<html lang="fr">
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width,
    initial-scale=1.0">
    <title>Calculatrice Interactive</title>
    <script>
        function calculer() {
            // Récupération des valeurs des champs
            const nombre1 = parseFloat(document.
    getElementById(’nombre1’).value);
      const nombre2 = parseFloat(document.
    getElementById(’nombre2’).value);
      const operation = document.getElementById(’
    operation’).value;
      const resultatElement = document.getElementById(’
    resultat’);
```

COURS
INTERROS
CORRIGÉS

CORRIGÉS

```
            // Vérification des entrées
            if (isNaN(nombre1) || isNaN(nombre2)) {
                resultatElement.textContent = 'Veuillez
     entrer des nombres valides.';
                return;
            }

            if (operation === '/' && nombre2 === 0) {
                resultatElement.textContent = 'Erreur :
     Division par zéro.';
                return;
            }

            // Calcul
            let resultat;
            switch (operation) {
                case '+':
                    resultat = nombre1 + nombre2;
                    break;
                case '-':
                    resultat = nombre1 - nombre2;
                    break;
                case '*':
                    resultat = nombre1 * nombre2;
                    break;
                case '/':
                    resultat = nombre1 / nombre2;
                    break;
                default:
                    resultatElement.textContent = 'Opé
    ration inconnue.';
                    return;
            }

            // Affichage du résultat
            resultatElement.textContent = `Résultat : $
    {resultat}`;
        }
    </script>
</head>
```

```
<body>
    <h1>Calculatrice Interactive</h1>
    <form onsubmit="event.preventDefault(); calculer();
    ">
        <label for="nombre1">Nombre 1 : </label>
        <input type="text" id="nombre1" name="nombre1">
    <br>

        <label for="nombre2">Nombre 2 : </label>
        <input type="text" id="nombre2" name="nombre2">
    <br>

        <label for="operation">Opération : </label>
        <select id="operation" name="operation">
            <option value="+">Addition (+)</option>
            <option value="-">Soustraction (-)</option>
            <option value="*">Multiplication (*)</
    option>
            <option value="/">Division (/)</option>
        </select><br>

        <button type="submit">Calculer</button>
    </form>
    <p id="resultat"></p>
</body>
</html>
```

Chapitre 6

Architectures matérielles et systèmes d'exploitation

Plan du chapitre

1. Architecture séquentielle
2. Systèmes d'exploitation
3. Réseaux

Exercice type

Sur un ordinateur, ouvrir un terminal.

1 Trouvez l'adresse MAC d'une carte réseau (avec la commande `ipconfig /all` sous Windows ou `ifconfig` sous Linux ou macOS).

2 À l'aide de la commande `ping` suivie de l'adresse Internet du site, trouver l'adresse IP des sites bsky.app et www.google.fr.
Y a-t-il des différences d'affichage ?

3 À l'aide de la commande `traceroute` (Linux/macOS) ou `tracert` (Windows), suivie de l'adresse IP du site, trouver les routeurs rencontrés pour accéder au site bsky.app.

Voir corrigé page 200

1 Architecture séquentielle

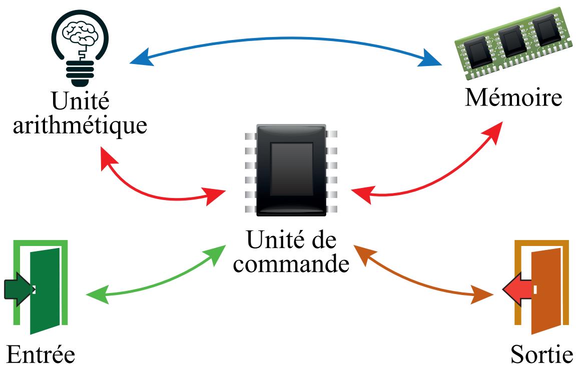
1.1 Schéma de référence

En 1834, Charles Babbage eut l'idée d'incorporer dans la machine à calculer des cartes perforées pour donner la suite d'instructions à exécuter. Il ne put jamais réaliser sa machine la plus performante, faute de fonds suffisants.

COURS

Ada Lovelace, qui travailla avec Babbage, écrivit le premier véritable programme informatique de l'histoire. Son portrait figure d'ailleurs sur les hologrammes d'authentification des produits Microsoft.

Depuis cette époque, le schéma qui sert de référence est celui-ci :



1.2 L'architecture de Von Neumann

L'architecture générale des ordinateurs a été imaginée par John Von Neumann, mathématicien et physicien américano-hongrois, en 1945.

L'architecture de Von Neumann décompose l'ordinateur en 4 parties distinctes :

- l'*unité arithmétique et logique* (UAL ou ALU en anglais) ou *unité de traitement* : son rôle est d'effectuer les opérations de base ;
- l'*unité de contrôle*, chargée de la partie séquentielle des opérations ;
- la *mémoire* qui contient à la fois les données et le programme qui indiquera à l'unité de contrôle quels sont les calculs à faire sur ces données. La mémoire se divise entre mémoire volatile (programmes et données en cours de fonctionnement) et mémoire permanente (programmes et données de base de la machine) ;
- les dispositifs d'entrée-sortie, qui permettent de communiquer avec le monde extérieur.

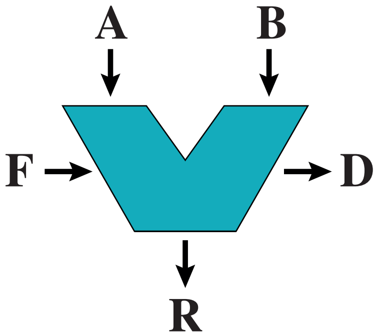
L'UAL et l'unité de commande sont regroupés dans le processeur, qui communique avec la mémoire par un bus. Ces composants constituent l'*unité centrale*, située sur la carte mère d'un ordinateur.

Le tout est rythmé par une horloge interne qui détermine la *fréquence* du processeur.

Exemple : le processeur Intel Core i7-9700 K peut atteindre 4,9 GHz (selon le constructeur), c'est-à-dire 4 900 000 000 cycles d'horloge par seconde.

1.2.1 - UAL

L'unité arithmétique et logique est le cœur de l'ordinateur.
On la représente habituellement par ce schéma :



- A et B sont les opérandes (les entrées) ;
- R est le résultat ;
- F est une fonction binaire (l'opération à effectuer) ;
- D est un drapeau indiquant un résultat secondaire de la fonction (signe, erreur, division par zéro, ...).

Les opérations que peuvent réaliser une UAL de base sont :

- les additions, soustractions, multiplications et divisions ;
- les opérations logiques (ET, OU, NON) ;
- les opérations de comparaison ($>$, $<$ et $=$).

Certaines UAL sont spécialisées (calcul sur les nombres en virgule flottante, cartes graphiques, cartes sons...). Un même processeur peut comporter plusieurs UAL. Dans certaines architectures, les UAL sont spécialisées et ne fonctionnent pas simultanément, dans d'autres les UAL fonctionnent en parallèle (architectures multi-cœur). Les architectures multi-cœur permettent d'augmenter la puissance de calcul.

Les UAL fonctionnent à l'aide de circuits électroniques qui exécutent des fonctions binaires.

1.2.2 - Mémoire

La mémoire est un composant qui peut être vu comme un tableau dans lequel seront stockés aussi bien les variables que les programmes. Chaque case de ce tableau possède une adresse X et la valeur de chaque case est notée M[X]. Les mémoires qui existent sont :

- la mémoire interne vive (RAM) : elle se présente le plus souvent sous forme de barrettes que l'on peut retirer ou ajouter dans un ordinateur ;
- la mémoire interne morte (ROM) : elle sert au démarrage de l'ordinateur ;
- la mémoire externe (de masse) : ce sont les disques durs, les clés USB, etc.

COURS

1.2.3 - L'unité de commande

C'est elle qui donne les ordres à toutes les autres parties de l'ordinateur.

Elle dispose d'emplacements de mémoire interne, de la taille d'un mot (souvent 32 ou 64 bits suivant le processeur), accessibles très rapidement. On distingue principalement quatre registres :

- CO : le compteur ordinal, qui contient l'adresse de la prochaine instruction à exécuter ;
- RI : le registre d'instructions qui contient l'instruction en cours d'exécution ;
- AC : l'accumulateur chargé de stocker des opérandes intermédiaires ;
- CC : le code condition, utilisé pour les instructions de rupture conditionnelle (saut).

Son algorithme de fonctionnement est le suivant :

```
Répéter indéfiniment :
   RI ← M[CO]
   CO ← CO + 1
   R.OP(RI.AD)
Fin du Répéter
```

Autrement dit :

- l'instruction désignée par le compteur ordinal M[CO] est chargée dans le registre d'instructions RI ;
- le compteur ordinal CO est incrémenté pour désigner l'instruction suivante ;
- l'opération chargée dans RI se décompose en deux parties : la fonction proprement dite désignée par OP, et l'opérande sur lequel cette fonction est exécutée, désignée par AD. L'opération désignée par OP est exécutée avec l'opérande AD.
  En cas de rupture conditionnelle, le contenu du compteur ordinal CO est remplacé par l'adresse du saut.

1.3 Aspect séquentiel des opérations

Le langage de base d'une machine est son langage machine, lequel est ensuite traduit en assembleur pour le rendre lisible par l'humain. Il existe presque autant de langages assembleur que de processeurs : un programme écrit en assembleur ne sera compris que par le processeur destinataire contrairement à un programme écrit dans un langage comme Python, qui sera compris par plusieurs machines.

En assembleur, dans sa forme la plus simple, une instruction est stockée dans une cellule mémoire désignée par une adresse. Elle est constituée de 2 parties :

- un code opération (CO) qui indique quelle opération doit être effectuée ;
- une adresse (ADR) désignant l'opérande de cette opération (son argument).

Exemple : un algorithme de calcul d'une addition décrit par l'expression arithmétique $z = x + y$ donnera naissance à une suite de trois instructions telles que :

- LD X : chargement (load) du contenu de la cellule mémoire X dans une cellule appelée accumulateur (c'est un des *registres*). On note ce registre A ici ;
- ADD Y : addition du contenu de la cellule mémoire Y avec le contenu de l'accumulateur ;
- STO Z : stockage (STO pour *store*) du contenu de l'accumulateur dans la cellule mémoire Z.

LD, ADD, STO sont des codes opération, et X, Y, Z les adresses des opérandes.

Toute expression arithmétique ou logique va être décomposée en une suite d'instructions de cette nature. L'unité de commande va alors se charger d'enchaîner séquentiellement la suite de ces instructions calculant cette expression.

Remarque : ce seul mécanisme ne suffit pas pour exécuter des instructions venant de langages de haut niveau (comme Python). Il faut introduire d'autres instructions dites *ruptures de séquence*.

2 Systèmes d'exploitation

2.1 Fonction d'un système d'exploitation

Programmer en assembleur n'est pas chose aisée ; pourtant, c'est le langage que comprend la machine. Il est donc nécessaire d'avoir un intermédiaire entre l'humain et la machine.

C'est le rôle principal d'un système d'exploitation (OS en anglais, pour *Operating System*) : il va « traduire » ce que nous voulons faire en langage machine afin que l'ordinateur comprenne et exécute l'action.

Le système d'exploitation va donc :

- fournir une interface entre l'humain et la machine ;
- gérer les ressources de l'ordinateur, à savoir sa mémoire, son processeur, ses périphériques (écrans, imprimantes, etc.), la gestion d'énergie... ;
- gérer les utilisateurs ainsi que leurs droits d'accès à certains fichiers ;
- être indépendant du matériel (un même OS peut être exécuté sur des ordinateurs n'ayant pas les mêmes composants) ;
- faire en sorte de rendre concret ce qui ne l'est pas (un fichier est par essence abstrait mais nous le considérons bel et bien comme une entité concrète).

COURS

2.2 Ligne de commande

Chaque système d'exploitation possède un « terminal » : c'est un sous-programme qui permet d'exécuter des tâches en écrivant les commandes (et non en cliquant sur des icônes).

- Sous Windows 11, on peut lancer le terminal en faisant :
  [Touche Windows]+[R], puis en entrant « cmd » et en validant.
  Une alternative plus moderne sous Windows est PowerShell, qui permet des commandes plus avancées. Pour le lancer :
  [Touche Windows]+[R], puis en entrant « powershell » et en validant.
- Sous Ubuntu (distribution Linux), il est accessible en cliquant sur l'icône adéquat (représentant un écran d'ordinateur noir).
- Sous macOS, dans le dock, cliquer sur l'icône Finder, puis cliquer sur *Applications*, puis *Utilitaires*, et enfin *Terminal* (représenté par un icône d'écran d'ordinateur noir).

Remarque : on trouvera les mêmes commandes sous Linux et macOS... Ce n'est pas un hasard car Unix est à l'origine de macOS et de Linux.

Quelques commandes de base communes aux OS que l'on peut utiliser dans un terminal sont :

Commande	Description
ls	Liste les fichiers et répertoires du répertoire courant
cd	Change de répertoire (Change Directory)
rm	Supprime un fichier ou un répertoire (ReMove)
<commande> –help	Affiche les options de la commande

Exemples

- Sous Windows, `cd/` mène à la racine du disque dur.
- `rm toto.txt` supprime le fichier « toto.txt ».

Certaines commandes ne sont pas identiques selon que l'on est sous Windows ou Linux/macOS :

Windows	Linux/macOS	Description
copy	cp	Copie un fichier à un autre endroit ou sous un autre nom
move	mv	Déplace un fichier ailleurs
ren	mv	Renomme un fichier

Exemples

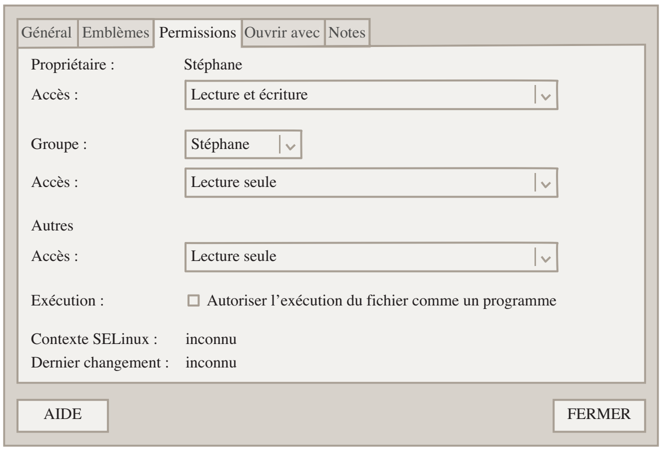
- Sous Windows, `copy toto.txt fichiers/temp.txt` copie le fichier « toto.txt » se trouvant dans le répertoire courant dans le répertoire *fichiers/* sous le nom de « temp.txt ».
- Sous Ubuntu, `mv toto.txt fichiers` déplace le fichier « toto.txt » dans le répertoire *fichiers/*.

2.3 Droits et permissions

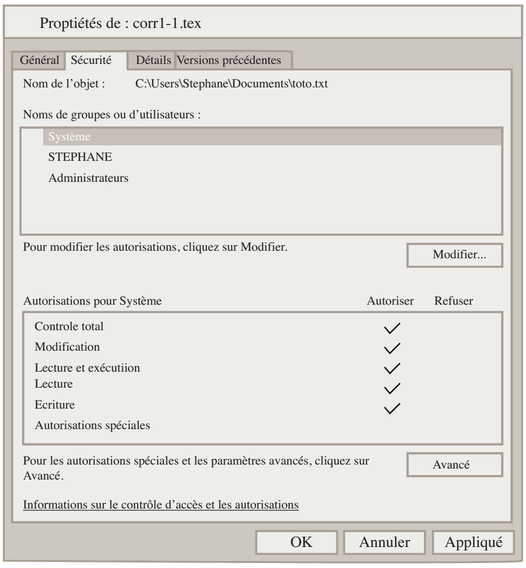
Par défaut, quel que soit le système d'exploitation utilisé, chaque fichier et répertoire a un droit d'accès et des permissions (d'être lu, modifié ou exécuté).

Ces droits sont visibles en cliquant avec le bouton droit de la souris sur le fichier ou répertoire afin d'afficher un menu contextuel, et en sélectionnant *Propriétés*.

Sous Ubuntu, on a la fenêtre :



Sous Windows, on a la fenêtre :



Vous pouvez ainsi, grâce à l'interface graphique, changer facilement les droits d'accès à un fichier, ainsi que les permissions (droits d'écriture, de lecture, lecture et exécution, etc.).

3 Réseaux

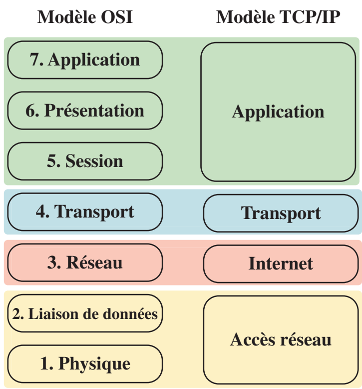
3.1 Principe de base

Quand un ordinateur souhaite transmettre des données à un autre, il suit les étapes suivantes :

- les données sont transmises au système d'exploitation à l'aide d'une application ;
- celui-ci les coupe en plusieurs paquets, les numérote, et les transmet à la carte réseau ;
- celle-ci les envoie sur le réseau ;
- le carte réseau de l'ordinateur destinataire reçoit les paquets transmis, et les envoie au système d'exploitation ;
- l'OS reconstitue le message et le transmet à l'application de destination.

C'est ce que l'on appelle le *modèle en couches*.

Le modèle OSI (*Open Systems Interconnection*) contient 7 couches et le TCP/IP en contient 4 :

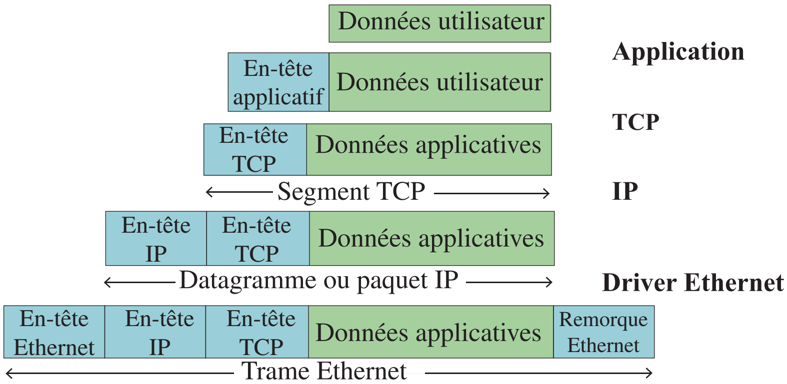


3.2 Encapsulation des données

Lorsqu'elles passent par une couche, les données sont « complétées » par un en-tête spécifique à la couche traversée. C'est ce que l'on appelle l'*encapsulation*.

Exemple : en suivant le protocole TCP/IP, lorsqu'un paquet passe par la carte réseau par exemple, l'en-tête adjointe contient l'adresse MAC (*Medium Access Control*), une suite de 6 nombres exprimés en hexadécimal, comme 4C-BB-58-BC-45-49.

À la réception des paquets, les en-têtes sont enlevés : on appelle cela la *décapsulation*.



3.3 Protocole du bit alterné

Retrouvez ce protocole en vidéo.

3.3.1 - Principe

Le protocole du bit alterné (*alternating bit protocol*) fait partie de la couche liaison (seconde couche du modèle OSI et couche accès réseau du modèle TCP/IP). Il permet de transférer des données entre deux entités, de façon à ce qu'il n'y ait pas d'erreur dans la transmission.

Appelons (T) la première entité, celle qui transmet les données à la seconde entité, que l'on va noter (R) comme *réceptrice*.

Dans l'idéal, c'est-à-dire quand tout se passe bien, le protocole du bit alterné fait que (T) envoie un message à (R) et à la réception de ce message, (R) renvoie un *acquittement* à (T) (*acknowledgment*). Mais il se peut que la liaison entre (T) et (R) ne soit pas fiable. Il est donc possible que certains messages ou certains acquittements se perdent.

Afin de détecter s'il y a des pertes, le protocole stipule que :

- chaque message et chaque acquittement correspondant contiennent un même bit de contrôle. Par exemple, le premier message contient le bit de contrôle « 0 », et son acquittement contient aussi le bit de contrôle « 0 » ;
- deux messages successifs contiennent des bits différents (la valeur du bit de contrôle alterne à chaque émission).

Si, après l'émission d'un message par l'entité (T) (par exemple avec le bit de contrôle « 0 »), aucun acquittement correspondant (noté *ACK0*) n'est reçu au bout d'une certaine durée, ou si un acquittement avec un bit de contrôle erroné (noté *ACK1*), elle émet à nouveau le dernier message envoyé, avec le même bit de contrôle (« 0 »).

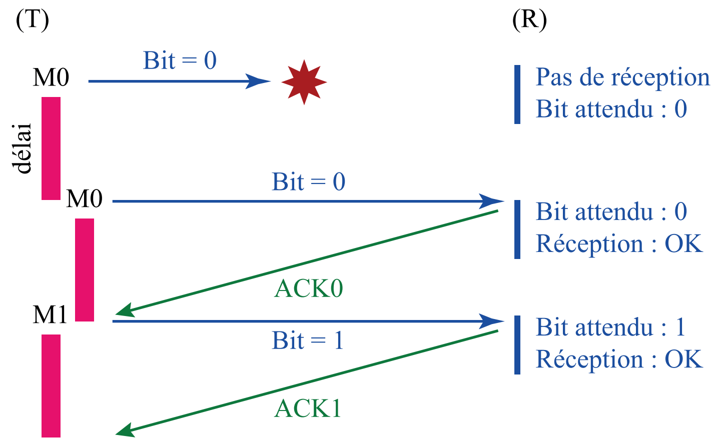
Si l'entité (R) reçoit un message avec un bit de contrôle différent de celui attendu, elle émet à nouveau le dernier acquittement envoyé.

COURS

3.3.2 - Non réception du message

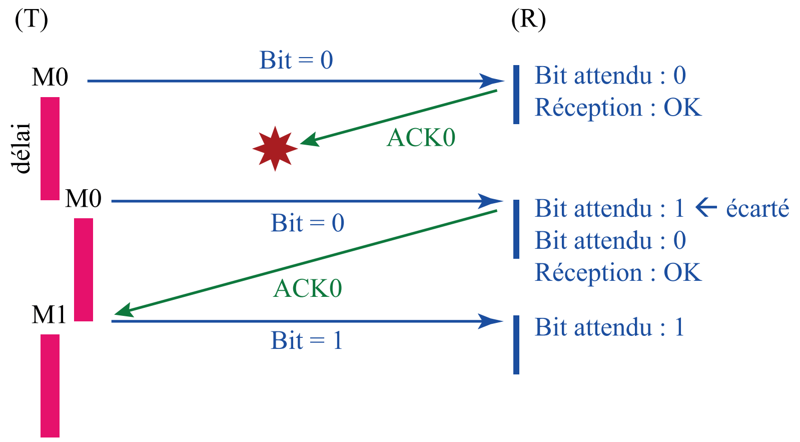
Un message est découpé en plusieurs blocs : M_0, M_1, etc. Par défaut, on attribue le bit de contrôle « 0 » à M_0.

M_0 est donc transmis de (T) à (R) avec le bit de contrôle « 0 ». Si le transfert n'est pas correctement fait, (R) ne renvoie pas d'acquittement. Au bout d'une durée prédéfinie, sans réception d'acquittement, (T) transfère à nouveau M_0.



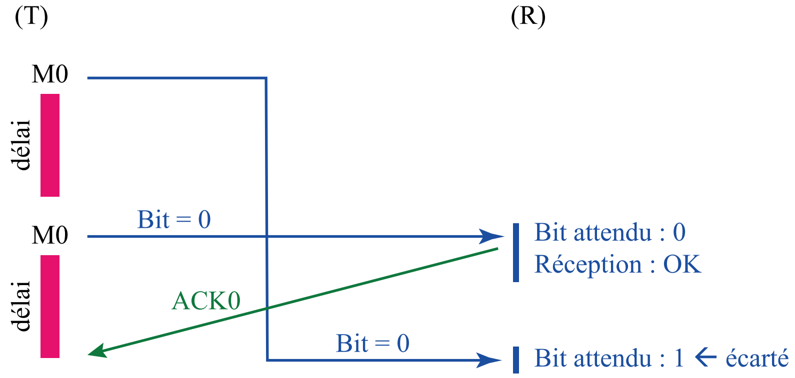
3.3.3 - Non réception de l'acquittement

Il se peut que l'acquittement lui-même ne soit pas délivré correctement ou ne soit pas délivré dans le délai imparti. Dans ce cas, le message est renvoyé.



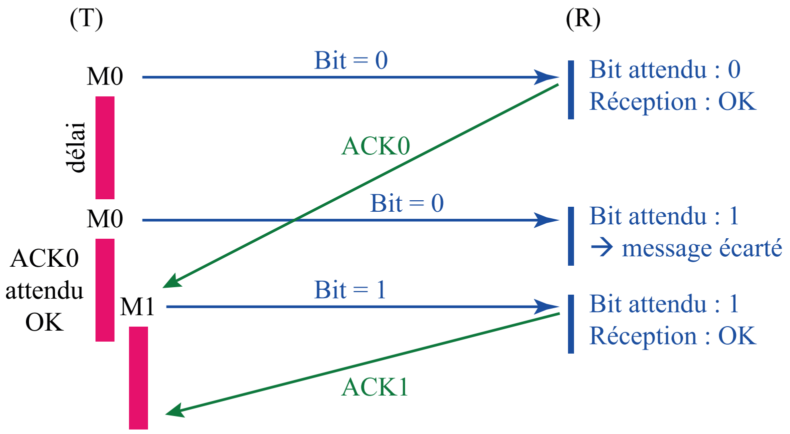
3.3.4 - Chevauchement de messages

Dans le cas où un message met trop longtemps à parvenir à (R), il est transmis à nouveau après que le délai d'attente soit passé. Dans ce cas, le message reçu après est écarté.



3.3.5 - Chevauchement d'acquittements

Dans le cas où un acquittement met trop longtemps à parvenir à (T), le message est transmis à nouveau après que le délai d'attente soit passé. Dans ce cas, le message dont l'acquittement est reçu après est écarté.



Remarque : il existe bien entendu d'autres possibilités, mais ce protocole n'étant plus utilisé dans la pratique, nous ne nous attarderons pas sur tous les différents cas possibles.

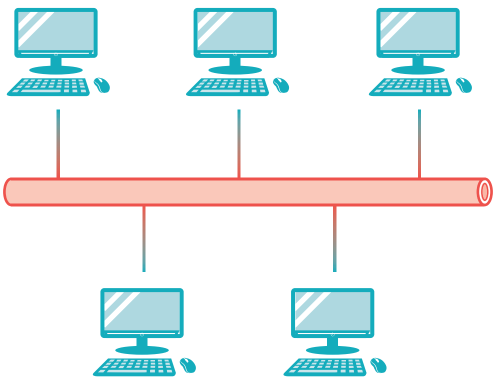
3.4 Architecture d'un réseau

Il existe 4 catégories de réseaux :

- le réseau personnel (PAN : Personal Area Network), qui relie des machines sur quelques mètres ;
- le réseau local (LAN : Local Area Network), adapté à la taille d'un site d'entreprise ;
- le réseau *métropolitain* (MAN : Metropolitan Area Network), étendu à l'échelle d'une ville ;
- le réseau *étendu* (WAN : Wide Area Network), pour une grande zone géographique, comme un pays ou un continent.

Nous ne nous intéresserons ici qu'aux réseaux de type LAN.
Il existe 3 topologies pour un tel réseau.

3.4.1 - Topologie en bus

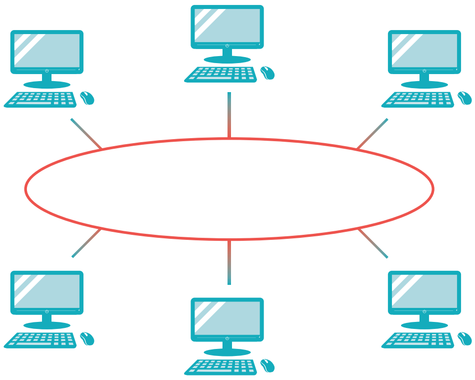


Le *bus* est ici un câble auquel sont reliées les machines.

L'un des avantages du bus réside dans la simplicité de sa mise en œuvre et son moindre coût.

Parmi les inconvénients, on peut citer qu'en cas de rupture du bus, le réseau devient inutilisable. De plus, le signal n'est jamais régénéré, ce qui limite la longueur des câbles.

3.4.2 - Topologie en anneau

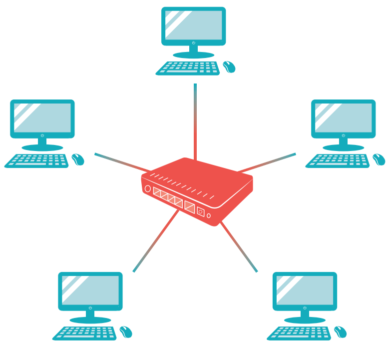


Parmi les avantages, cette topologie permet d'avoir un débit proche de 90 % de la bande passante. De plus, le signal qui circule est régénéré par chaque station.

Dans les inconvénients, on peut citer le fait que si un poste tombe en panne, le réseau ne fonctionne plus. De plus, plus le nombre de machines augmente, plus la transmission d'une information est longue.

En réalité, les ordinateurs d'un réseau en anneau ne sont pas reliés en boucle, mais sont reliés à un répartiteur (appelé MAU, *Multistation Access Unit*) qui va gérer la communication entre les ordinateurs qui lui sont reliés en impartissant à chacun d'entre eux un temps de parole.

3.4.3 - Topologie en étoile



Cette topologie est la plus courante.

Toutes les stations sont reliées à un unique composant central : le concentrateur (hub) ou le commutateur (switch). Le concentrateur rediffuse le message d'une station à toutes les autres stations qui lui sont connectées alors que le commutateur envoie le message uniquement à la station destinataire.

COURS

Ce type de réseau est facile à mettre en place et à surveiller. La panne d'une station ne met pas en cause l'ensemble du réseau. En revanche, il faut plus de câbles que pour les autres topologies, ce qui engendre un coût plus élevé que les autres topologies, et si le concentrateur tombe en panne, tout le réseau est hors d'état de fonctionner. De plus, le débit pratique est moins bon que pour les autres topologies.

Solution de l'exercice type

1. Avec la commande `ipconfig /all` ou `ifconfig`, on obtient une suite de paragraphes. Dans chacun d'eux concernant une carte réseau figure une « adresse physique » : c'est l'adresse MAC de la carte, qui est de la forme `45:7a:df:8b:9c:78`.

2. À l'aide de la commande `ping bsky.app`, on obtient que l'adresse IP de ce site est `18.224.121.90`.

   À l'aide de la commande `ping www.google.fr`, on obtient que l'adresse IP de ce site est `2a00:1450:400c:c0a::5e`.

   On constate qu'il y a une différence d'affichage : la première IP est une IPv4, la seconde une IPv6.

3. À l'aide de la commande `traceroute 18.224.121.90` ou `tracert 18.224.121.90`, on peut voir que l'on passe (si on est chez nous) par notre box Internet, un nœud à proximité de chez nous (par lequel passe la connexion de vos voisins s'ils ont le même fournisseur d'accès à Internet), éventuellement d'autres routeurs de votre ville et d'ailleurs... jusqu'à arriver finalement au serveur web de BlueSky.

Voir énoncé page 187

1 QCM Connaître le cours

5 min

Corrigé p. 206

Pour chaque question, 4 propositions de réponses sont faites. Une seule est correcte. Laquelle ?

1 Un processeur peut atteindre 3,5 GHz. Cela correspond à combien de cycles d'horloge par seconde ?

- a 3 500 000 000
- b 3 500 000
- c 3 500 000 000 000
- d 3 500

2 Dans l'architecture de Von Neumann, l'UAL est généralement représentée par :

- a un « N »
- b un « V »
- c un « C »
- d un « U »

3 Une UAL ne peut pas :

- a faire des additions
- b calculer des puissances entières de nombres entiers
- c faire des racines carrées
- d faire des opérations logiques

4 Dans le protocole TCP/IP, il y a :

- a 3 couches
- b 4 couches
- c 5 couches
- d 7 couches

2 V/F Connaître le cours

5 min

Corrigé p. 206

Indiquer si les propositions suivantes sont vraies ou fausses.

1 Dans l'architecture de Von Neumann, l'unité de commande donne les ordres aux autres parties.

2 Une carte réseau possède une adresse IP.

3 Le réseau d'un établissement scolaire comme un lycée est un réseau PAN.

4 La topologie en étoile est la topologie réseau la plus répandue.

3 IPv4 et IPv6

★ 5 min Corrigé p. 206

Lycée François Villon, Les Mureaux

1. Quel est le format d'une adresse IPv4 ?
   Sur combien de bits est codée une adresse IPv4 ?
2. En déduire le nombre total d'adresses IPv4 disponibles en théorie.
   Est-ce suffisant à l'heure actuelle ?
3. Mêmes questions avec IPv6.

4 La commande chmod sous Linux

★★ 10 min Corrigé p. 207

Retrouvez plus d'explications sur la commande chmod sous Linux en vidéo.

Sous Linux, il existe trois groupes d'utilisateurs :

- propriétaire ;
- groupe du propriétaire ;
- autre.

Pour chaque fichier, il y a trois types de droits :

- *r* : droit de lecture (« r » pour *read*) ;
- *w* : droit d'écriture (« w » pour *write*) ;
- *x* : droit d'exécution (« x » pour... la consonance !).

À chaque fichier, on attribue un triplet de 3 bits représentant le droit de chaque groupe. Par exemple, « 111 101 100 » signifie que :

- le propriétaire a tous les droits (111 = 7 en décimal) ;
- le groupe a le droit de lecture et d'exécution (101 = 5 en décimal) ;
- les autres ont le seul droit de lecture (100 = 4 en décimal).

Pour donner ces droits, on peut utiliser la commande *chmod* ainsi :

```
chmod 754 monfichier.sh.
```

1. Quelle ligne de commande un propriétaire souhaitant donner tous les droits à tout le monde doit-il écrire ?
2. Que doit-il écrire pour se donner tous les droits sans en donner un seul aux autres ?
3. Il écrit : `chmod 123 fichier.ext` ; quels droits a-t-il donné ?

5 Ordinateur virtuel et mémoire

★★ 20 min Corrigé p. 207

Lycée François Mauriac, Bordeaux

On imagine un ordinateur ultra simplifié d'une mémoire centrale constituée d'un tableau de 32 cellules, chacune d'elles étant constituée d'un octet. Chaque instruction est constituée d'un code opération sur 3 bits et d'une partie adresse sur 5 bits.

On dispose du tableau d'instructions suivant :

Instruction	Effet
LD X	Lit le contenu de la cellule X (ACC ← M[X])
STO X	Enregistre dans la cellule X (M(X] ← ACC)
ADD X	Ajoute le contenu de la cellule X (ACC ← ACC+M[X])
SUB X	Soustrait le contenu de la cellule X
JMP ADR	Saute à l'adresse ADR (CO ← A)
JMPZ ADR	Saute à l'adresse ADR si nul
JMPP ADR	Saute à l'adresse ADR si positif
JMPN ADR	Saute à l'adresse ADR si négatif

On considèrera qu'un programme commence à l'adresse 8, les adresses de 0 à 7 étant réservées au stockage des données.

1 Que fait le programme suivant :

Adresse	Contenu
0	48
1	37
...	...
8	LD 0
9	SUB 1
10	STO 2
11	END

2 On suppose qu'une valeur est dans la case mémoire 0, et une autre dans la case mémoire 1.

Écrire un programme qui lit celle de la case mémoire 1, qui la compare à celle de la case mémoire 0 et si elle est supérieure ou égale, l'écrit dans la case mémoire 2.

3 Écrire un programme qui lit une valeur x dans la case mémoire 0, et qui calcule le plus grand exposant n tel que $2^n < x$ pour le stocker en case mémoire 2.

INTERROS

6 Adressage IP et sous-Réseaux

★★ 20 min Corrigé p. 209

Lycée Richelieu, Versailles

Un réseau est caractérisé par son *masque de sous-réseau* (*subnet mask*) est une adresse dont la représentation binaire est de la forme `1...1 0...0`, c'est-à-dire une suite de `1` ininterrompue suivie d'une suite de `0` ininterrompue. Ce masque permet de séparer une adresse IP en deux parties (*composante réseau* et *composante hôtes*) et de retrouver, dans une adresse IP, la partie désignant le réseau lui-même (les bits du masque égaux à `1`) et celle désignant les hôtes (les bits du masque égaux à `0`).

1. Représenter en binaire l'adresse 255.255.0.0 et vérifier que c'est un masque de sous-réseau valide.
2. Quel est le nombre de machines adressables au total dans ce sous-réseau ?
3. Mêmes questions pour le masque 255.255.248.0.

7 Architecture Von Neumann

★★★ 30 min Corrigé p. 210

Lycée François Mauriac, Bordeaux

L'objectif de cet exercice est d'appliquer vos connaissances sur l'architecture Von Neumann et les interactions avec les systèmes d'exploitation en analysant et en simulant des opérations simples.

- *Initialisation de la mémoire :* vous disposez d'une mémoire centrale organisée en 16 cellules, où chaque cellule peut contenir une instruction ou une donnée. Le contenu initial de la mémoire est le suivant :

```
Adresse | Contenu
--------|--------
0       | 10 (X)
1       | 20 (Y)
2       | 0  (Résultat Z)
3       | LD 0
4       | ADD 1
5       | STO 2
6       | END
```

- *Instructions disponibles :*
  - `LD X` : charge la valeur de l'adresse `X` dans l'accumulateur.
  - `ADD X` : ajoute la valeur de l'adresse `X` à l'accumulateur.
  - `SUB X` : soustrait la valeur de l'adresse `X` à l'accumulateur.
  - `STO X` : stocke la valeur de l'accumulateur à l'adresse `X`.
  - `END` : arrête l'exécution.

1. Décrivez pas à pas l'exécution de ce programme en précisant le contenu de l'accumulateur et les modifications apportées à la mémoire après chaque instruction.
2. Proposez une modification de ce programme pour calculer `Z = X - Y` au lieu de `Z = X + Y`.
3. Imaginez que ce programme soit exécuté sur une architecture multi-cœur. Expliquez comment cette architecture pourrait améliorer les performances si plusieurs programmes similaires doivent être exécutés en parallèle.
4. Implémentez une version en pseudo-assembleur pour calculer `Z = X × Y`, en supposant que l'instruction `MUL X` soit disponible.

CORRIGÉS

1 QCM Connaître le cours

Énoncé p. 201

1. Réponse [a]. Une fréquence de 3,5 GHz correspond à $3{,}5 \times 10^9$ cycles d'horloge, soit 3 500 000 000.
2. Réponse [b]. La représentation conventionnelle d'une UAL est un « V ».
3. Réponse [c]. Une UAL ne peut pas faire des racines carrées car elle ne peut faire que des additions, soustractions, multiplications (donc les calculs de puissances entières sont possibles) et divisions.
4. Réponse [b]. Dans le protocole TCP/IP, il y a 4 couches : « Accès réseau », « Internet », « Transport » et « Application ».

2 V/F Connaître le cours

Énoncé p. 201

1. La proposition est *vraie*. Des 4 parties, l'unité de commande agit comme le cerveau de la machine : c'est elle qui décide ce qui doit être fait. Elle donne donc les ordres.
2. La proposition est *fausse*. Une carte réseau possède une adresse MAC, mais pas une adresse IP.
3. La proposition est *fausse*. Un réseau PAN est un réseau que l'on peut retrouver chez soi par exemple, mais pas dans une structure à la taille d'un lycée. Il s'agit plutôt d'un réseau LAN.
4. La proposition est *vraie*. En effet, la topologie en étoile est facile à mettre en place et un dysfonctionnement d'un poste n'entraîne pas le plantage du réseau tout entier, ce qui constitue un avantage majeur de cette topologie.

3 IPv4 et IPv6

Énoncé p. 202

Lycée François Villon, Les Mureaux

1. Une adresse IPv4 est composée de 4 octets, et représentée par 4 nombres décimaux de 0 à 255 séparés par un point (format décimal pointé). Chaque octet correspondant à 8 bits, une adresse IPv4 est codée sur 32 bits.

   Ainsi, une adresse IPv4 comme 192.168.1.1 se représenterait ainsi :

   11000000.10101000.00000001.00000001

2. Il y a 256 possibilités pour chaque octet, soit un total de

$$256^4 = 2^{32} = 4\ 294\ 967\ 296$$

adresses IPv4 possibles.

De plus, il faut retirer les adresses réservées, ainsi que les adresses de réseau et les masques, ce qui fait encore moins. On constate donc que quatre milliards d'adresses sont insuffisantes pour adresser de manière unique toutes les machines sur internet.

3 Une adresse IPv6 est composée de 16 octets, soit 128 bits. Il y a 2 possibilités pour chaque bit (0 ou 1). Il y a donc $2^{128} \approx 3{,}4 \times 10^{38}$ IPv6 possibles, ce qui semble très largement suffisant.

4 La commande chmod sous Linux

Énoncé p. 202

1 Si tous les droits sont à donner à tout le monde, cela se traduit par le triplet binaire 111 111 111, soit 777 en décimal.

Le propriétaire du fichier devra alors taper :

```
chmod 777 fichier.ext
```

2 Si lui seul a tous les droits, cela se traduit par le triplet binaire 111 000 000, soit 700 en décimal. Il devra donc taper :

```
chmod 700 fichier.ext
```

3 Le nombre décimal 123 correspond au triplet binaire 001 010 011. Il s'est donc donné le seul droit d'exécution, puis a donné le droit d'écriture à son groupe, et enfin a donné les droits d'exécution et d'écriture à tous les autres... ce qui n'est pas très malin !

Remarque : pour modifier de façon plus rapide les droits, on peut utiliser la manière symbolique : `chmod [ugoa]±[rwx]`, le premier argument étant une des lettres entre les crochets.

- « u » pour *user* : le propriétaire ;
- « g » pour *group* : le groupe ;
- « o » pour *others* : les autres ;
- « a » pour *all* : tout le monde.

Ainsi, en écrivant `chmod u+w fichier.ext`, il se redonne le droit d'écriture. En écrivant `chmod o-w, o-x, o+r`, il enlève les droits d'exécution et d'écriture, et ajoute les droits de lecture.

5 Ordinateur virtuel et mémoire

Énoncé p. 203

Lycée François Mauriac, Bordeaux

1 Le programme commence par lire la donnée dans la cellule 0 (donc 48), puis soustrait le contenu de la cellule 1 ($48 - 37 = 11$), et enfin stocke le résultat (11) dans la cellule 2.

CORRIGÉS

2 On a le programme suivant :

Adresse	Contenu	Interprétation
0	x	
1	y	
⋮	⋮	
8	LD 1	Lit y
9	SUB 0	Calcule $y - x$
10	JMPP 12	Si $y - x > 0$, va en 13
11	JMPZ 12	Si $y - x = 0$, va en 13
12	JMPN 14	Si $y - x < 0$, va en 14
13	STO 2	Stocke le résultat en 2
14	END	Fin du programme

3 Le programme est le suivant :

Adresse	Contenu	Interprétation
0	x	
1	1	v : valeur courante de 2^n (initialement $2^0 = 1$)
2	0	exposant n (initialement 0)
3	1	valeur d'incrément pour n
⋮	⋮	
8	LD 1	Lit $v = 2^n$
9	SUB 0	Calcule $2^n - x$
10	JMPP 19	Si $2^n - x > 0$, va en 19
11	JMPZ 19	Si $2^n - x = 0$, va en 19
12	LD 2	Lit n
13	ADD 3	Incrémente n
14	STO 2	Stocke la nouvelle valeur de n
15	LD 1	Lit $v = 2^n$ courant
16	ADD 1	Multiplie par 2
17	STO 1	Stocke nouvelle valeur de 2^n
18	JMP 9	Retour en 9 pour nouvelle itération
19	LD 2	Lit n
20	SUB 3	Décrémente n
21	STO 2	Stocke la plus grande valeur n telle que $2^n < x$
22	END	Fin du programme

6 Adressage IP et sous-Réseaux

Énoncé p. 204

Lycée Richelieu, Versailles

1 Pour représenter l'adresse 255.255.0.0 en binaire, nous devons convertir chaque octet en binaire :

- 255 en binaire est 11111111 ;
- 0 en binaire est 00000000.

Donc, 255.255.0.0 en binaire est :

11111111.11111111.00000000.00000000

Pour être un masque de sous-réseau valide :

- tous les bits de la partie « 1 » (bits de réseau) doivent être en continu, sans interruption ;
- tous les bits de la partie « 0 » (bits d'hôtes) doivent être également en continu.

L'adresse binaire 11111111.11111111.00000000.00000000 respecte bien cette règle, car les 16 premiers bits sont des 1 (partie réseau), suivis de 16 bits à 0 (partie hôtes).

2 Dans un masque de sous-réseau, les bits à 1 représentent la partie réseau, et les bits à 0 représentent la partie hôtes.

Dans 255.255.0.0, les 16 premiers bits sont à 1 (partie réseau). Les 16 derniers bits sont à 0 (partie hôtes).
Ainsi, il y a 16 bits pour les hôtes.

Le nombre total d'adresses possibles est donné par 2^n, où n est le nombre de bits pour les hôtes :

$$2^{16} = 65\,536 \text{ adresses possibles.}$$

Dans chaque sous-réseau, deux adresses sont réservées :

- l'adresse réseau (où tous les bits d'hôtes sont à 0) ;
- l'adresse de *broadcast* (où tous les bits d'hôtes sont à 1).

Le nombre d'adresses utilisables pour les machines est donc :

$$65\,536 - 2 = 65\,534 \text{ machines adressables.}$$

3 En binaire, 255.255.248.0 est :

11111111 11111111 11111000 00000000.

Ce réseau peut comporter $2^{11} - 2 = 2\,046$ hôtes.

CORRIGÉS

7 Architecture Von Neumann

Énoncé p. 204

Lycée François Mauriac, Bordeaux

1

- Instruction `LD 0` : charge `10` dans l'accumulateur.
- Instruction `ADD 1` : ajoute `20` à l'accumulateur, qui contient maintenant `30`.
- Instruction `STO 2` : stocke `30` à l'adresse `2`.
- Instruction `END` : arrête le programme.

Nous avons donc l'état final de la mémoire suivant :

État final de la mémoire :

```
Adresse | Contenu
--------|--------
0       | 10
1       | 20
2       | 30
3       | LD 0
4       | ADD 1
5       | STO 2
6       | END
```

2 Voici une modification du programme précédent pour calculer `Z = X - Y` au lieu de `Z = X + Y` :

```
3       | LD 0
4       | SUB 1
5       | STO 2
6       | END
```

3 Dans une architecture multi-cœur, plusieurs instances de ce programme pourraient être exécutées en parallèle, chaque cœur traitant une instance distincte. Cela permettrait de traiter simultanément plusieurs calculs indépendants, réduisant ainsi le temps total d'exécution.

4 Voici une version en pseudo-assembleur pour calculer `Z = X × Y`, en supposant que l'instruction `MUL X` soit disponible :

```
3       | LD 0
4       | MUL 1
5       | STO 2
6       | END
```

Chapitre 7

Algorithmique

Plan du chapitre

1. Introduction
2. Parcours séquentiel d'un tableau
3. Tris
4. Algorithme des k plus proches voisins
5. Recherche dichotomique dans un tableau trié
6. Algorithmes gloutons

Exercice type — **Lycée Camille Claudel, Blois**

Étant donnée une série de n valeurs x_1, x_2, ..., x_n, on appelle *variance* le nombre :

$$V = \frac{1}{n}\sum_{i=1}^{n}\left(\overline{x} - x_i\right)^2, \quad \text{où } \overline{x} \text{ représente la moyenne des } n \text{ valeurs.}$$

1 En supposant la moyenne connue, écrire un algorithme permettant de calculer la variance de n valeurs.

2 Montrer que cet algorithme est de coût linéaire.

3 Dans quel cas cet algorithme peut-il être de coût quadratique ?

Voir corrigé page 225

Nous avons vu au chapitre 3 qu'il existait beaucoup de langages de programmation, chacun ayant ses spécificités, mais tous ayant quelques points en commun.

Le point commun le plus important est la façon dont les programmes sont construits : leurs algorithmes, qui constituent la base de tout programme.

1 Introduction

1.1 Définition

Définitions 1

Un *algorithme* est une succession d'instructions permettant d'aboutir à un résultat souhaité.

L'*algorithmique* est l'étude et la conception de règles permettant d'écrire des algorithmes.

COURS

Autrement dit, l'algorithmique va nous permettre d'écrire un algorithme qui, une fois traduit en un langage, va donner un programme.

1.2 Historique

Le mot « algorithme » vient du nom du mathématicien perse *Al-Khwârizmî*.

On retrouve les premières traces d'algorithmes dans la civilisation babylonienne, où des méthodes de calculs et de résolutions d'équations sont décrites. L'un des algorithmes les plus connus est celui d'Euclide, permettant de calculer le plus grand commun diviseur de deux nombres entiers, et présent dans le livre 7 des *Éléments*, recueil mathématique de 13 livres écrit par Euclide lui-même. Fait remarquable : cet algorithme contient une itération (c'est-à-dire que l'on y voit la répétition d'un même processus) et la démonstration du fait que l'algorithme fait ce qu'on lui demande (on dit dans ce cas que l'algorithme est *correct*).

Un second algorithme connu par les mathématiciens est celui d'Archimède pour calculer le nombre π.

De nos jours, il existe plusieurs types d'algorithmiques. Citons par exemple :

- l'algorithmique itérative ;
- l'algorithmique récursive ;
- l'algorithmique parallèle.

Selon l'algorithmique choisie, l'algorithme n'aura pas la même structure.

2 Parcours séquentiel d'un tableau

2.1 Recherche d'une occurrence

Dans ce paragraphe, on s'intéresse à la recherche d'une occurrence sur des valeurs de type quelconque : nombres, chaînes de caractères ou tableaux.

Nous allons noter `b` l'élément recherché et `a[0]`, `a[1]`, ..., `a[n-1]` les n éléments parmi lesquels `b` est recherché.

Pour cela, une simple boucle itérative bornée suffit :

```
p ← Faux (p est un booléen)
Pour i allant de 0 à n-1:
   Si b = a[i] alors:
     p ← Vrai
   Fin du Si
Fin du Pour
```

On voit ici qu'il est nécessaire d'effectuer $n+1$ affectations : une avant la boucle, et n dans la boucle. Ceci n'est pas la méthode la plus efficace car on effectue n itérations même si l'élément recherché est le premier du tableau.

2.2 Recherche d'un extremum

Dans ce paragraphe, on recherche les valeurs minimale et maximale sur une liste de n nombres `a[0]`, `a[1]`, ..., `a[n-1]`.
Là encore, une simple boucle itérative bornée suffit :

```
Pour i allant de 0 à n-1:
   Si i=0 alors:
     min ← a[0]
     max ← a[0]
   Fin du Si
   Si a[i]<min alors:
     min ← a[i]
     Sinon:
       Si a[i]>max alors:
         max ← a[i]
       Fin du Si
   Fin du Si
Fin du Pour
```

On constate ici qu'il y a :

- 2 affectations (`min ← a[0]` et `max ← a[0]`);
- au maximum $n-1$ affectations pour `min ← a[i]` ou `max ← a[i]` (les deux affectations ne peuvent pas survenir en même temps).

Il y a donc au maximum $n+2$ affectations.

2.3 Calcul d'une moyenne

On souhaite ici calculer la moyenne des valeurs `a[0]`, `a[1]`, ..., `a[n-1]`.
Encore une fois, une simple boucle itérative bornée suffit :

```
m ← 0
Pour i allant de 0 à n-1:
   m ← m + a[i]
Fin du Pour
m ← m/n
```

Ici, `m` est la variable dans laquelle est stockée la moyenne à la fin de la boucle. En effet, on sait que la moyenne de n nombres se calcule ainsi :

$$m = \frac{a_0 + a_1 + \cdots + a_{n-1}}{n} = \frac{a_0}{n} + \frac{a_1}{n} + \cdots + \frac{a_{n-1}}{n}.$$

On constate alors qu'il y a ici :

- $n+2$ affectations (`m ← 0`, n affectations dans la boucle et `m ← m/n`);
- n opérations arithmétiques (une addition par itération).

Il y a ainsi $2n+2$ affectations et opérations arithmétiques.

COURS
INTERROS
CORRIGÉS

COURS

2.4 Le coût d'un algorithme

2.4.1 - Définition

Définition 2

Le *coût d'un algorithme* est le nombre d'opérations élémentaires (arithmétiques ou logiques) ainsi que d'affectations nécessaires à son exécution.

Ainsi,

- pour la recherche d'une occurrence, le coût de l'algorithme est égal à $n+1$;
- pour la recherche d'un extremum, il est égal à $n+2$;
- pour le calcul de la moyenne, il est égal à $2n+2$.

À RETENIR

Dans la pratique, le coût est souvent une approximation.

Ainsi, on dira que le coût de l'algorithme du calcul de la moyenne de n nombres est de l'ordre de $2n$.

Concrètement, dans un algorithme comportant une seule boucle, le coût est généralement proportionnel au nombre d'itérations effectuées. En revanche, dans le cas de boucles imbriquées, le coût dépend du produit du nombre d'itérations de chaque boucle.

2.4.2 - Coût linéaire et coût quadratique

Définitions 3

On dira qu'un algorithme a un *coût linéaire* si son approximation est de la forme kn, où $k > 0$.
On dira qu'un algorithme a un *coût quadratique* si son approximation est de la forme kn^2, où $k > 0$.

Exemples

- Les algorithmes précédents ont des coûts linéaires.
- Si le coût calculé précisément est égal à $2n^2+n+2$ alors son approximation est $2n^2$ et il est alors quadratique.

3 Tris

3.1 Tris par insertion

3.1.1 - Principe

Définition 4

L'algorithme de *tri par insertion* est l'algorithme de tri « naturel ».

Exemple : soit le tableau `t = [8,5,1,9,7]` à trier.

- 1re itération : `[8,5,1,9,7]` devient `[5,8,1,9,7]` ;
- 2e itération : `[5,8,1,9,7]` devient `[1,5,8,9,7]` ;
- 3e itération : `[1,5,8,9,7]` devient `[1,5,7,8,9]`.

L'algorithme est le suivant :

```
t est un tableau de n valeurs
Pour i allant de 1 à n-1:                          (1)
   x ← t[i]                                        (2)
   j ← i                                           (3)
   Tant que j > 0 et t[j-1] > x:                   (4)
     t[j] ← t[j - 1]                               (5)
     j ← j - 1                                     (6)
   Fin du Tant que
   t[j] ← x                                        (7)
Fin du Pour
```

Explications :

- la boucle itérative (ligne 1) parcourt le tableau `t` ;
- ensuite, on affecte à la variable `x` la valeur sur laquelle on est (ligne 2) ;
- on introduit ensuite (ligne 3) une variable auxiliaire `j` qui prend la valeur du rang auquel on est (`i`) ;
- on créé une boucle conditionnelle « Tant que » pour exécuter des instructions quand les valeurs précédentes sont inférieures à celle sur laquelle on est (celle de rang `i`) (ligne 4) ;
- dans ce dernier cas, la valeur de rang `j` devient celle de rang `j-1` (ligne 5) et on passe au rang précédent (ligne 6) ;
- la boucle conditionnelle étant terminée, la valeur qui était auparavant au rang `i` est mise au rang `j` : le premier rang nul ou pour lequel `t[j-1]` $\leqslant$ `x` (ligne 7).

Regardons ce qui se passe pour la première itération (`i=1`) sur l'exemple précédent :

- `x ← 5` car `t[1]=5` (ligne 2) et `j ← 1` (ligne 3) ;
- `j>0` et `8>5` donc on entre dans la boucle conditionnelle (ligne 4) ;
- `t[1] ← 8` (ligne 5) et `j ← 0` (ligne 6) ;
- la condition `j>0` n'est plus remplie (ligne 4) donc la boucle conditionnelle est terminée. Alors, `t[0] ← 5` (ligne 7).

3.1.2 - Terminaison de l'algorithme

La boucle itérative (ligne 1) ne pose aucun problème. En revanche, une boucle conditionnelle (ligne 4) est dangereuse car il se peut qu'elle ne se termine jamais.

Dans notre algorithme, la condition de la boucle est de la forme « c_1 et c_2 », où c_1 et c_2 sont deux conditions. La boucle s'arrête quand « non(c_1 et c_2) » est vérifiée, c'est-à-dire quand « non(c_1) ou non(c_2) » est vérifiée. Il faut donc qu'au moins l'une des conditions contraires soit vérifiée.

Dans notre algorithme, `j` est décrémentée à chaque fois, donc nous sommes assurés que la condition `j>0` ne sera plus remplie à une certaine étape.

Ceci nous assure que l'algorithme se termine.

3.1.3 - Correction de l'algorithme

Posons $P(i)$ la propriété :

$P(i)$: « le tableau [t[0], t[1],...,t[i-1]] est trié. »

On peut montrer que pour tout `i` variant de 1 à `n-1`, donc à chaque itération, $P(i)$ est vraie.

Comme $P(n-1)$ signifie que le tableau entier est trié, l'algorithme est correct.

MÉTHODE : invariant de boucle

La propriété $P(i)$ est appelée un *invariant de boucle* : c'est une propriété qui est vraie avant et après chaque itération. S'il existe un invariant de boucle alors l'algorithme est correct.

3.1.4 - Coût de l'algorithme

Le pire des cas que l'on peut rencontrer est un tableau où toutes les valeurs sont dans l'ordre décroissant (comme `[9,8,7,5]` par exemple).

Dans ce cas, pour chaque itération, il y a 2 affectations (lignes 2 et 3), puis $2n$ affectations (lignes 5 et 6) et enfin 1 affectation (ligne 7), soit $2n+3$ affectations.

Il y a n itérations, donc au final, il y a $n(2n+3)$ affectations, ce qui entraîne un coût approximatif de $2n^2$.

COURS
INTERROS
CORRIGÉS

Propriété 1

Le coût de l'algorithme de tri par insertion est quadratique.

3.2 Tri par sélection

3.2.1 - Principe

Définition 5

L'algorithme de *tri par sélection* consiste à :

- rechercher le plus petit élément d'un tableau de valeurs et l'échanger avec l'élément d'index 0 ;
- rechercher le second plus petit élément du même tableau et à l'échanger avec l'élément d'index 1 ;
- de continuer ainsi avec les autres éléments.

Exemple : reprenons le tableau de l'exemple 3.1.1 : `t = [8,5,1,9,7]`.

- `t = [8,5,1,9,7]` devient `t = [1,5,8,9,7]` ;
- `t = [1,5,8,9,7]` devient `t = [1,5,8,9,7]` (le « 5 » ne bouge pas) ;
- `t = [1,5,8,9,7]` devient `t = [1,5,7,9,8]` ;
- `t = [1,5,7,9,8]` devient `t = [1,5,7,8,9]`.

L'algorithme est le suivant :

```
Pour i allant de 0 à n - 2:
   min ← i
   Pour j allant de i+1 à n-1:
     Si t[j] < t[min]:
       min ← j
     Fin du Si
   Fin du Pour
   Si min ≠ i:
     x ← t[min]
     t[min] ← t[i]
     t[i] ← x
   Fin du Si
Fin du Pour
```

L'algorithme ne comporte que des boucles bornées donc il se termine nécessairement.

COURS

3.2.2 - Correction de l'algorithme

Posons $P(i)$ la propriété :

$P(i)$: « À la fin de la boucle Pour sur `i`, le tableau obtenu est un tableau ne contenant que des valeurs du tableau initial, et les i premiers éléments de ce tableau coïncident avec ceux du tableau final. »

C'est un invariant de boucle, donc l'algorithme est correct.

3.2.3 - Coût de l'algorithme

Dans le « pire » des cas, pour chaque itération de `i`, il y a :

- 1 affectation (ligne 2) ;
- $n - i - 1$ affectations (ligne 5) car il y a $n - i + 1$ valeurs de `j` (ligne 3) ;
- 3 affectations (lignes 8, 9 et 10).

Le nombre total d'affectations est donc :

$$\begin{aligned}\sum_{i=0}^{n-2}(3+n-i) &= \sum_{i=0}^{n-2} 3 + \sum_{i=0}^{n-2}(n-i)\\ &= 3(n-1) + \frac{n(n+1)}{2} - 1\\ &= \frac{1}{2}n^2 + \frac{7}{2}n - 4.\end{aligned}$$

Propriété 2

Le coût de l'algorithme de tri par sélection est quadratique.

4 Algorithme des k plus proches voisins

4.1 Première approche

Définition 6

Étant donnés n nombres $x_0, x_1, ..., x_{n-1}$ décrivant un ensemble X, l'algorithme des k plus proches voisins consiste à trouver les k valeurs de X les plus proches d'un nombre x donné.

Pour cela,

- on prend les k premières valeurs de X : $x_0, ..., x_{k-1}$, et on les met dans un ensemble Y ;
- on note y la valeur de Y la plus éloignée de x ;
- pour chaque valeur restante de X, si elle est plus proche de x que y alors on remplace y par cette valeur.

On obtient :

```
/* on met les k premiers éléments de X dans Y */
Pour i allant de 0 à k-1:
   y[i] ← x[i]
Fin du Pour

/* on parcourt les éléments restants de X */
Pour i allant de k à n-1:
   Pour j allant de 0 à k-1:                            (1)
     Si j=0:
       d ← |x-y[0]|
       r ← 0
     Sinon:
       Si |x-y[j]| > d:
         d ← |x-y[j]|
         r ← j
       Fin du Si
     Fin du Si
   Fin du Pour                                          (2)
/* à ce stade, "d" représente la plus grande distance et
"r" l'index du  terme le plus éloigné de x */
   Si |x-x[i]| < d:                                     (3)
     y[r] ← x[i]                                        (4)
   Fin du Si
Fin du Pour
```

Pour chaque itération de k à $n-1$,

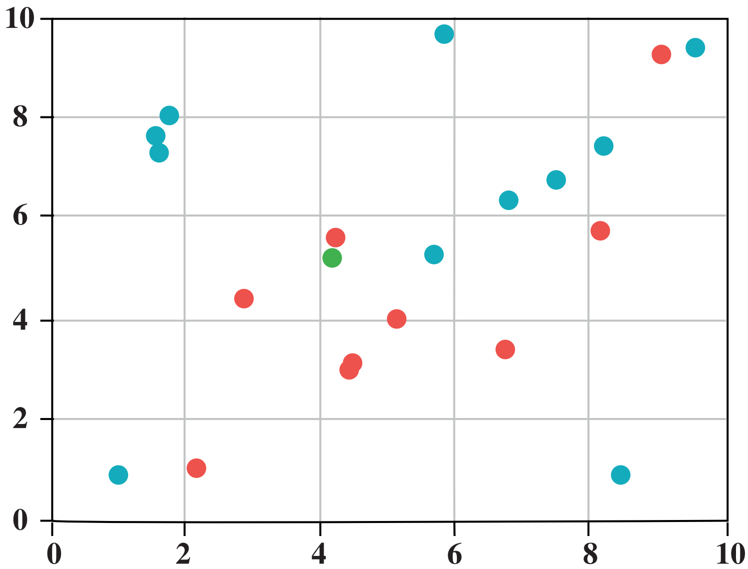
- entre les lignes (1) et (2), on détermine la distance la plus grande entre x et toutes les valeurs de Y ainsi que le rang du terme (`r`) le plus éloigné de x ;
- ensuite, on teste si la valeur de rang `i` est plus proche de x que le terme le plus éloigné de Y (ligne 3). Si tel est le cas, on remplace cette valeur (la plus éloignée de x) par `x[i]` (ligne (4)).

4.2 Généralités

L'algorithme des k plus proches voisins est en réalité un peu plus complexe que ce que nous venons de voir. En effet, dans la pratique, il permet de « prédire » la nature d'un élément (on parle ici de *classe* de l'élément) informé à partir d'éléments déjà enregistrés.

COURS

Prenons un exemple visuel : dans un repère, on considère une série de n points de coordonnées $(x_i\ ;\ y_i)$, i variant de 0 à $n-1$. Chaque point est de couleur bleue ou rouge. On met ensuite un point au hasard (en vert) :



Ici, les éléments sont les points et leur classe est leur couleur.

L'objectif est de prédire la classe de l'élément rajouté selon nos critères. On va prendre par exemple $k = 5$ et on va regarder les 5 plus proches voisins du point vert.

On constate alors que parmi les 5 points les plus proches du point vert, 4 sont rouges et 1 est bleu. On prédit alors que la classe du point ajouté sera « rouge ».

5 Recherche dichotomique dans un tableau trié

5.1 Première approche

Soit `t` un tableau de taille n dont les éléments sont triés. On souhaite retrouver un élément `e` parmi les valeurs de ce tableau.

Bien sûr, on pourrait parcourir le tableau en partant du début jusqu'à trouver l'élément, comme dans l'algorithme page suivante.

```
i ← 0
bingo ← Faux
Tant que bingo ≠ Vrai ou i < n:
   Si t[i] = e:
     bingo ← Vrai
   Sinon:
     i ← i+1
   Fin du Si
Fin du Tant que
```

Le coût de cet algorithme est linéaire dans le pire des cas (si `e` se trouve à la fin du tableau). Mais on peut tout de même faire mieux en accélérant la recherche.

5.2 Dichotomie

Le principe est le suivant :

1. on pose `min=0]` et `max=n-1` ;
2. on prend le milieu `t[m]` de l'intervalle `[min ; max]` ;
3. si `t[m]=e` alors on a fini ;
4. si `t[m] > e` alors on pose `max=m`, sinon on pose `min=m` et on retourne au point 2.

Dans la pratique, il faudra faire attention au fait que `m` n'est pas toujours un entier ; il faudra donc faire en sorte qu'il le soit.
Avant de voir l'algorithme en lui-même, regardons sur un exemple son fonctionnement.

Exemple : soit `t = [1,2,5,7,8,9,12,15,16,18,19,23,25,26,27,30]`.
Alors `min=0, max=15`. Prenons `e = 9`.

- `0 < 15` donc on pose $m = E\left(\dfrac{0+15}{2}\right) = 7$.

  `t[7]=15 > 9` donc `max = m = 7`, en gardant `min = 0`.
- `0 < 7` donc on recommence : $m = E\left(\dfrac{0+7}{2}\right) = 3$.

  `t[3]=7 < 9` donc `min = m = 3`, en gardant `max = 7`.
- `3 < 7` donc on recommence : $m = E\left(\dfrac{3+7}{2}\right) = 5$.

  `t[5]=9`. On a trouvé ! On arrête donc la boucle.

L'algorithme itératif est alors le suivant :

```
min ← 0
max ← n-1
Tant que min < max:
   m ← E((min + max)/2)
   Si t[m] < e:
     min ← m+1
   Sinon:
     max ← m
   Fin du Si
Fin du Tant que
Si t[min] = e:
   Afficher min
Sinon:
   Afficher : "Non trouvé"
```

Nous parlons ici d'*algorithme itératif* car vous verrez l'année prochaine qu'un *algorithme récursif* existe.

Remarque : cet algorithme n'est bien entendu pas unique. Il existe d'autres syntaxes, mais le fond reste le même.

5.3 Terminaison de l'algorithme

MÉTHODE : variant de boucle

Pour démontrer qu'une boucle conditionnelle de type « Tant que » se termine, on considère un entier qui décroît à chaque itération.
Cette quantité est appelée *variant de boucle*.

Dans notre algorithme, `min` et `max` sont deux entiers naturels, par conséquent la variable `d = max - min` est aussi un entier naturel. Elle n'est pas présente dans notre algorithme, mais nous pourrions la définir.

À chaque itération, `d` décroît car soit `min` augmente, soit `max` diminue. Ainsi, il existe une itération pour laquelle `d` ne sera pas strictement positive : la boucle « Tant que » s'arrêtera alors, ce qui prouve la terminaison de l'algorithme.

COURS INTERROS CORRIGÉS

6 Algorithmes gloutons

6.1 Principe général

Les algorithmes gloutons existent pour résoudre des *problèmes d'optimisation*. Ils correspondent à une solution optimale obtenue en effectuant une suite de meilleurs choix pour chaque étape de l'algorithme (à chaque étape, on fait le meilleur choix possible).

De plus, il n'y a pas de retour en arrière : quand un choix est fait à une étape, il n'est pas modifié ultérieurement et il ne modifie pas les choix précédents.

L'autre caractéristique des algorithmes gloutons est que lorsqu'un choix est fait, on tente de résoudre un problème plus petit.

On appelle cela la *progression descendante*.

6.2 Le problème du rendu de monnaie

Nous allons illustrer ce qu'est un algorithme glouton à travers l'un des plus célèbres problèmes : « étant donné un système de monnaie (pièces et billets), comment rendre une somme donnée de façon optimale, c'est-à-dire avec le nombre minimal de pièces et billets ? »

Plaçons-nous dans notre système de monnaie ; nous disposons alors de la monnaie suivante (toutes les valeurs sont exprimées en euros) :

- *pièces* : 0,01 – 0,02 – 0,05 – 0,10 – 0,20 – 0,50 – 1 – 2 ;
- *billets* : 5 – 10 – 20 – 50 – 100 – 200 – 500.

Notons :

- S ce système (composé donc de 15 valeurs) : c'est un tableau trié ;
- m le montant à obtenir ;
- T le tableau contenant (à la fin de l'algorithme) toutes les valeurs telles que leur somme soit égale à m ;

Alors, l'algorithme glouton correspondant à notre problème est le suivant :

```
i ← 14, r ← 0
Tant que m ≠ 0:
   Si m ⩾ S[i]:
     T[r] ← S[i]
     r ← r+1
     m ← m - S[i]
   Sinon:
     i ← i-1
   Fin du Si
Fin du Tant que
```

COURS

Voyons chacune des étapes pour $m = 746$:

T	m	i	r	$m \neq 0$	$m \geqslant S[i]$
[]	746	14	0	vrai	vrai
[500]	246	14	1	vrai	faux
[500]	246	13	1	vrai	vrai
[500,200]	46	13	2	vrai	faux
[500,200]	46	12	2	vrai	faux
[500,200]	46	11	2	vrai	faux
[500,200]	46	10	2	vrai	vrai
[500,200,20]	26	10	3	vrai	vrai
[500,200,20,20]	6	10	4	vrai	faux
[500,200,20,20]	6	9	4	vrai	faux
[500,200,20,20]	6	8	4	vrai	vrai
[500,200,20,20,5]	1	7	5	vrai	faux
[500,200,20,20,5]	1	8	5	vrai	faux
[500,200,20,20,5]	1	6	5	vrai	vrai
[500,200,20,20,5,1]	0	6	6	faux	—

ANECDOTE

Pourquoi n'y a-t-il pas de pièces ou de billets de 7 € ? Parce que si tel était le cas, l'algorithme glouton du rendu de monnaie ne serait plus optimal (alors qu'il l'est avec le système monétaire actuel).
En effet, supposons qu'il y ait une pièce ou un billet de 7 €. Sur un montant de 14 €, l'algorithme donne la décomposition $14 = 10 + 2 + 2$ alors que la décomposition $14 = 7 + 7$ serait optimale.
Afin que l'algorithme glouton soit optimal, il est nécessaire qu'un système monétaire soit au minimum composé des premiers termes d'une suite *super-croissante*, c'est-à-dire une suite (u_n) telle que pour tout entier naturel n,

$$u_{n+1} > \sum_{k=0}^{n} u_k.$$

Mais ce n'est pas suffisant (ça le serait si on ne pouvait rendre qu'une pièce ou qu'un billet d'un montant donné, ce qui n'est pas le cas).
On dit que le système est *canonique* s'il est composé de valeurs d'une suite super-croissante et si l'algorithme glouton est optimal ; le système monétaire européen est canonique.

Solution de l'exercice type

Lycée Camille Claudel, Blois

1 Un algorithme de calcul de la variance est le suivant :

```
V ← 0
m ← moyenne
Pour i allant de 1 à n:
   V ← V + (m-x[i])*(m-x[i])
Fin du Pour
V ← V/n
```

2 En dehors de la boucle itérative, il y a 3 affectations.

Dans la boucle, il y a 1 affectation par itération. Donc en tout, il y a $n + 3$ affectations.

Le coût de l'algorithme est alors linéaire.

3 Supposons que la moyenne ne soit pas connue (ce qui est souvent le cas quand un utilisateur entre les valeurs au clavier). Il faut alors la calculer :

```
V ← 0
Pour i allant de 1 à n:
   V ← V + (moyenne(X)-x[i])*(moyenne(X)-x[i])
Fin du Pour
V ← V/n
```

Dans ce cas, on suppose que la fonction `moyenne(X)` calcule la moyenne des valeurs du tableau `X`. Cette fonction est de coût linéaire (comme nous l'avons vu dans le cours) et il y a n affectations (1 par itération) ; le coût final est donc d'ordre $n \times n$, soit quadratique.

À RETENIR

Si `f` est une fonction à coût linéaire, alors la boucle :

```
Pour i allant de 1 à n:
   x ← x + f(i)
Fin du Pour
```

est à coût quadratique.

Voir énoncé page 211

INTERROS

1 QCM Coût d'un algorithme

5 min Corrigé p. 233

Pour chacune des questions suivantes, 2 choix sont proposés. Quel est le bon ?

1. Le coût d'un algorithme de tri par insertion est :
   - a linéaire
   - b quadratique
2. Le coût d'un algorithme de recherche dans un tableau est :
   - a linéaire
   - b quadratique
3. Le coût d'un algorithme de recherche de maximum est :
   - a linéaire
   - b quadratique
4. Un algorithme dans lequel il y a une boucle itérative qui appelle une fonction de coût linéaire a un coût :
   - a linéaire
   - b quadratique

2 V/F Connaître le cours

5 min Corrigé p. 233

Les affirmations suivantes sont-elles vraies ou fausses ?

1. Le coût d'un algorithme de calcul de moyenne est linéaire.
2. Les algorithmes gloutons donnent toujours une solution optimale.
3. Un variant de boucle permet de corriger un algorithme.
4. Un algorithme dans lequel il y a une boucle itérative dans une autre boucle itérative peut avoir un coût quadratique.

Algorithmes de tri

3 Tri à bulle

★ 15 min Corrigé p. 233

Lycée François Mauriac, Bordeaux

On considère l'algorithme de la page ci-contre.

1. Si `a=[3,2,1]` au début de l'algorithme, qu'en est-il à la fin ?
2. Expliquer ce que fait cet algorithme.
3. Quel est son coût dans le pire des cas ?

```
a est un tableau de n valeurs
passage ← 0
permut ← Vrai
Tant que permut = Vrai:
   permut ← Faux
   Pour i allant de 0 à n-2-passage:
     Si a[i] > a[i+1]:
       temp ← a[i+1]
       a[i+1] ← a[i]
       a[i] ← temp
       permut ← Vrai
     Fin du Si
   Fin du Pour
   passage ← passage + 1
Fin du Tant que
```

4 Le tri Gnome

★★ 20 min Corrigé p. 235

Lycée Camille Claudel, Blois

On considère l'algorithme suivant :

```
a est un tableau de n valeurs
i ← 1
Tant que i < n:
   Si a[i] ⩾ a[i-1]:
     i ← i+1
   Sinon:
     temp ← a[i]
     a[i] ← a[i-1]
     a[i-1] ← temp
     Si i > 1:
       i ← i-1
     Fin du Si
   Fin du Si
Fin du Tant que
```

1 Exécuter cet algorithme sur le tableau `a = [3,2,1]`.

2 Quel est son coût ?

5 Le tri à peigne

★★★ 20 min Corrigé p. 235

Lycée Sainte-Clotilde, Strasbourg

On considère l'algorithme suivant :

```
a est un tableau de n valeurs
gap ← n-1
permut ← Vrai
Tant que permut = Vrai ou gap > 1:
   permut ← Faux
   gap ← gap / 1.3
   Si gap < 1:
     gap ← 1
   Fin du Si
   Pour i allant de 1 à n-1 avec un pas de gap:
     Si a[i] > a[i+gap]:
       temp ← a[i]
       a[i] ← a[i+gap]
       a[i+gap] ← temp
       permut ← Vrai
     Fin du Si
   Fin du Pour
Fin du Tant que
```

1 En comparant cet algorithme aux différents algorithmes précédents, donner son coût.

2 Quelle incidence le changement de la valeur « 1,3 » aurait sur cet algorithme ?

Correction d'algorithmes

6 Factorielle

★ 10 min Corrigé p. 236

Lycée Henri Bergson, Angers

On considère l'algorithme suivant :

```
i ← 0
f ← 1
Tant que i < n:
   i ← i+1
   f ← f*i
Fin du Tant que
```

1 Justifier que la boucle n'est pas infinie.

2 Montrer, en utilisant la pré-condition $n \geqslant 0$, que la propriété :

$$P : \texttt{f=i! et i} \leqslant \texttt{n}$$

est un invariant de cette boucle.

3 Formuler une post-condition qui établit la correction du programme.

7 Calcul du PGCD

★★ 15 min Corrigé p. 237

Lycée Henri Alain-Fournier, Bourges

On considère l'algorithme suivant :

```
a > b deux entiers
r entier
Tant que b > 0:
   r ← a%b
   a ← b
   b ← r
Fin du Tant que
```

1 Justifier que la boucle n'est pas infinie.

2 Trouver un invariant de cette boucle.

Aide : on pourra introduire deux variables `x` et `y` prenant les valeurs de `a` et `b` avant l'entrée dans la boucle, avec la pré-condition $y \geqslant 0$.

Note : « $a\%b$ » représente le reste de la division euclidienne de a par b.

Remarque : une propriété mathématique nous dit que si $a = bq + r$ alors $\text{pgcd}(a,b) = \text{pgcd}(b,r)$.

8 Palindrome

★★ 15 min Corrigé p. 238

Lycée Camille Jullian, Bordeaux

On considère l'algorithme suivant :

```
mot est une chaîne de caractères de longueur n
i ← 0
j ← n-1
p ← Vrai
Tant que i ⩽ j:
   Si mot[i] = mot[j]:
      i ← i+1
      j ← j-1
   Sinon:
      p ← Faux
   Fin du Si
Fin du Tant que
Afficher p
```

1 Exécuter cet algorithme pour `mot = radar`. Quelle est la fonction de cet algorithme ?

2 Montrer que `d = j-i` est un variant de la boucle « Tant que ».
En déduire que cette boucle se termine.

3 Quel est le coût de cet algorithme ?

INTERROS

9 Puissances de 2

★★ 15 min Corrigé p. 238

Lycée Turgot, Paris

Trouver un invariant de boucle dans l'algorithme suivant, puis en déduire la valeur affichée à la fin de son exécution.

```
n est un entier
p ← 1
Pour i allant de 1 à n:
   p ← 2*p
Fin du Pour
```

10 Contrôle de puissance

★★★ 20 min Corrigé p. 239

Lycée Jean Macé, Niort

1. Écrire un algorithme utilisant une boucle « Tant que » permettant de déterminer si un entier n strictement positif donné est une puissance entière de 2.
2. Montrer que la boucle « Tant que » se termine.
3. Montrer que l'algorithme produit le résultat attendu.

11 Trouver l'invariant

★★★ 20 min Corrigé p. 240

Lycée Sainte-Anne, Brest

Montrer que « $z \times x^k = b^n$ et $k \geqslant 0$ » est un invariant de la boucle de l'algorithme suivant, puis expliquer ce que fait cet algorithme.

```
n > 0 est un entier
b est un réel
x ← b
k ← n
z ← 1
Tant que k ≠ 0:
   Si k%2 = 1:
     z ← z*x
   Fin du Si
   x ← x*x
   k ← E(k/2)
Fin du tant que
```

Conception d'algorithmes

12 Fonction d'échange

★ 10 min Corrigé p. 241

Lycée Richelieu, Versailles

Étant donné un tableau `tab` et deux variables `m` et `n` contenant les indices de deux cases du tableau à permuter, écrire un algorithme d'une fonction `echange(tab, m, n)` qui échange les valeurs situées aux indices `m` et `n`, puis implémentez cet algorithme en Python.

Remarque : cette fonction ne retourne rien car le tableau est modifié *en place*, donc pas besoin de `return`.

Algorithmes gloutons

13 Le problème du sac à dos

★★ 20 min Corrigé p. 242

Lycée Camille Jullian, Bordeaux

On dispose de n objets x_1, ... , x_n, ayant chacun une valeur v_i et une masse m_i, pour $1 \leqslant i \leqslant n$.

On possède un sac à dos ne pouvant supporter qu'une masse maximum M. On cherche à le remplir avec le plus d'objets possibles afin que la valeur totale de ces objets soit maximale.

Approche mathématique

On note μ la masse totale des objets mis dans le sac qui satisfont nos conditions, et V leur valeur totale.

Justifier les écritures :

$$\mu = \sum_{i=1}^{n} \delta_i m_i \qquad ; \qquad V = \sum_{i=1}^{n} \delta_i v_i \ ,$$

où $\delta_i \in \{0 \ ; 1\}$.

L'algorithme

1. Quelles sont les conditions sur V et μ ?
2. On souhaite trier les objets par ordre de priorité décroissante. Que cela signifie-t-il ?
3. Proposer alors un algorithme glouton répondant à notre problème, c'est-à-dire indiquant les objets à mettre dans le sac.

INTERROS

14 Loueur de camions

★★★ 30 min Corrigé p. 243

Lycée Henri Bergson, Angers

Retrouvez le corrigé de cet exercice en vidéo.

Un loueur de camions reçoit des demandes de location d'un camion sous la forme $[d_i\ ;\ \ f_i]$, où d_i est la date du début de la location et f_i, la date de la fin de la location, d_i et f_i étant deux nombres de $[\![1\ ;\ 365]\!]$.

Imaginer un algorithme glouton permettant de sélectionner au mieux les clients.

15 Deuxième valeur minimale

★★★ 30 min Corrigé p. 244

Lycée Jean Macé, Niort

On considère un tableau T de n nombres entiers ($n \geqslant 2$). On souhaite écrire un algorithme permettant de trouver la **deuxième plus petite valeur** de ce tableau.

1. Écrire un algorithme en pseudo-code qui détermine la deuxième plus petite valeur de T.
2. Déterminer le coût de cet algorithme en fonction de n dans le pire des cas.
3. Modifier l'algorithme pour qu'il affiche également l'indice de la deuxième plus petite valeur.

1 QCM Coût d'un algorithme

Énoncé p. 226

1 Réponse $\boxed{\text{b}}$. Un algorithme de tri par insertion (comme par sélection) a un coût quadratique.

2 Réponse $\boxed{\text{a}}$. Un algorithme de recherche séquentielle dans un tableau a un coût linéaire.

3 Réponse $\boxed{\text{a}}$. Un algorithme de recherche séquentielle dans un tableau a un coût linéaire.

4 Réponse $\boxed{\text{b}}$. En effet, la boucle itérative peut entraîner n affectations, et pour chacune d'elles, il y en a n autres (parce que la fonction appelée est de coût linéaire). Ainsi, il y aura un coût d'environ $n \times n = n^2$.

Le coût est donc quadratique.

2 V/F Connaître le cours

Énoncé p. 226

1 La proposition est *vraie*. Nous l'avons vu dans le cours, à la page 214.

2 La proposition est *fausse*. Nous avons vu dans le cours un contre-exemple sur le problème du rendu de monnaie quand on utilise un système non canonique (voir l'anecdote de la page 224).

3 La proposition est *fausse*. En effet, pour corriger un algorithme, on utilise un *invariant* de boucle (et non un *variant*).

4 La proposition est *vraie*. En effet, si la première boucle itérative est du type « Pour i allant de 1 à n » et si la seconde est de la forme « Pour j allant de 1 à n », alors il y aura $n \times n$ affectations au minimum, ce qui donne un coût quadratique à l'algorithme.

3 Tri à bulle

Énoncé p. 226

Lycée François Mauriac, Bordeaux

1 Au début, `passage=0`, `permut=Vrai` et `n=3`.

- Entrée dans la boucle itérative : `i=0`. `a[0] > a[1]` (3 > 2) donc `temp=2`, `a[1]=3`, `a[0]=2` et `permut=Vrai`.
- `i=1 : a[1]>a[2]` (3 > 1) donc `temp=1`, `a[2]=3`, `a[1]=1` et `permut=Vrai`.
- Fin du Pour : `passage=1`. À ce stade, `a=[2,1,3]`.
- `permut=Vrai` donc on entre dans la boucle « Tant que ».
- `permut=Faux`, `n-2-passage=3-2-1=0` et `i=0 : a[0]>a[1]` (2 > 1) donc `temp=1`, `a[1]=2`, `a[0]=1` et `permut=Vrai`.
- Fin de la boucle itérative Pour. À ce stade, `a=[1,2,3]`, `permut=Vrai` et `passage=2`.

COURS INTERROS CORRIGÉS

- Entrée dans la boucle « Tant que » car `permut=Vrai` : `permut=Faux` et comme `n-2-passage=-1`, on n'entre pas dans la boucle « Pour ».
- Fin de la boucle « Tant que » car `permut=Faux`.

Ainsi, le tableau `[3,2,1]` devient `[1,2,3]`.

Une autre façon de représenter les résultats successifs est le tableau suivant :

a	[3,2,1]	[2,3,1]	[2,1,3]	[1,2,3]	[1,2,3]
permut	vrai	vrai	vrai	vrai	faux
n-2-passage	1	1	0	-1	–
i	0	1	0	–	–
a[i]>a[i+1]	vrai	vrai	vrai	–	–
temp	2	1	1	–	–
a[i+1]	3	3	2	–	–
a[i]	2	1	1	–	–
passage	0	1	2	3	–

2 La boucle itérative nous informe que l'on parcourt le tableau `a`. La boucle conditionnelle « Si a[i] > a[i+1] » nous dit que l'on intervertit deux valeurs consécutives si la première est supérieure à la seconde.

Ainsi, on parcourt le tableau et on intervertit deux valeurs consécutives s'il le faut pour les mettre dans l'ordre croissant. Si tel est le cas, le booléen `permut` vaut *Vrai*.

La boucle « Tant que » nous informe que ce processus est répété jusqu'à ce qu'il n'y ait plus d'échange, donc jusqu'à ce que les valeurs du tableau soit toutes dans l'ordre croissant.

L'algorithme trie alors le tableau initial.

3 Le pire des cas est un tableau où toutes les valeurs sont dans l'ordre décroissant. Dans ce cas, la boucle « Pour » effectue dans un premier temps $n-1$ itérations, puis $n-2$ itérations, etc. jusqu'à 1 itération, soit au total :

$$\sum_{k=1}^{n-1}(n-k) = n^2 - (1+2+3+\cdots+1) = n^2 - \frac{n(n+1)}{2} = \frac{n^2-n}{2}$$

itérations.

Le coût de l'algorithme est donc de l'ordre de n^2.

Il est donc quadratique.

4 Le tri Gnome

Énoncé p. 227

Lycée Camille Claudel, Blois

1 Aidons-nous d'un tableau pour exécuter cet algorithme ($n = 3$) :

`a`	[3,2,1]	[2,3,1]	[2,3,1]	[2,1,3]	[1,2,3]	[1,2,3]	[1,2,3]
`i`	1	1	2	1	1	2	3
`i < n`	vrai	vrai	vrai	vrai	vrai	vrai	faux
`a[i]` ⩾ `a[i-1]`	faux	vrai	faux	faux	vrai	vrai	–
`temp`	2	–	1	1	–	–	–
`a[i]`	3	–	3	2	–	–	–
`a[i-1]`	2	–	1	1	–	–	–
`i > 1`	faux	–	vrai	faux	–	–	–

2 L'algorithme est similaire au tri par insertion, sauf que, au lieu d'insérer directement l'élément à sa bonne place, l'algorithme effectue une série de permutations, comme dans un tri bulle.

L'algorithme a donc le même coût que le tri par insertion, c'est-à-dire quadratique.

5 Le tri à peigne

Énoncé p. 228

Lycée Sainte-Clotilde, Strasbourg

1 Cet algorithme ressemble à l'algorithme de tri à bulle. La différence est dans le pas de la boucle itérative (dans l'algorithme à bulle, il est égal à 1 alors qu'ici, il est variable).

L'objectif ici est de ne plus comparer uniquement les éléments adjacents d'un tableau, mais de commencer par comparer des éléments plus lointains, puis de raccourcir l'intervalle entre les éléments pour aboutir à un tri à bulle classique exécutable plus rapidement du fait que le tableau est déjà mieux trié.

Ainsi, dans le pire des cas, le coût de l'algorithme est quadratique.

Remarque : dans le tri à bulle, les grandes valeurs en début de tableau montent rapidement vers la fin (on parle de « lièvres »), alors que les petites valeurs proches de la fin du tableau ne redescendent que très lentement vers le début (on les appelle les « tortues »).

2 L'instruction `gap` $\leftarrow$ `gap/1.3` signifie que le pas diminue de plus en plus jusqu'à valoir 1 (à l'aide de la condition « Si gap < 1 »). Si on change cette valeur pour un nombre plus petit (par exemple 1.25), cela aura pour conséquence un plus grand nombre de comparaisons et donc un ralentissement de l'algorithme.

COURS INTERROS CORRIGÉS

Au contraire, si on l'augmente légèrement, par exemple en lui attribuant la valeur de 1.5, les « tortues » ne sont pas toutes éliminées quand le pas est devenu égal à 1, et que l'on aboutit à un tri à bulle ce qui, au final, ralentit le tri.

ANECDOTE

Ce tri est aussi appelé le « tri de Dobosiewicz », du nom de son créateur, Wodzimierz Dobosiewicz en 1980. Il a été redécouvert et popularisé par Stephen Lacey et Richard Box dans un article publié en avril 1991 dans la revue *Byte Magazine*. Les auteurs de cet article ont proposé un coefficient égal à 1,3 pour une meilleure efficacité de l'algorithme.

6 Factorielle

Énoncé p. 228

Lycée Henri Bergson, Angers

1 La variable `i` vaut initialement 0. À chaque itération, elle est incrémentée. Or, `n` est constante donc il existe une itération à partir de laquelle `i = n`, ce qui aura pour conséquence la fin de la boucle.

2 On suppose ici que `n`$\geqslant$`0`.

- Avant d'entrer dans la boucle, `i=0` et `f=1=0!=i!` (par convention). De plus, `i` est bien inférieure ou égale à `n`, puisque les deux variables valent le même nombre.
  Ainsi, la propriété P est vraie avant de rentrer dans la boucle.
- Notons `i1`, `f1` les valeurs de `i` et `f` lors d'une certaine étape, et supposons P vraie à cette étape. Donc `f1 = (i1)!` et `i1 < n`.
  Notons `i2` et `f2` les valeurs de ces mêmes variables après l'exécution des instructions de la boucle. On a ainsi :
  $$\texttt{i2 = i1+1} \leqslant \texttt{n}$$
  et
  $$\texttt{f2 = f1*(i1+1) = (i1!)*(i1+1) = (i1+1)!.}$$
  Ainsi, P est vraie après exécution de la boucle, donc P est un invariant de la boucle.
- À la sortie de la boucle, on a `i` $\geqslant$ `n` (négation de la condition), et P est vraie (donc `f=i!` et `i`$\leqslant$`n`). Donc `i = n`, d'où `f = n!`, qui est une bonne post-condition pour prouver la correction de l'algorithme.

7 Calcul du PGCD

Énoncé p. 229

Lycée Henri Alain-Fournier, Bourges

1 `r` représente le reste de la division euclidienne de a par b, donc $0 \leqslant r < b$ par définition mathématique. Or, `b` devient `r` à chaque itération, donc `r` diminue à chaque itération. Comme `r` est un entier naturel, il existe une itération pour laquelle `r=0`, donc pour laquelle `b=0`, ce qui assure une sortie de la boucle.

2 Pré-condition : $y \geqslant 0$. Considérons l'algorithme complété suivant :

```
a > b deux entiers
x ← a
y ← b
r entier
Tant que b > 0:
   r ← a%b
   a ← b
   b ← r
Fin du Tant que
```

Posons alors :

$$P : \text{« pgcd}(a,b) = \text{pgcd}(x,y) \text{ et } b \geqslant 0.\text{ »}$$

- Avant l'entrée dans la boucle, P est vraie.
- Notons `a1`, `b1` et `r1` les valeurs de `a`, `b` et `r` à une certaine étape.
  Supposons que P est vraie à cette étape ; donc pgcd(`a1`,`b1`) = pgcd(`x`,`y`) et `b1` $\geqslant$ 0.
  Notons `a2`, `b2` et `r2` les valeurs de `a`, `b` et `r` lors de l'itération suivante.
  Alors, `r2 = a1%b1`, `a2=b1` et `b2=r2`.
  D'après la propriété mathématique rappelée en énoncé,
  pgcd(`a2`,`b2`) = pgcd(`b1`,`r2`) = pgcd(`a1`,`a2`) = pgcd(`x`,`y`).
  De plus, `b2` $\geqslant$ 0 car `r2` $\geqslant$ 0.
  Ainsi, la propriété P est vraie à cette étape ; c'est donc un invariant de la boucle « Tant que ».

La post-condition qui découle de ceci est que pgcd(`x`,0) = pgcd(`a`,`b`), et donc $x = \text{pgcd}(a,b)$.

COURS INTERROS CORRIGÉS

CORRIGÉS

8 Palindrome

Énoncé p. 229

Lycée Camille Jullian, Bordeaux

1 Utilisons un tableau de variables pour exécuter l'algorithme.

i $\leqslant$ j et p = vrai	–	vrai	vrai	vrai	faux
mot[i] = mot[j]	–	vrai	vrai	vrai	–
i	0	1	2	3	–
j	4	3	2	1	–
p	vrai	vrai	vrai	vrai	–

L'algorithme affiche la dernière valeur de `p`, c'est-à-dire ici « vrai ».

Cet algorithme affiche donc si le mot est un palindrome (mot qui se lit de façon identique de gauche à droite et de droite à gauche) ou pas.

2 `d = j-i` représente la différence entre `i` et `j`. Or, à chaque itération, `i` est incrémentée (donc augmente) et `j` est décrémentée (donc diminue), ce qui implique que `d` diminue. Cette variable étant un entier (relatif), elle va valoir 0 lors d'une itération, puis -1, ce qui assure la sortie de la boucle « Tant que ».

3 Le coût de cet algorithme est linéaire. En effet, le nombre d'itérations dépend directement de la longueur n de la chaîne de caractères donc le nombre d'affectations et autres opérations logiques est proportionnel à n.

9 Puissances de 2

Énoncé p. 230

Lycée Turgot, Paris

Posons :

$$P : « \, p = 2^i \, »$$

- avant d'entrer dans la boucle itérative, $p = 1 = 2^0 = 2^n$, donc la propriété P est vraie.
- Notons p_1 et i_1 les valeurs de `p` et `i` lors d'une itération. Notons p_2 et i_2 les valeurs de ces mêmes variables à l'itération suivante. Alors :

$$i_2 = i_1 + 1 \qquad \text{et} \qquad p_2 = 2 \times p_1 = 2 \times 2^{i_1} = 2^{i_1+1} = 2^{i_2}.$$

La propriété P est donc vraie, ce qui montre que P est un invariant de la boucle itérative.
Quand la boucle est finie, $i = n$ et P est vraie, donc $p = 2^n$.
L'algorithme permet donc de calculer la puissance de 2 lorsque l'exposant est égal à un entier n.

10 Contrôle de puissance

Énoncé p. 230

Lycée Jean Macé, Niort

1 Un algorithme possible est le suivant :

```
x est un nombre entier entré par l'utilisateur
p ← Vrai
Si x%2 = 1:
   p ← Faux
Sinon:
   Tant que x%2 = 0:
     x ← E(x/2) (partie entière de x/2)
     n ← n+1
   Fin du Tant que
   Si x ≠ 1:
     p ← Faux
   Fin du Si
Fin du Si
Afficher p
```

Explications :

- on commence avant tout par tester si x est pair (dans ce cas, $x\%2 = 0$). Si tel n'est pas le cas, le booléen p vaut *Faux* ;
- si x est pair, on le divise par 2 autant de fois que possible (boucle « Tant que »), c'est-à-dire tant que le résultat est pair ;
- la boucle terminée, si x vaut 1, c'est que l'on a divisé par 2 tout le temps, et donc que x est une puissance de 2. Sinon, x n'est pas une puissance de 2, et le booléen p vaut alors *Faux*.

Remarque : la variable n n'est pas utile au premier abord, mais elle nous servira pour trouver un invariant de boucle à la question 3.

2 Un variant de boucle possible est le reste r de x dans sa division euclidienne par 2 ($r = x\%2$). En effet, le reste vaut 0 ou 1 dans une telle division et prend nécessairement la valeur 1 :

- soit quand x n'est pas une puissance de 2 (le résultat de la division sera alors à un moment égal à un nombre impair plus grand que 1),
- soit quand x est une puissance de 2 (en divisant par 2 tout le temps, on arrive à un moment à $2 \div 2 = 1$ et là, le reste est égal à 1).

La boucle n'est donc pas infinie.

3 Avant d'entrer dans la boucle « Tant que », posons $a = x$ et la pré-condition « a est une puissance de 2 ». Posons alors :

$$P : « a = 2^m, x = a \div 2^n \text{ et } n \leqslant m ».$$

CORRIGÉS

Montrons que c'est un invariant de la boucle « Tant que » :

- avant d'entrer dans la boucle, $x = a = a \div 2^0 = a \div 2^n$. De plus, $n = 0 \leqslant m$ car m est un entier naturel.
  Donc P est vraie ;
- posons x_1 et n_1 les valeurs de x et n lors d'une itération où P est vraie. Ainsi, $x_1 = \dfrac{a}{2^{n_1}}$ et $n_1 < m$ (différent de m car il faut qu'il y ait une autre itération).
  Posons ensuite x_2 et n_2 les valeurs de ces mêmes variables à l'itération suivante. Alors, $n_2 = n_1 + 1$ et :
  $$x_2 = \frac{x_1}{2} = \frac{1}{2}x_1 = \frac{1}{2} \times \frac{a}{2^{n_1}} = \frac{a}{2^{n_1+1}} = \frac{a}{2^{n_2}}.$$
  De plus,
  $$n_2 = n_1 + 1 < m + 1 \leqslant m.$$
  P est donc vraie à cette étape.

Donc P est un invariant de la boucle « Tant que ».

La boucle étant finie, `n` contient le nombre de fois que l'on a divisé `x` par 2 et `x` vaut 1, donc :
$$1 = \frac{a}{2^n}, \qquad \text{soit} \qquad a = 2^n.$$

11 Trouver l'invariant

Énoncé p. 230

Lycée Sainte-Anne, Brest

Posons :
$$P : \text{« } z \times x^k = b^n \text{ et } k \geqslant 0 \text{ ».}$$

- Avant d'entrer dans la boucle, $z \times x^k = 1 \times b^n = b^n$ et $k = n$ donc $k \geqslant n$. P est donc vraie.
- Posons z_1, x_1 et k_1 les valeurs respectives de `z`, `x` et `k` lors d'une itération où P est vraie. Alors,
  $$z_1 \times x_1^{k_1} = b^n \qquad \text{et} \qquad k_1 > 0.$$
  ($k_1 \neq 0$ car il faut qu'il y ait une autre itération)
  À l'étape suivante, notons z_2, x_2 et k_2 les valeurs des mêmes variables. Alors,
  – si k_1 est impair ($k_1 = 2p_1 + 1$) :
  $$z_2 = z_1 \times x_1$$
  et
  $$x_2 = x_1 \times x_1 \qquad \text{et} \qquad k_2 = E(k_1/2) = p_1.$$
  Dans ce cas,
  $$z_2 \times x_2^{k_2} = \underbrace{z_1 \times x_1}_{=z_2} \times \left(x_1^2\right)^{p_1} = z_1 \times x_1 \times x_1^{2p_1} = z_1 \times x_1 \times x_1^{k_1-1} = \underbrace{z_1 \times x_1^{k_1}}_{=b^n}.$$

– Si k_1 est pair ($k_1 = 2p_1$) :

$$x_2 = x_1 \times x_1 \qquad \text{et} \qquad k_2 = E(k_1/2) = p_1.$$

Dans ce cas,

$$z_2 \times x_2^{k_2} = z_1 \times \left(x_1^2\right)^{p_1} = z_1 \times x_1^{2p_1} = z_1 \times x_1^{k_1} = b^n.$$

Dans les deux cas, P est vraie.

Donc P est bien un invariant de la boucle.

Quand la boucle se termine, $k = 0$ et P est vraie, donc $z \times x^0 = b^n$, soit $z = b^n$.
L'algorithme calcule donc b à l'exposant n.

12 Fonction d'échange

Énoncé p. 231

Lycée Richelieu, Versailles

Un algorithme possible est le suivant :

```
tab : liste de valeurs
m, n : deux entiers
Si m < len(tab) et n < len(tab):
        temp ← m
        m ← n
        n ← temp
Fin du Si
```

Un implémentation possible en Python est alors la suivante :

```python
def echange(tab, m, n):
    temp = tab[m]
    tab[m] = tab[n]
    tab[n] = temp
```

Il est à noter que Python permet la solution suivante :

```python
def echange(tab, m, n):
    tab[m], tab[n] = tab[n], tab[m]
```

CORRIGÉS

13 Le problème du sac à dos

Énoncé p. 231

Lycée Camille Jullian, Bordeaux

Approche mathématique

On passe en revue tous les objets x_i ; si on met x_i dans le sac alors la masse du contenu du sac augmente de m_i, sinon elle augmente de 0. On peut donc poser :

$$\begin{cases} \delta_i = 1 & \text{si } x_i \text{ est mis dans le sac,} \\ \delta_i = 0 & \text{si } x_i \text{ n'est pas mis dans le sac.} \end{cases}$$

Ainsi, la masse totale des objets mis dans le sac est :

$$\mu = \sum_{i=1}^{n} \delta_i m_i.$$

Sur le même principe, la valeur totale V des objets mis dans le sac est :

$$V = \sum_{i=1}^{n} \delta_i v_i.$$

L'algorithme

1 Le masse totale des objets doit être inférieure ou égale à M, d'où la première condition : $\mu \leqslant M$.

Il faut ensuite obtenir une valeur totale optimale, d'où la condition : V optimale.

2 Un « tri optimisé » est, dans notre problème, des objets triés selon leur valeur (de la plus grande jusqu'à la plus petite), puis leur masse (de la plus petite jusqu'à la plus grande).

Trier les objets d'abord selon leur masse croissante n'est pas nécessairement un bon choix car les objets légers peuvent ne pas valoir cher.

3 L'algorithme doit parcourir tous les objets triés selon les critères mentionnés précédemment, ajoutés dans le sac tant que la masse totale ne dépasse pas M. On peut alors considérer l'algorithme page suivante.

Explications :

- on parcourt toutes les valeurs de `m` et `v` (ligne 1) ;
- Si la somme de la masse totale précédente et de la masse de l'objet sur lequel nous sommes est inférieure ou égale à la masse totale possible M (ligne 2) alors on indique que l'on prend l'objet ($\delta_i = 1$, ligne 3) en mettant « 1 » dans le tableau des coefficients ; sinon (ligne 6), on indique que l'on ne prend pas l'objet ($\delta_i = 0$, ligne 7) ;
- à la sortie de la boucle itérative, on a un tableau de la forme `d = [1,1,0,1,0,1]` indiquant quels sont les objets à mettre dans le sac à dos (objets correspondant aux nombres 1), ainsi que la masse totale (stockée dans la variable `MT`) et la valeur totale (`VT`).

COURS
INTERROS
CORRIGÉS

```
MT ← 0 (masse totale du contenu du sac)
VT ← 0 (valeur totale du contenu du sac)
m = tableau des masses des objets
v = tableau des valeurs des objets
d = tableau des coefficients
Pour i allant de 1 à n:                        (1)
   Si MT+m[i]⩽ M:                              (2)
     d[i] ← 1                                  (3)
     MT ← MT + m[i]                            (4)
     VT ← VT + v[i]                            (5)
   Sinon:                                      (6)
     d[i] ← 0                                  (7)
   Fin du Si
Fin du Pour
```

14 Loueur de camions

Énoncé p. 232

Lycée Henri Bergson, Angers

La première chose à faire est de se demander sur quel(s) critère(s) nous allons réfléchir pour optimiser les locations. Pour que deux demandes de location soient possibles, il faut que les périodes ne se chevauchent pas (par exemple, [2 ; 8] et [7 ; 9] sont impossibles à satisfaire).

On va donc trier les périodes selon leur date de fin.

Prenons l'exemple de la demande suivante :

Clients i	1	2	3	4	5	6	7	8	9
Débuts de périodes d_i	1	2	4	1	5	8	9	13	11
Fins de périodes f_i	8	5	7	3	9	11	10	16	14

Après avoir trié selon les dates de fins croissantes, on obtient :

Clients i	4	2	3	1	5	7	6	9	8
Débuts de périodes d_i	1	2	4	1	5	9	8	11	13
Fins de périodes f_i	3	5	7	8	9	10	11	14	16

Ensuite, on satisfait les demandes possibles de la gauche vers la droite :

- la demande du client 4 est satisfaite ;
- celle du client 2 ne l'est pas car la date du début est inférieure à celle de la fin de la demande précédente ;
- celle du client 3 est satisfaite car la date du début est supérieure à celle de la demande précédente ;
- etc.

Les clients 4, 3, 7 et 9 voient alors leur demande satisfaite.

CORRIGÉS

Un algorithme glouton est alors le suivant, après tri (donc sur le 2e tableau dans l'exemple que nous avons pris) :

```
c ← liste des numéros de clients (4, 2, ..., 8
                                    pour notre exemple)
s ← liste des numéros de clients à satisfaire
s[0] ← c[0] (1er client satisfait)
fin ← f[0]
Pour i allant de 1 à n-1:
   Si d[i] ⩾ fin:
     s[k+1] ← c[i]
     fin ← f[i]
   Fin du Si
Fin du Pour
```

Pour chaque client, on regarde si la date de début de période est supérieure ou égale à la date de fin de la période du dernier client satisfait. Si tel est le cas, ce client est ajouté à la liste des clients satisfaits et la variable `fin` prend la valeur de la date de fin de la période du client ajouté, pour servir de référence pour les clients suivants.

15 Deuxième valeur minimale

Énoncé p. 232

Lycée Jean Macé, Niort

1 Voici un exemple d'algorithme en pseudo-code qui détermine la deuxième plus petite valeur de T.

```
Entrée: un tableau T de n entiers (n ⩾ 2)
Sortie: la deuxième plus petite valeur du tableau T
min1 ← +∞
min2 ← +∞
Pour i allant de 0 à n-1 :
     Si T[i] < min1 alors
         min2 ← min1
         min1 ← T[i]
     Sinon si T[i] < min2 alors
         min2 ← T[i]
Fin du Pour
Afficher "La deuxième plus petite valeur est", min2
```

2 L'algorithme effectue une unique boucle sur les n éléments du tableau. À chaque itération, un nombre constant de comparaisons et affectations est réalisé. Le coût total de l'algorithme est donc *linéaire*, soit de l'ordre de $O(n)$.

3 Voici un algorithme qui affiche également l'indice de la deuxième plus petite valeur.

```
Entrée: un tableau T de n entiers (n ⩾ 2)
Sortie: la deuxième plus petite valeur du tableau T
        et son indice
min1 ← +∞, ind1 ← -1
min2 ← +∞, ind2 ← -1
Pour i allant de 0 à n-1 :
     Si T[i] < min1 alors
         min2 ← min1
         ind2 ← ind1
         min1 ← T[i]
         ind1 ← i
     Sinon si T[i] < min2 alors
        min2 ← T[i]
        ind2 ← i
Fin du Pour
Afficher "La deuxième plus petite valeur est", min2,
"à l'indice", ind2
```

Cet algorithme conserve le même coût linéaire $O(n)$.

Annexe

Liste des vidéos

Chapitre	Titre	Page
Chapitre 1	**Cours :** octogone de Fou-Hi	**2**
Chapitre 1	**Exercice 12 :** algorithme de conversion	**17**
Chapitre 2	**Exercice 25 :** jeu du « plus ou moins »	**56**
Chapitre 3	**Exercice 22 :** fichier INSEE	**112**
Chapitre 4	**V/F 2 :** rédaction de programmes	**145**
Chapitre 4	**Exercice 5 :** programme en C	**146**
Chapitre 4	**Exercice 8 :** convertisseur de temps	**148**
Chapitre 5	**Exercice type :** requêtes SQL	**157**
Chapitre 6	**Cours :** protocole du bit alterné	**195**
Chapitre 6	**Exercice 4 :** commande chmod sous Linux	**202**
Chapitre 7	**Exercice 14 :** loueur de camions	**232**