Conversions

• Du décimal au binaire

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

Ainsi, $\overline{25}^{10} = 2^4 + 2^3 + 2^0 = \overline{11001}^{2}$.

• Du binaire au décimal

Binaire	1	1	0	0	1	0	1
Puissances de 2	2^6	2^5	2^4	2^3	2^2	2^1	2^0
Décimal	$2^6 + 2^5 + 2^2 + 2^0$						

Ainsi, $\overline{1100101}^{2} = \overline{101}^{10}$.

• Du décimal à l'hexadécimal

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

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

• De l'hexadécimal au binaire

Hexadécimal	A	B	C	D	E	F
Décimal	10	11	12	13	14	15
Binaire	1010	1011	1100	1101	1110	1111

Ainsi, $\overline{BD}^{16} = \overline{10111101}^{2}$.

• Du binaire à l'hexadécimal

Binaire	0100	1101
Pseudo-décimal	4	13
Hexadécimal	4	D

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

Algèbre de Boole

• Conjonction logique

ET	Faux	Vrai
Faux	Faux	Faux
Vrai	Faux	Vrai

• Disjonction logique

OU	Faux	Vrai
Faux	Faux	Vrai
Vrai	Vrai	Vrai

Représentation binaire d'un entier

• Binaire pur

Codage dans lequel sont écrits les entiers *naturels*.
Avec une mémoire de n bits, tous les entiers de 0 à $2^n - 1$ sont représentés.
Exemples : $n = 8$ → les entiers de 0 à $2^8 - 1 = 255$ sont représentés.
$n = 24$ → les entiers de 0 à $2^{24} - 1 = 16\,777\,215$ sont représentés.

• Binaire signé

Codage dans lequel sont écrits les entiers *relatifs*.
Avec une mémoire de n bits, tous les entiers de $-(2^{n-1} - 1)$ à $2^{n-1} - 1$ sont représentés.
Le bit de poids fort représente le signe (« 0 » pour « + » et « 1 » pour « − »).

Exemple : $\overline{11011001}^{2} = \overline{-89}^{10}$.

• Complément à 2

Autre codage dans lequel sont écrits les entiers *relatifs*.

- si $n > 0$, on utilise le binaire signé ;
- si $n < 0$,
  → on code $|n|$ en binaire signé,
  → 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 : pour trouver le complément à 2 de $\overline{-57}^{10}$ avec une mémoire de 8 bits,

→ on code le nombre sans son signe en binaire signé : $\overline{00111001}^{2}$;

→ on prend le complément à 1 : $\overline{11000110}^{2}$;

→ on ajoute 1 au binaire obtenu : $\overline{11000111}^{2}$.

Représentation binaire d'un réel

• Nombres dyadiques

Ce sont les nombres de la forme $\frac{x}{2^n}$, où x est un nombre relatif et n un entier naturel.

• Développement dyadique

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

Exemple : $\frac{13}{4} = \frac{13}{2^2}$ et $\overline{13}^{10} = \overline{1101}^{2}$ donc $\frac{\overline{13}^{10}}{4} = \overline{11,01}^{2}$.

Le développement dyadique de tout nombre décimal non dyadique est infini ; par exemple, $\overline{0,1}^{10} = \overline{0,0001100110011\ldots}^{2}$.

• Représentation approximative d'un réel

Elle est obtenu à partir de l'écriture décimale du nombre.

Exemple : $\frac{1}{3} \approx 0,333\ldots$.

- $0,333\ldots \times 2 = \boxed{0},\underline{666\ldots}$
- $0,666\ldots \times 2 = \boxed{1},\underline{333\ldots}$

Donc $\frac{\overline{1}^{10}}{3} = \overline{0,010101\ldots}^{2}$.

Python

Calculs de base

Calcul	Syntaxe	Exemple	Résultat
Addition	+	13+6	19
Soustraction	-	13-6	7
Multiplication	*	13*6	78
Division décimale	/	13/6	2.1666666666666665
Division entière	//	13//6	2 (partie entière de $\frac{13}{6}$)
Reste de la division euclidienne	%	13%6	1
Puissances	**	2**5	32
Racine carrée	**0.5	9**0.5	3

Boucles/instructions itératives et conditionnelles

Dénomination	Syntaxe
Tant que $i = j$	`while i == j:`
Tant que $i \leqslant j$	`while i <= j:`
Tant que $i < j$	`while i < j:`
Tant que $10 < i < 20$	`while i > 10 and i < 20:`
Pour i allant de 0 à n	`for i in range(n+1):`
Pour i allant de 1 à n	`for i in range(1,n+1):`
Si i > j	`if i > j:`

Affectations et déclarations

Dénomination	Syntaxe
Affecter la valeur 4 à la variable k	`k = 4`
Ajouter 7 à k	`k += 7`
Soustraire 7 à k	`k -= 7`
Multiplier k par 7	`k *= 7`
Diviser k par 7	`k /= 7`
Affectation de plusieurs variables	`a,b,c = 3,4,5`
Déclarer une chaîne de caractères	`mot='Bonjour' / mot="Bonjour"`
Déclarer un tableau vide	`tableau = []`
Déclarer un tableau rempli	`tableau = [2,-3,5]`
Déclarer un tableau rempli de « 0 »	`tableau = [0]*5`
Déclarer un dictionnaire vide	`dico = {}`
Déclarer un dictionnaire rempli	`dico = {1:'Paul' , 2:'Jean'}`
Déclarer la matrice $\begin{pmatrix} 1 & 2 \\ 3 & 4 \end{pmatrix}$	m = [[1,2] , [3,4]]
Construire une matrice remplie de « 0 »	`add = [[0] * 9` `for i in range(9)]`

Par la suite, L désigne une liste et `dico` un dictionnaire.

Méthodes principales

Méthode	Dénomination	Exemple
`[2:5]`	Affiche les éléments de l'item 2 à 4.	L[2 :5]
`append`	Ajoute une valeur à la fin d'une liste.	`L.append(5)`
`copy`	Copie un dictionnaire ou une liste.	`dico2 = dico.copy()`
`count`	Compte le nombre d'occurrences d'une valeur dans une liste.	`L.count(7)`
`del`	Supprime une entrée avec un index.	`del L[1], del dico[clé]`
`extend`	Ajoute plusieurs éléments à la fin d'une liste.	`L.extend([5,6])`
`get`	Récupère la valeur associée à une clé.	`dico.get('clé')`
`insert`	Insère un élément à une position donnée.	`L.insert('a',8)`
`join`	Transforme une liste en chaîne de caractères, où chaque entrée est séparée par le séparateur indiqué (dans notre exemple, une virgule).	`chaine = ",".join(L)`
`len`	Compte le nombre d'entrées dans une liste.	`len(L)`
`remove`	Supprime la première occurrence de l'élément donné.	`L.remove('z')`
`reverse`	Inverse la liste.	`L.reverse()`
`split`	Divise une chaîne de caractères en une liste de sous-chaînes, en utilisant un séparateur spécifié.	`L = ",".split(chaine)`

Parcourir une liste, une chaîne de caractères ou un dictionnaire

Syntaxe	Dénomination
`for i in range(L):`	i désigne l'item sur lequel on est.
`for i in range(len(L)):`	i désigne l'index sur lequel on est.
`for cle,valeur in dico.items():`	Récupère la clé et la valeur de chaque entrée.

Fonction lambda

Syntaxe : `nom = lambda <variable> : expression`
Exemple : `carre = lambda x: x**2`

Tri avec une fonction lambda

Syntaxe : `L.sort(key = lambda colonnes:colonnes[n])`
Tri suivant la colonne $n + 1$ la liste L.

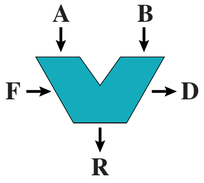
Interaction client-serveur

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 client est à l'initiative de la communication.
- le *serveur* est la machine sur laquelle est la page web : c'est elle qui répond aux requêtes.

Architecture de Von Neumann

Unité Arithmétique Logique (UAL)



- 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, ...).

Opérations possibles :

- addition, soustraction, multiplication, division ;
- opérations logiques (ET, OU, NON) ;
- opérations de comparaison (>, < et =).

Mémoire

Sorte de tableau dans lequel seront stockés aussi bien les données que les programmes.
Chaque cellule de ce tableau possède une adresse X et la valeur de chaque cellule est noté M[X].

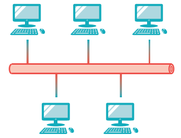
Unité de commande

Elle donne les ordres à toutes les autres parties de l'ordinateur.
Elle est principalement composée de quatre registres :

- le compteur ordinal ;
- l'instruction courante ;
- un ensemble de drapeaux servant pour les branchements conditionnels ;
- des registres permettant de stocker des opérandes et des valeurs intermédiaires.

Architecture réseau

Topologie en bus



Avantages :

- facilité de mise en œuvre ;
- faible coût.

Inconvénients :

- réseau inutilisable en cas de rupture du bus ;
- signal jamais régénéré.

Topologie en étoile

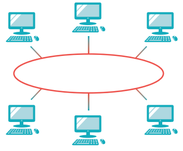
Avantages :

- précision d'envoi ;
- ajout facile de postes.

Inconvénients :

- dépend du nœud central ;
- coûteux (nécessite plusieurs câbles).

Topologie en anneau



Avantages :

- débit proche de 90% de la bande passante,
- signal régénéré par chaque station.

Inconvénients :

- réseau inutilisable si un poste ne fonctionne pas,
- s'il y a trop de machines, le temps de transmission s'allonge.