ALABOUCH Jasmine
110

Test fonctions
durée 10 minutes

3,5/5

Exercice 1
Fonctions

1,5/1,5 **Question 1.1** Donnez **le nom, les paramètres** et **le type de sortie** de la fonction suivante :

```
1 def inconnue(n):
2     m = 0
3     valeur = 0
4     while m <= n :
5         valeur = valeur + m
6         m = m + 1
7     return valeur
```

#Nom: inconnue ✓

#Parametres: m ; m et valeur (2 variables)

#Type de sortie: integer - entier ✓

0/1,5 **Question 1.2** Expliquez ce que fait la fonction et donnez un exemple d'exécution :

La fonction prend un paramètre n en entrée et renvoie ~~le nombre saisi par l'utilisateur~~ la somme de 0 à n
~~Si n = 2 valeur = 1~~
si n = 2 renvoie 3.

2/2 **Question 1.3** Quels sont les types respectifs des variables suivantes ?

```
1 a = 7.3
2 b = "False"
3 c = 2
4 d = c > a
```

a : float - flottant ✓
b : string - chaîne de caractère ✓
c : integer - entier ✓
d : booléen ✓

ALABOUCH Jasmine
11C

Test fonctions
durée 10 minutes

4,5/5

Exercice 1
Fonctions

1,5/2,5 **Question 1.1** Donnez **le nom**, **les paramètres** et **le type de sortie** de la fonction suivante :

```
1 def recherche(T,x):
2     res = False                    #Nom: recherche ✓
3     for i in range(len(T):
4         if T[i]==x:                #Parametres: un tableau T de type liste
5             res = True                  une inconnue x ✓
6         else:                      #Type de sortie:
7             res = False                 un Booléen (True ou False) ✓
8     return res
```

0/1 **Question 1.2** Que doit-on modifier dans la fonction précédente pour qu'elle renvoie Faux si et seulement si x n'appartient pas à T ?

elif T[i] != x: res = False remplacer else: res = False par if T[i] != x: res = False +

Question 1.3 Compléter la fonction **tris_selection** qui prend en entrée une liste non vide d'entiers, notée **T** et qui **renvoie** la liste triée par ordre croissant. **(bonus) Donner la complexité du tri par sélection.**

1,5/2,5

```
1 def tris_selection(T):
2     '''Trie le tableau tab dans l'ordre croissant par la
      methode du tri par selection.'''
3     N = len(T)
4     for i in range(.N....): ✓
5         mini = .min(T)  ✗ i
6         for j in range(mini , N ): ✓
7             if T[j] < T[i].. : ✗ T[mini]
8                 mini = j
9         T[i],T[mini] = T[mini],T[i] ✓
10    return T
```

Complexité $O(n^2)$ ✓

+ 1,5

ALABOUCH Jasmine 110

Lycée Albert Calmette — Première NSI

16/17

Êtes-vous au top ?

durée 1h

L'usage de la calculatrice est interdit

Exercice 1

Des hors-d'œuvres pour vous mettre en appétit

Question 1.1 Soit le code Python suivant :

```
a = (12,25,32)
print (a[1])
a[2] = 43
```

☐ À l'exécution, il va s'afficher 12 et le *tuple* vaudra `(43,25,32)`
☐ À l'exécution, il va s'afficher 25 et le *tuple* vaudra `(12,25,43)`
☒ À l'exécution, il va s'afficher 25 et afficher un message d'erreur
☐ À l'exécution, il va immédiatement s'afficher un message d'erreur

Question 1.2 Avec la variable alphabet définie par l'affectation suivante, quel est l'indice de la lettre `'E'`

```
alphabet = ['A','B','C','D','E','F','G','H']
```

E a pour indice 4.

Question 1.3 Quelle commande faut-il taper pour faire afficher l'élément d'indice 7 ?

Il faut taper ~~alphabet[~~ print(alphabet[7])

Question 1.4 Quel est le résultat de la fonction `range(1,9,2)` ?

Tous les chiffres de 2 en 2 compris entre 1 (inclus) et 9 (exclus) donc 8, soit 1 ; 2 ; 4 ; 6 ; 8.
1 3 5 7

Question 1.5 On désire écrire une fonction `table(n)` qui affiche la table de multiplication de `n` :

Un affichage typique serait (pour `table(2)`) :

```
1 x 2 = 2
.
.
.
10 x 2 = 20
```

Compléter le code suivant en remplaçant les pointillés par les bonnes instructions :

range(1, 11)

```
def table(n):
    for i in range(10) :
        r = i * n
        print(f"{i} x {n} = {r}")
```

Question 1.6 Cocher la ou les bonnes réponse concernant les listes...

- [x] elles sont mutables
- [x] elles sont itérables
- [ ] elles sont non ordonnées
- [x] on peut avoir leur taille avec la méthode `len()`
- [ ] elles sont interchangeables en tous points avec des *tuples*.

Question 1.7 Écrire le résultat de l'évaluation de l'expression Python suivante

```
[n * n for n in range(5)]
```

0 ; 1 ; 4 ; 9 ; 16

Question 1.8 Quelle est la valeur de la variable `table` après exécution du programme Python suivant ?

```
table = [12, 43, 6, 22, 37]
for i in range(len(table) - 1):
    if table[i] > table[i+1]:
        table[i],table[i+1] = table[i+1],table[i]
```

- [ ] `[6, 12, 22, 37, 43]`
- [ ] `[43, 12, 22, 37, 6]`
- [x] `[12, 6, 22, 37, 43]`
- [ ] `[43, 37, 22, 12, 6]`

Question 1.9 Quelle est la valeur de la variable `image` après exécution du code suivant ?

```
image = [[0,0,0,0],[0,0,0,0],[0,0,0,0],[0,0,0,0]]
for i in range(4):
    for j in range(4):
        if (i+j) == 3:
            image[i][j] = 1
```

- [ ] `[[0,0,0,0],[0,0,0,0],[0,0,0,0],[1,1,1,1]]`
- [ ] `[[1,0,0,0],[0,1,0,0],[0,0,1,0],[0,0,0,1]]`
- [ ] `[[1,0,0,0],[1,1,0,0],[1,1,1,0],[1,1,1,1]]`
- [x] `[[0,0,0,1],[0,0,1,0],[0,1,0,0],[1,0,0,0]]`

Exercice 2

Le plat de résistance : avez-vous bien travaillé vos séquences de cours ?

Question 2.1 Écrire ci-dessous une portion de code permettant de compter de 2 en 2, à partir de 5 et jusqu'à 29 inclus. Il vous est demandé d'utiliser obligatoirement une structure de type `while`.

```
i=0
i=5
while i!=29:
    i = 2+i
print(i)
```

Question 2.2 Écrire une fonction `est_majeur(age)` qui prend en paramètre une variable `age` de type entier et qui renvoie `True` si age est plus grand ou égal à 18 et `False` sinon.

```
def est_majeur(age):
    if age>=18:
        return True
    else:
        return False
    if age == not(int):
        return TypeError
```

Question 2.3 Écrire une portion de code qui fabrique la liste des 10 premiers nombres pairs à l'aide d'une structure de boucle `for` ou d'une liste par compréhension.

```
liste_pairs = [i for i in range(10) if i%2==0]
```

Question 2.4 Écrire une fonction `total` qui prend en paramètre une liste d'entiers `tab` et qui renvoie la `somme` de ces nombres.

```
def total(tab):
    somme=0
    for tab[i] in tab:
        somme += tab[i]
    return somme
```

Question 2.5 Quel est le résultat de l'évaluation de `mystere([2,4,1,9,8,7,3,6,12],4,10)` ?

```
def mystere(tab, a, b):
    resultat = []
    for e in tab :
        if e >= a and e < b:
            resultat.append(e)
    return resultat
```

[4,9,8,7,6]

Question 2.6 Expliquer ce que fait cette fonction.

Cette fonction prend un tableau de valeurs sous forme de liste dans un tulpe (a, b). Elle met dans une variable résultat chaque élément du tableau compris dans l'intervalle de a (inclus) et b (exclus).

On désire écrire une fonction `moyenne` qui prend en paramètres deux listes `notes` et `coefs` et qui renvoie la moyenne **coefficientée** des notes. Par exemple, la commande `moyenne([10,11,12,14], [1, 2, 3, 4])` renvoie `12.4` car on a :

$$\frac{10\times 1+11\times 2+12\times 3+14\times 4}{1+2+3+4}=12{,}4$$

Question 2.7 Quelle condition doit-on avoir concernant les tailles des 2 listes ?

Les deux listes doivent être de la même taille.

Question 2.8 Compléter les portions de code manquantes :

```
def moyenne(notes, coefs):
    num ..... = 0
    for i in range(len(notes)...):
        num = num + notes[i] * coefs[1,2,3,4.....]
    moy = num / (sum(coefs.....))
    return ..moy.....
```

Exercice 3

Un petit dessert type épreuve pratique du bac ?

On considère des chaînes de caractères, appelées '*mots à trous*', contenant uniquement des majuscules et des caractères `*`.

Par exemple `INFO*MA*IQUE`, `***I***E**` et `*S*` sont des mots à trous. On se propose de programmer une fonction `correspond` :

- qui prend en paramètres deux chaînes de caractères de mêmes longueurs `mot` et `mot_a_trous` où `mot_a_trous` est un mot à trous comme indiqué ci-dessus ;
- et qui renvoie :`True` si on peut obtenir `mot` en remplaçant convenablement les caractères `'*'` de `mot_a_trous` ; `False` sinon.

L'algorithme est le suivant :

---

1 **Algorithme :** fonction correspond()

Entrées : `mot` (chaîne de caractères) - mot qui va être testé
Entrées : `mot_a_trous` (chaîne de caractères) - un mot à trous
Sorties : Une booléen

pour *i qui varie entre 0 et la taille de mot* **faire**
  si *l'élément d'indice i de mot_a_trous n'est pas égal à l'élément d'indice i de mot et l'élément d'indice i de mot_a_trous n'est pas égal à '*'* **alors**
    retourner Faux
  retourner Vrai

...Écrire le code python définissant la fonction `correspond` ci-dessous...

```
def correspond(mot, mot_a_trous):
    for i in range(0, len(mot)):
        if mot_a_trous[i] != mot[i] and mot_a_trous[i] != '*':
            return False
    return True
```

```
def correspond(mot, mot_a_trous):
    """ Cette fonction permet de savoir si on peut remplacer
    convenablement les caractères * de mot_a_trous en
    obtenant mot. Il faut savoir que mot et mot_a_trous
    sont deux chaînes de caractères de même longueur.
    La fonction renvoie des booléens True ou False !!! """
    for i in range(0, len(mot)):
        if mot_a_trous[i] != mot[i] and mot_a_trous[i] != '*':
            return False
    return True
```

Exercice 4

Le tri par sélection

Question 4.1 Rappeler, sans coder, le principe du tri par sélection.

Le tri par sélection cherche le plus petit élément de la liste passée en argument et s'il est plus petit que le 1er élément de la liste, on permute. Une fois que ces deux éléments sont bien rangés, on cherche le 2e plus petit élément que l'on compare avec le 2e élément de la liste etc...

Question 4.2 On réalise un tri en place et par sélection de la liste `[28, 2, 34, 12, 16]`. Donner les 3 premières valeurs de l'état de liste lors de ce tri.

1re itération : [2, 28, 34, 12, 16]
2e itération : [2, 28, 12, 34, 16]
3e itération : [2, 12, 28, 34, 16]

Question 4.3 Combien d'opérations sont globalement nécessaires pour trier cette liste ?

- ☐ 10
- ☐ 15
- ☐ 20
- ☒ 25
- ☐ 50

Exercice 5

Épreuve pratique sujet 28 session 2021

On considère l'algorithme de tri de tableau suivant : à chaque étape, on parcourt depuis le début du tableau tous les éléments non rangés et on place en dernière position le plus grand élément.
Exemple avec le tableau : `t = [41, 55, 21, 18, 12, 6, 25]`

- Étape 1 : on parcourt tous les éléments du tableau, on permute le plus grand élément avec le dernier. Le tableau devient `t = [41, 25, 21, 18, 12, 6, 55]`
- Étape 2 : on parcourt tous les éléments sauf le dernier, on permute le plus grand élément trouvé avec l'avant-dernier. Le tableau devient : `t = [6, 25, 21, 18, 12, 41, 55]`

..et ainsi de suite. Le code de la fonction `tri_iteratif` qui implémente cet algorithme est donné ci-dessous. Complétez ce code.

```
def tri_iteratif(tab):
    for k in range(len(tab), 0 ,-1):
        imax = max(tab[k])
        for i in range(0, len(tab)-1):
            if tab[i] > tab[imax]:
                imax = i
        if tab[imax] > tab[i+1] :
            tab[a], tab[imax] = tab[imax], tab[i]
    return tab
```

Exercice 6

Épreuve pratique sujet 11 session 2023

La fonction `tri_insertion` suivante prend en argument une liste `tab` et trie cette liste en utilisant la méthode du tri par insertion. Complétez cette fonction pour qu'elle réponde à la spécification demandée.
On rappelle le principe du tri par insertion : on considère les éléments à trier un par un, le premier élément constituant, à lui tout seul, une liste triée de longueur 1. On range ensuite le second élément pour constituer une liste triée de longueur 2, puis on range le troisième élément pour avoir une liste triée de longueur 3 et ainsi de suite... À chaque étape, le premier élément de la sous-liste non triée est placé dans la sous-liste des éléments déjà triés de sorte que cette sous-liste demeure triée.
Le principe du tri par insertion est donc d'insérer à la n-ième itération, le n-ième élément à la bonne place.
En pratique voici un exemple détaillé sur la liste initiale `[5,2,9,1,5]`

- initialement, on suppose donc que le **5** à gauche constitue à lui tout seul une liste déjà triée (de 1 seul élément) soit `[5,2,9,1,5]`
- on cherche donc à positionner 'au bon endroit' le 1er élément non trié, cad le *2* (rangé dans la variable `valeur_insertion`). On le compare à l'élément à sa gauche... qui est 5. Comme il est plus petit que le 5 on fait 2 choses...
  1. on 'décale' le 5 à droite on obtient `[5,5,9,1,5]`
  2. Comme il n'y a plus d'élément encore à gauche, on place le 2 à sa place (donc en début de liste) et on obtient `[2,5,9,1,5]`
- Maintenant la partie bien triée est de longueur 2, et on cherche à insérer à la bonne place le premier élément non encore trié, à savoir le 9. Comme celui-ci est plus grand que le nombre se trouvant à sa gauche, il est bien placé. La liste devient : `[2, 5, 9,1,5]`
- On cherche à bien positionner le *1*.
  1. Comme le *1* est plus petit que le **9**, on décale le **9 sur sa droite** : `[2, 5, 9, 9,5]`
  2. Comme le *1* est plus petit que le **5**, on décale le **5 sur sa droite** : `[2, 5, 5, 9,5]`
  3. Comme le *1* est plus petit que le **2**, on décale le **2 sur sa droite** : `[2, 2, 5, 9,5]`
  4. Comme on a atteint le début de la liste, il n'est pas possible de décaler, c'est donc que le *1* se trouvera en début de liste... On obtient : `[1, 2, 5, 9,5]`
- On cherche à bien positionner le *5*.
  1. Comme le *5* est plus petit que le **9**, on décale le **9** vers la droite. La liste devient `[1, 2, 5, 9, 9]`
  2. Comme le *5* n'est pas plus petit que le **5** on s'arrête et on insère le *5* à sa place. La liste devient `[1, 2, 5, 5, 9]`

```
def tri_insertion(tab):
    n = len(tab)
    for i in range(1, n):
        valeur_insertion = tab[...]
        # la variable j sert à déterminer où placer la valeur à ranger
        j = ...
        # tant qu'on n'a pas trouvé la place de l'élément à insérer
        # on décale les valeurs du tableau vers la droite
        while j > ... and valeur_insertion < tab[...]:
            tab[j] = tab[j-1]
            j = ...
        tab[j] = ...
```