

TP – Sur une étagère

Exercice 1

Liste *T* :

indice	0 / -4	1 / -3	2 / -2	3 / -1
élément	1	5	7	9

a. `T[1]` retourne 5 ; `T[4]` retourne une erreur ("IndexError : list index out of range" = l'indice demandé est en dehors du domaine d'indices possibles) ; `T[-1]` retourne 9 ; `T[-4]` retourne 1.

Matrice *M* :

indice ligne \ indice colonne	0	1
	0	1
0	1	5
1	7	9

b. La valeur 7 est appelée avec `M[1][0]`.

c. La commande `print(M[1][0], M[0][0], M[0][1], M[1][1])` affiche "7, 1, 5, 9" :

d. La commande `print("M possède", len(M), " lignes et ", len(M[0]), " colonnes")` affiche les dimensions de *M*.

Exercice 2

Liste *mois* :

indice	0	1	2	3	4	5
élément	janvier	février	mars	avril	mai	juin

indice	6	7	8	9	10	11
élément	juillet	août	septembre	octobre	novembre	décembre

a. `mois[:6]` permet d'extraire uniquement les six premiers mois de l'année (l'indice 6 est exclus).

b. `mois[8:]` permet d'extraire uniquement les quatre derniers mois de l'année.

c. `mois[6:8]` permet d'extraire les deux mois d'été (l'indice 8 est exclus).

d. `mois[2:10]` permet d'extraire la période de mars à octobre inclus (l'indice 10 est exclus).

Exercice 3

Pour chaque élément de la liste *L_originale* donnée en argument, cet élément est mis au carré puis on soustrait 1 et on ajoute le résultat à la liste *L_modif*. La fonction renvoie donc : [0, -1, 0, 3]

Exercice 4

Note : Il existe plusieurs façons d'écrire ce code. Ceci n'est qu'un exemple.

```

a. 1 div_7 = [False]*1000          # liste de 1000 False
    2 for i in range(len(div_7)):    #lecture de la liste indice par indice
    3     if i%7 == 0:              #on met True si l'indice est divisible par 7
    4         div_7[i]=True
    5
b. 6 N = 0
    7 for elem in div_7:            #lecture de la liste élément par élément
    8     if elem == True:          #si l'indice (et donc le nombre) est divisible par 7
    9         N+=1                  #on ajoute 1 au compteur
   10 print(N)                      #résultat: 143 nombres divisibles par 7
   11
c. 12 k = 999                      #on part de la fin de la liste
   13 #on parcourt la liste tant que le nombre n'est pas divisible par 7
   14 while div_7[k] == False:
   15     k -= 1
   16 print(k)                      #résultat: 994 est le plus grand nombre <1000 divisible par 7

```

Exercice 5

Note : Il existe plusieurs façons d'écrire cette fonction. Ceci n'est qu'un exemple.

```
1 def Ordonnee(L):
2     for i in range(1,len(L)): #on lit le tableau à partir du 2ème indice
3         if L[i] < L[i-1]:      #si dans la liste, un élément est inférieur à l'élément précédent
4             return False      #on renvoie False
5     return True               #Si on arrive au bout de la boucle, on renvoie True
```

Exercice 6

Note : Il existe plusieurs façons d'écrire cette fonction. Ceci n'est qu'un exemple.

```
1 def Echange(L, i, j):
2     if i >= 0 and j >= 0:      #précondition
3         if i < len(L) and j < len(L): #Si les deux indices existent
4             temp = L[i]        #on sauvegarde la valeur d'indice i
5             L[i] = L[j]        #on remplace L[i] par L[j]
6             L[j] = temp        #on remplace L[j] par L[i]
7             return L
8     else: #si l'un des indices (ou les deux) n'existe pas, "Impossible"
9         print("Impossible")
```

Exercice 7

a. La commande renvoie dans l'ordre décroissant les carrés des nombres pairs entre 6 et 0 inclus, soit la liste [36, 16, 4, 0].

b. `L = [i**3 for i in range(1,16)]`

c. `L = [i**3 for i in range(1,16) if i**3%2 == 0]`

d. `L1 = [elem for elem in L if elem%3==0]`

Exercice 8

`[[k**i for k in range(1,6)] for i in range(1,6)]`

TP – Traitement de données

Exercice 9

Note : Il existe plusieurs façons d'écrire cette fonction. Ceci n'est qu'un exemple.

```
1 def Inv_liste(L):
2     L_inv = []
3     for i in range(len(L)-1, -1, -1): #on lit la liste L de la fin vers le début
4         L_inv += [L[i]]              #on ajoute les éléments de L à L_inv de la fin vers le début
5     return L_inv
```

Exercice 10

Note : Il existe plusieurs façons d'écrire cette fonction. Ceci n'est qu'un exemple.

```
1 def Tri_Comptage(L):
2     #initialisation de la matrice du nombre d'occurrences de chaque entier.
3     Fois = [[i,0] for i in range(101)] #le premier indice de la matrice correspond à l'entier.
4     for elem in L: #on lit la liste L élément par élément
5         Fois[elem][1] += 1 #on ajoute 1 au nombre d'occurrences de l'entier
6     return Fois
7
8 #Test de la fonction
9 from random import randint
10 k = randint(10,100) #taille de liste aléatoire entre 10 et 100
11 L = []              #Création de la liste test
12 for i in range(k):
13     L += [randint(0,100)]
14 print(L)
15 print(Tri_Comptage(L))
```

Exercice 11

Note : Il existe plusieurs façons d'écrire ces fonctions. Ceci n'est qu'un exemple.

```
a. 1 def TrouveL(L,v):
    2     indice = []
    3     for i in range(len(L)):      #on parcourt la liste indice par indice
    4         if L[i] == v:           #si l'élément est égal à v
    5             indice += [i]        #on enregistre l'indice
    6     #le nombre d'occurrences correspond à la taille de la liste d'indice
    7     print(len(indice), " occurrences")
    8     return indice

b. 1 def TrouveM(M,v):
    2     indice = []
    3     for i in range(len(M)):      #on parcourt la matrice ligne par ligne
    4         for j in range(len(M[0])): #puis colonne par colonne
    5             if M[i][j] == v:     #si l'élément est égal à v
    6                 indice += [[i,j]] #on enregistre les indices
    7     #le nombre d'occurrences correspond à la taille de la liste d'indice
    8     print(len(indice), " occurrences")
    9     return indice
```

Exercice 12

Note : Il existe plusieurs façons d'écrire cette fonction. Ceci n'est qu'un exemple.

```
1 def SousListe(L,L1):
2     #on regarde les blocs d'éléments de L de même taille que L1
3     for i in range(len(L)-len(L1)+1):
4         #si on trouve la sous-liste L1, on renvoie True
5         if L[i:i+len(L1)] == L1:
6             return True
```

TP – Manipuler le texte

Exercice 13

a. Une chaîne de caractère.

b. Un entier.

c. Une liste

Exercice 14

a. Ce code comporte une première boucle allant de $i = 0$ à $i = 9$. Pour chaque valeur du compteur i , on initialise une variable s de type chaîne de caractère, vide. On remplit alors s avec i symboles $*$, puis on l'affiche. Le dessin final est donné ci-contre.

```
b. 1 for i in range(10) :
    2     print("*" * i)      #on affiche i fois le symbole "*"

c. 1 for i in range(9, 0, -1) :
    2     print("*" * i)
```

```
*
**
***
****
*****
*****
*****
*****
*****
*****
```

Exercice 15

Note : Il existe plusieurs façons d'écrire cette fonction. Ceci n'est qu'un exemple.

```
1 def MotDePasse(mot):
2     alphabet = "abcdefghijklmnopqrstuvwxyz"
3     chiffre = "0123456789"
4     minu = 0
5     maj = 0
6     nbre = 0
7     special = 0
8     #on regarde chaque élément et on compte les minuscules, majuscules, chiffres
9     #et caractères spéciaux
```

```

10     for elem in mot:
11         if elem in alphabet:
12             minu += 1
13         if elem in alphabet.upper():
14             maj += 1
15         if elem in chiffre:
16             nbre += 1
17         if (elem not in alphabet) and (elem not in alphabet.upper()) and (elem not in chiffre):
18             special += 1
19     #On vérifie si toutes les conditions sont remplies
20     if len(mot) >= 8 and minu != 0 and maj != 0 and nbre != 0 and special != 0:
21         return True
22     else:
23         return False

```

Exercice 16

Note : Il existe plusieurs façons d'écrire cette fonction. Ceci n'est qu'un exemple.

```

1  def nb_mots(phrase):
2      N = 1      #le compteur est initialisé à 1 car la phrase a au moins 1 mot
3      #on regarde chaque élément indice par indice, en commençant par le 2eme caractère
4      #et en s'arrêtant à l'avant dernier (pour le compteur i)
5      for i in range(1,len(phrase)-1):
6          #si on a un espace suivi d'un caractère, c'est qu'on arrive à un nouveau mot
7          #(évite les problèmes d'espaces multiples)
8          if phrase[i]==" " and phrase[i+1]!=" ":
9              N+=1
10     return N

```

Exercice 17

Note : Il existe plusieurs façons d'écrire cette fonction. Ceci n'est qu'un exemple.

```

1  def anagramme(mot1,mot2):
2      #on ne vérifie que les chaines sont des anagrammes QUE si elles ont la
3      #même taille et qu'elles ne sont pas identiques
4      if len(mot1)==len(mot2) and mot1 != mot2:
5          test=""
6          #on met mot1 et mot2 en minuscules par sécurité
7          mot1 = mot1.lower()
8          mot2 = mot2.lower()
9          for elem1 in mot1:          #pour chaque caractère de mot1
10             i=0
11             while i < len(mot2):      #on regarde, par indice, les caractères de mot2
12                 if elem1 == mot2[i]:  #si le caractère existe dans mot2
13                     test += elem1     #on l'ajoute à la chaine test
14                     i = len(mot2)     #on sort de la boucle
15                 else:                #sinon, on regarde le caractère suivant
16                     i += 1
17             #si on retrouve mot1 dans test, mot1 et mot2 sont des anagrammes
18             if test == mot1:
19                 return True
20     #sinon, ce n'est pas le cas
21     else:
22         return False
23     else:
24         return False

```

Exercice 18

- a. Longueur = 4 (2 entiers, un tuple et une chaîne de caractères)
 - b. T[1] renvoie "S"
 - c. Elle renvoie ("Paul", 2008, "Axel", 2009).
 - d. Elle renvoie ("orange", 6, "orange", 6, "orange", 6).
 - e. T[0] == 3 renvoie True.
- T[0] = 3 renvoie une erreur ("TypeError: 'tuple' object does not support item assignment") car un tuple n'est pas mutable par affectation.

Exercice 19

- a. Par affectation multiple, a contient "Tom", b contient "Crouse", c contient 2002, d contient "1ere2".
- b. La fonction renvoie le tuple (None, None, None, None), sans message d'erreur.

Exercice 20

Note : Il existe plusieurs façons d'écrire ces fonctions. Ceci n'est qu'un exemple.

```

a. 1 def Date_valide(date):
    2     #Tableau du nombre de jours pour chaque mois
    3     JoursMois = ((1,31),(2,28),(3,31),(4,30),(5,31),(6,30),(7,31),(8,31),(9,30),(10,31),(11,30),(12,31))
    4     #Condition 3 chiffres
    5     if len(date) != 3:
    6         print("La date doit contenir 3 chiffres")
    7         return False
    8     #Condition: Jour et Mois positifs non-nul
    9     elif date[0] < 1 or date[1] < 1:
   10         print("Pas de chiffre négatifs ou nuls dans la date!")
   11         return False
   12     #Condition Mois inférieur ou égal à 12
   13     elif date[1] > 12:
   14         print("Le mois ",date[1]," est trop grand")
   15         return False
   16     #Condition 3: nombre de jour par mois
   17     #on utilise le tableau JoursMois, où l'indice pour le mois m est m-1
   18     elif date[0] > JoursMois[date[1]-1][1]:
   19         print("Le mois ",date[1]," a ",JoursMois[date[1]-1][1]," jours, donc ",date[0]," est trop grand")
   20         return False
   21     else:         #Sinon, la date est valide
   22         return True

b. 23
   24 def Nbre_jours(date1,date2):
   25     #Tableau du nombre de jours pour chaque mois
   26     JoursMois = ((1,31),(2,28),(3,31),(4,30),(5,31),(6,30),(7,31),(8,31),(9,30),(10,31),(11,30),(12,31))
   27     if Date_valide(date1) == True and Date_valide(date2) == True:         #Préconditions
   28         #Nombre de jours depuis J.-C. dans date1
   29         joursDate1 = date1[2]*365         #années
   30         for i in range(date1[1]-1):         #jours du 01/01 au début du mois (donc indice m-1)
   31             joursDate1 += JoursMois[i][1]
   32         joursDate1 += date1[0]         #jours depuis le début du mois
   33         #Nombre de jours depuis J.-C. dans date2
   34         joursDate2 = date2[2]*365
   35         for i in range(date2[1]-1):
   36             joursDate2 += JoursMois[i][1]
   37         joursDate2 += date2[0]
   38         #On renvoie le nombre de jours entre les 2 dates
   39         return joursDate2 - joursDate1
  
```

Exercice 21

Le code affiche les valeurs de la clef nom du dictionnaire voyage pour lesquels il y a coïncidence entre la valeur contenue dans la clef dest du dictionnaire voyages et la valeur contenue dans la clef dest du dictionnaire durees, si la valeur contenue dans la clef nbheures du dictionnaire durees est supérieure ou égale à 2 : il affichera donc "Riton".

Exercice 22

Note : Il existe plusieurs façons d'écrire cette fonction. Ceci n'est qu'un exemple.

```
1 def QuelPays(ville):
2     #Initialisation du dictionnaire
3     capitales = {"France": "Paris", "Angleterre": "Londres", "Espagne": "Madrid", "Portugal": "Lisbonne",
4                 "Allemagne": "Berlin", "Italie": "Rome", "Belgique": "Bruxelles", "Irlande": "Dublin"}
5     for elem in capitales:          #on lit le dictionnaire clef par clef
6         if capitales[elem] == ville: #si la valeur de la clef correspond à la ville
7             return elem             #on renvoie la clef (le pays)
8     return None                    #sinon, la ville n'est pas dans le dictionnaire
```

Exercice 23

Note : Il existe plusieurs façons d'écrire ces fonctions. Ceci n'est qu'un exemple.

```
a. 1 def Age(nom):
2     #Dictionnaire de base
3     personnes = { "Jean Aymar" : {"taille" : 178, "pays" : "USA", "age" : 31}, "Clio Patre" : {"pays" :
4                 "Portugal", "age" : 32, "taille" : 179 }}
5     if nom in personnes:          #Si le nom est une des clefs du dictionnaire
6         print(personnes[nom]["age"])      #on affiche l'age
```

```
b. 1 def Taille():
2     #Dictionnaire de base
3     personnes = { "Jean Aymar" : {"taille" : 178, "pays" : "USA", "age" : 31}, "Clio Patre" : {"pays" :
4                 "Portugal", "age" : 32, "taille" : 179 }}
5     taille = 0
6     for elem in personnes.values(): #on lit le dictionnaire par valeur
7         taille += elem["taille"]    #on additionne les valeurs des tailles
8     print(taille/len(personnes))    #on affiche la moyenne
```

Exercice 24

Note : Il existe plusieurs façons d'écrire ces codes et fonctions. Ceci n'est qu'un exemple.

a. « pizza » : 3 + 1 + 10 + 10 + 1 = 25 points

« whisky » : 10 + 4 + 1 + 1 + 10 + 10 = 36 points

« dedramatiser » : 2 + 1 + 2 + 1 + 1 + 2 + 1 + 1 + 1 + 1 + 1 + 1 = 15 points

```
b. 1 #Dictionnaire de base
2 Valeurs_Scrabbles = {10 : "kwxzy", 8 : "jq", 4 : "fhv", 3 : "bcp", 2 : "dmg"}
3 #Nouveau dictionnaire
4 Lettres_Scrabbles = {}
5 alphabet = "abcdefghijklmnopqrstuvwxyz"
6 for lettre in alphabet:          #pour chaque lettre de l'alphabet
7     #on crée une clef correspondante avec une valeur par défaut de 1
8     Lettres_Scrabbles[lettre] = 1
9     #si la valeur est en fait dans Valeurs_Scrabbles, on la modifie
10    for val in Valeurs_Scrabbles.keys():
11        if lettre in Valeurs_Scrabbles[val]:
12            Lettres_Scrabbles[lettre] = val
```

c. 1 | def Valeur(mot):
2 | score = 0
3 | for lettre in mot: #pour chaque lettre du mot
4 | score += Lettres_Scrabbles[lettre] #on ajoute les points de la lettre
5 | return score

Les 5 mots donnés sont des anagrammes : ils valent tous 7 points.

d. 1 | def TripleA4(mot):
2 | score = 0
3 | for i in range(len(mot)): #on lit le mot indice par indice
4 | if i != 3: #pour toutes les lettres sauf la 4ème
5 | score += Lettres_Scrabbles[mot[i]] #on ajoute les points de la lettre
6 | else:
7 | score += 3*Lettres_Scrabbles[mot[i]] #on ajoute 3 fois les points de la lettre
8 | return score

Le mot « trope » rapporte alors 13 points, alors que les autres ne rapportent que 9 points.