Tuple et liste : Exercices

Capacités attendues :

p-uplets.	Écrire une fonction renvoyant un p-uplet de valeurs.
Tableau indexé, tableau donné en compréhension	Lire et modifier les éléments d'un tableau grâce à leurs index. Construire un tableau par compréhension. Utiliser des tableaux de tableaux pour représenter des matrices : notation a [i] [j]. Itérer sur les éléments d'un tableau.

Note : p-uplet est un autre terme pour désigner un tuple.

Note 2 : Un tableau est une séquence de taille fixe et indexée d'éléments de même type. Le type Python associé au tableau est le type Liste. Ici, on confondra les 2 termes.

Exercice 1 (p-uplets)

On considère la suite d'instructions donnée ci-contre. Quelles sont les valeurs affectées aux variables a, b et c à la fin de cette séquence d'instructions ?

```
tuple1 = (19, -2.2, 888)
tuple2 = ("Mlle", "Mme", "M.")
a, b, c = tuple1
c, a = a, c
d, e, f = tuple2
(b, a) = (f, e)
```

Exercice 2 (p-uplets)

On rappelle que la fonction randint(a, b) du module random retourne un nombre entier aléatoire compris entre a et b (b compris).

```
import random

def lancer_trois_des():
    ...
    ...
```

1) Écrire le prototype d'une fonction `lancer_trois_des` permettant de retourner un 3-tuple de 3 nombres entiers simulant le lancer de 3 dés cubiques.

2) Compléter le code de cette fonction dans l'encadré ci-contre.

3) On appelle cette fonction avec l'instruction suivante :
   `mon_lancer = lancer_trois_des()`

   Quel est le type de la variable `mon_lancer` ?

   Quelle instruction permet alors de calculer la somme des trois dés ?
   a) `somme = mon_lancer[1] + mon_lancer[2] + mon_lancer[3]`
   b) `somme = mon_lancer[0] + mon_lancer[1] + mon_lancer[2]`
   c) `somme = mon_lancer + mon_lancer + mon_lancer`

4) On appelle cette fonction avec l'instruction suivante :
   `de_1, de_2, de_3 = lancer_trois_des()`

   Cette instruction génère-t-elle une erreur ? Si oui, pourquoi ? Si non, donner une instruction permettant d'affecter à une variable la somme des trois dés.

Exercice 3 (p-uplets)

On considère les deux instructions ci-dessous en langage python. Lors de l'exécution elles provoquent une erreur.

```
>>> debut_alphabet = ('a', 'b', 'c', 'd', 'd', 'f', 'g', 'h', 'i', 'j')
>>> debut_alphabet[4] = 'e'
---------------------------------------------------------------------------
TypeError                                 Traceback (most recent call last)
<ipython-input-7-131a7cf1d918> in <module>()
      1 tuple1 = ('a', 'b', 'c', 'd', 'd', 'f', 'g', 'h', 'i', 'j')
----> 2 tuple1[4] = 'e'

TypeError: 'tuple' object does not support item assignment
```

Pourquoi ?

Exercice 4 (Lire et modfier les éléments d'un tableau)

On considère la suite d'instructions données ci-contre.

Donner l'état des deux tableaux à la fin de la suite d'instructions.

```
tab_x = [7, 77, 777, 7777]
tab_y = [5, 55, 555, 5555]
tab_x[2] = tab_y[1]
tab_y[3] = tab_x[0]
tab_x[1] = tab_x[2]
tab_y[1] = tab_y[3]
```

Exercice 4-bis (Lire et modfier les éléments d'un tableau)

On considère la suite d'instructions donnée ci-contre.

Donner les valeurs de chacune des variables de a à f.

```
tab = ['z', 'yy', 'xxx', 'wwww', 'paf']

a = len(tab)
b = tab[ len(tab)-1 ]
c = tab[-1]
d = tab[ len(tab)-2 ]
e = tab[-2]
f = tab[-3]
```

Exercice 5 (tableaux construits par compréhension)

Pour chacune des instructions ci-dessous, écrire le tableau qui a été créé.
On rappelle que :

♦ `x**y` calcule x à la puissance y

♦ `x%y` calcule x modulo y (c'est-à-dire le reste de la division euclidienne de x par y)

♦ `x//y` calcule le quotient de la division euclidienne de x par y

```
>>> 2**4
16
>>> 97 % 10
7
>>> 97 // 10
9
>>> 25 % 2
1
>>> 25 // 2
12
```

```
tab_a = [ 2**x for x in range(11) ]

tab_b = [ 7 * x%2 for x in range(11) ]

tab_c = [ 7 * ((10**x) // 9) for x in range(1, 5) ]

tab_d = [ 'M. ' + car + ' ?' for car in 'XYZ' ]
```

Exercice 6 (Itérer sur les éléments d'un tableau)

On rappelle que la fonction `len()` permet de retourner la longueur d'un tableau mais aussi la longueur d'une chaîne de caractères.

```
>>> len("Coucou")
6
>>> len("Pardon ?")
8
```

1. Combien de noms vont être affichés à l'issue de la séquence d'instructions ci-dessous ?

```
grands_noms = ["Lovelace", "Clarke", "Goldstine", "Hopper", "Recoque", "Hamilton"]
for nom in grands_noms:
    if "o" in nom:
        print(nom)
```

2. Quelle sera la valeur de `quantite_mystere` à la fin de la sequence d'instructions ci-dessous ?

```
grands_noms = ["Lovelace", "Clarke", "Goldstine", "Hopper", "Recoque", "Hamilton"]
quantite_mystere = 0
for nom in grands_noms:
    quantite_mystere = quantite_mystere + len(nom)
```

REMARQUE : le parcours séquentiel de tableau sera approfondi avec quelques algorithmes standard, en particulier le parcours de tableaux par indice (ici on itère sur les éléments)

Exercice 7 (Utiliser des tableaux de tableaux pour représenter des matrices)

On représente la matrice donnée ci-contre grâce au tableau ma_belle_matrice.

'a'	'b'	'c'	'd'
'e'	'g'	'h	'i'
'j'	'k'	'l'	'm'
'n'	'o'	'p'	'q'
'r'	's'	't'	'u'

On remarque au passage que lorsqu'on définit des matrices en python, il est possible de faire des passages à la ligne après les virgules. On obtient ainsi visuellement quelque chose de très clair.

```
>>> ma_belle_matrice = [ ['a', 'b', 'c', 'd'],
                         ['e', 'f', 'g', 'h'],
                         ['i', 'j', 'k', 'l'],
                         ['m', 'n', 'o', 'p'],
                         ['q', 'r', 's', 't'] ]

>>> a = ma_belle_matrice[3]
>>> b = ma_belle_matrice[3][2]
>>> c = ma_belle_matrice[1][2]
>>> d = ma_belle_matrice[2][1]
```

1) Donner les valeurs de chacune des variables a, b, c et d

2) Proposer une suite d'instructions permettant de modifier ma_belle _matrice afin qu'elle corresponde à la matrice donnée ci-contre.

'a'	'b'	**'X'**	'd'
'X'	**'X'**	**'X'**	**'X'**
'j'	'k'	**'X'**	'm'
'n'	'o'	**'X'**	'q'
'r'	's'	**'X'**	'u'

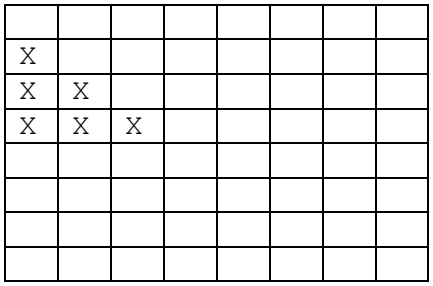
Exercice 8 (Doubles boucles et coloriages)

Dans certains algorithmes (de tri par exemple) vous aurez à faire à des doubles boucles.
Pour parcourir une matrice il faut également recourir à une double boucle.

Dans tout cet exercice et pour chaque cas, M est initialement une matrice de taille 8 x 8 dont tous les éléments sont égaux à ' ' (chaîne de caractères vide).
Pour chacune des doubles boucles données, vous indiquerez l'état de la matrice M à la fin de la double boucle en écrivant des 'X' là où il faut.
Pour vous aider le premier cas a été commencé. On rappelle qu'avec la notation `M[i][j]`, `i` indique la ligne et `j` la colonne (en respectant la "tradition" `M = [ [ .. ligne0 .. ], [ .. ligne1 .. ], [ .. ligne2 .. ], ..]`)

Cas numéro 1

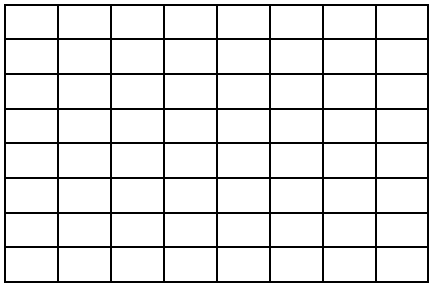
```
for i in range(0, 8):
    for j in range(0, i):
        M[i][j] = 'X'
```



Pour i=0, j va aller de 0 à .. euh … nulle part.
Pour i= 1, j va aller de 0 à … 0.
Pour i=2, j va aller de 0 à … 1.
Pour i=3, j va aller de 0 à … 2.

Cas numéro 2

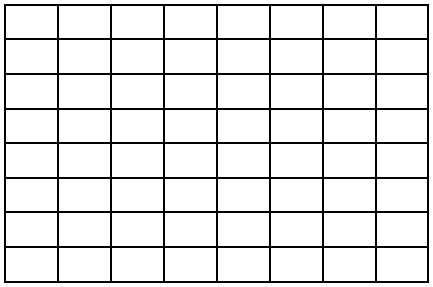
```
for i in range(0, 8):
    for j in range(i+1, 8):
        M[i][j] = 'X'
```



Pour i=0, j va aller de … à ….
Pour i= 1, j va aller de … à ….
Pour i=2, j va aller de … à ….
Pour i=3, j va aller de … à …..

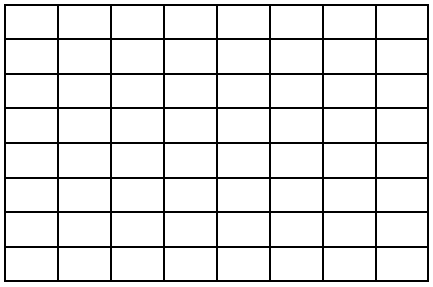
Cas numéro 3

```
for i in range(1, 8):
    for j in range(0, 8-i):
        M[i][j] = 'X'
```



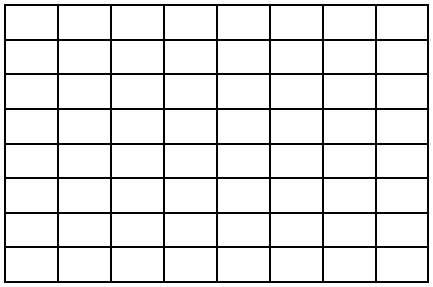
Cas numéro 4

```
for i in range(0, 8):
    for j in range(3, 5):
        M[i][j] = 'X'
```



Cas numéro 5

```
for i in range(3, 5):
    for j in range(0, 8):
        M[i][j] = 'X'
```



Cas numéro 6

```
for i in range(1, 7):
    for j in range(8-i, 8):
        M[i][j] = 'X'
```