

RND. Types construits (uplets, listes). Exercices

Exercice 1 : Vrai / Faux

	VRAI	FAUX
Le n-uplet (1,2,(3,4),5) a pour longueur 5.		
Si $t = 'a', 'b', 'd'$ alors l'expression $t[1] = 'v'$ provoque une erreur.		
L'expression $3*(1,2)$ a pour valeur (3,6).		
Soit L la liste [1,3,5]. Après l'instruction $L.append([4,6])$, la liste L a pour valeur $L = [1,3,5,4,6]$.		
Soit L une liste de nombres. Pour modifier le dernier élément de la liste L, on peut écrire $L[len(L)] = ...$.		
Soit L la liste [[1,2,3], [4,5,6], [7,8,9]]. La valeur de $L[1][2]$ est 6.		

Exercice 2 : QCM

Pour chaque question, une seule réponse est correcte parmi les quatre proposées.

Question 1 : On considère le n-uplet $t = (3, 5, 1)$. Qu'obtient-on après l'instruction $t[1] = 4$?

1. La valeur de t est $t = (4, 5, 1)$.
2. La valeur de t est $t = (3, 4, 1)$.
3. La valeur de t est $t = (3, 5, 4)$.
4. Une erreur.

Question 2 : On considère l'instruction $a = 0,2+0,3 == 0,5$. Quelle est la valeur de a ?

1. *False*, c'est lié à la représentation en nombres flottants.
2. *True*, cela revient à écrire $a = (0,2 + 0,3 == 0,5)$.
3. $(0,2, False, 5)$, a est du type tuple.
4. L'écriture de l'instruction provoque une erreur.

Question 3 : On dispose d'une liste $L = [15, 17, 12, 23]$. Après l'instruction $L[2] = 25$, que vaut L ?

1. $L = [15, 17, 25, 23]$.
2. $L = [15, 25, 12, 23]$.
3. $L = [15, 17, 25, 12, 23]$.
4. $L = [15, 25, 17, 12, 23]$.

Question 4 : Après l'instruction $L = [[i, i+1] \text{ for } i \text{ in range}(2)]$, la valeur de L est :

1. $L = [[0,1], [1,2]]$.
2. $L = [[1,2], [2,3]]$.
3. $L = [0, 1, 1, 2]$.
4. $L = [1, 2, 2, 3]$.

Question 5 : On construit une matrice avec le code suivant :

```
matrice = [3*[0] for i in range(3)]

for i in range(3) :
    matrice[i][i] = i + 1
    matrice[0][i] = matrice[0][i] + i + 1
    matrice[i][2] = matrice[i][2] + i + 1
```

Quel est le résultat obtenu ?

1. [[1, 2, 4], [0, 2, 2], [0, 0, 6]].
2. [[2, 2, 4], [0, 2, 2], [0, 0, 4]].
3. [[1, 2, 4], [0, 2, 4], [0, 0, 6]].
4. [[2, 2, 4], [0, 2, 2], [0, 0, 6]].

A noter : Pour les exercices suivants, on pourra vérifier les réponses en programmant les algorithmes sur « Jupyter »

Exercice 3 :

Que **fait** le programme suivant ? On ne considèrera que des listes de nombres.

```
def IDoIt(liste : list) -> bool :      # Indiquer le type des arguments
    for i in range(1, len(liste)) :    # et ceux du retour aide à comprendre
        if liste[i-1] > liste[i] :    # la fonction
            return False
    return True
```

Exercice 4 :

Ecrire un programme qui prend en **argument** une **liste de 10 nombres** composée d'entiers et qui **renvoie** une **liste** ne contenant que des **nombres pairs**.

Exercice 5 :

Quelle est la **valeur** de *couples* à la fin de l'exécution du programme suivant ?

```
lettres = ['a', 'b', 'c']
nombres = [1, 6]
couples = [(c, n) for c in lettres for n in nombres]
```

Exercice 6 : Constructions de listes

1/ **Ecrire** une fonction multiples1 qui prend en paramètre un entier naturel non nul n et qui renvoie la liste des dix premiers multiples non nuls de n .

2/ **Ecrire** une fonction multiples2 qui prend en paramètre un entier naturel non nul n et qui renvoie la liste des multiples non nuls de n strictement inférieurs à 1000.

3/ **Ecrire** une fonction diviseurs qui prend en paramètre un entier naturel non nul n et qui renvoie la liste des diviseurs de n .

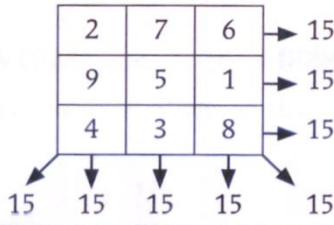
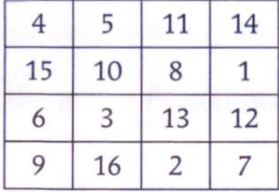
Exercice 7 : Carrés magiques

Un carré d'ordre n est un tableau carré contenant n^2 entiers strictement positifs.

On dit qu'un carré est **magique** si :

- Il contient tous les nombres entiers de 1 à n^2 .
- Les sommes des nombres de chaque rangée, les sommes des nombres de chaque colonne et les sommes des nombres de chaque diagonale principale sont égales.

On modélise un carré par une **matrice**, c'est-à-dire une **liste de listes de nombres**.

Carré d'ordre n	Modélisation proposée
	<pre>carre3 = [[2, 7, 6], [9, 5, 1], [4, 3, 8]]</pre>
	<pre>carre4 = [[4, 5, 11, 14], [15, 10, 8, 1], [6, 3, 13, 12], [9, 16, 2, 7]]</pre>

1/ Lecture de liste de listes :

- Quelle est la **longueur** de carre4 ?
- Que **vaut** carre3[1] ?
- Que **vaut** carre3[0][2] ?
- Quelle **instruction** permet de récupérer la valeur 3 de carre4 ?

2/ On propose le code suivant :

```
def sommeLigne(carre,n) :  
    somme = 0  
    for nombre in carre[n] :  
        somme = somme + nombre  
    return somme
```

Que **vaut** sommeLigne(carre4,2) ? A quoi **sert** cette **fonction** ?

3/ **Ecrire** une fonction qui prend un carré en paramètre et qui vérifie que **la somme des nombres de chaque ligne** est **égale**.

A faire et noté :

4/ (*) : **Proposer** le code d'un programme qui vérifie si un carré est **magique** ou non. **Vérifier** avec un exemple qui fonctionne (un des exemples ci-dessus) et un qui ne fonctionne pas à trouver.