RND. Types Construits (uplets, listes). Exercices. Corrigés

***Exercice 1* : Vrai / Faux**

Q.1 : FAUX. Il y a 4 éléments dont un tuple (3,4).
Q.2 : VRAI. Un tuple n'est pas mutable.
Q.3 : FAUX. La valeur sera (1,2) , (1,2) , (1,2).
Q.4 : FAUX. La méthode append() ajoute un élément, ici c'est la liste [4,5] : L = [1,2,3, [4,5]].
Q.5 : FAUX. Il faut écrire L[len(L) – 1] ou L[-1].
Q.6 : VRAI. On demande l'élément à la deuxième ligne et troisième colonne.

***Exercice 2* : QCM**

Q.1 : Réponse 4. Un n-uplet est non modifiable.
Q.2 : Réponse 3. A est bien un n-uplet (attention, il faut utiliser le « . » pour les nombres décimaux !).
Q.3 : Réponse 1. On remplace le troisième élément de la liste par 25.
Q.4 : Réponse 1. On construit une liste de listes de deux éléments, « i » commençant à 0.
Q.5 : Réponse 4. Il faut lire les instructions une par une.

***Exercice 3* :**

Le programme vérifie si une liste est triée dans l'ordre croissant ou pas.

```
def IDoIt(liste : list) -> bool :
    for i in range(1,len(liste)) :
        if liste[i - 1] > liste[i] :
            return False

    return True

print(IDoIt([-2,0,2,5,6])) # Liste triée dans l'ordre croissant
print(IDoIt([2,5,7,8,4]))  # Liste non triée
```

```
True
False
```

***Exercice 4* :**

Un exemple de programme

```
listeInitiale = [1,3,5,7,4,9,18,-6,4,0]  # Exemple de liste

listePairs = [val for val in listeInitiale if not val%2] # Liste composée de nombres pairs
print(listePairs)
```

```
[4, 18, -6, 4, 0]
```

***Exercice 5* :**

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

print(couples)
```

```
[('a', 1), ('a', 6), ('b', 1), ('b', 6), ('c', 1), ('c', 6)]
```

***Exercice 6* : Construction de listes**

Exemples de programmes

```
def multiples1(n) :
    liste = [i*n for i in range(1,11)]
    return liste

def multiples2(n) :
    liste = []
    i = 1
    while i*n < 1000 :
        liste.append(i*n)
        i = i + 1
    return liste

def diviseurs(n) :
    liste = [d for d in range(1,n+1) if not n%d]
    return liste

print(multiples1(12))
print(multiples2(12))
print(diviseurs(12))
```

```
[12, 24, 36, 48, 60, 72, 84, 96, 108, 120]
[12, 24, 36, 48, 60, 72, 84, 96, 108, 120, 132, 144, 156, 168, 180, 192, 204, 216, 228, 240, 252,
336, 348, 360, 372, 384, 396, 408, 420, 432, 444, 456, 468, 480, 492, 504, 516, 528, 540, 552, 56
6, 648, 660, 672, 684, 696, 708, 720, 732, 744, 756, 768, 780, 792, 804, 816, 828, 840, 852, 864,
948, 960, 972, 984, 996]
[1, 2, 3, 4, 6, 12]
```

***Exercice 7* : Carré magique**

1/ a) La longueur vaut 4 (4 listes).

b) carre3[1] = [9,5,1]

c) carre3[0][2] = 6

d) carre4[2][1] = 3

2/ Cette fonction calcule la somme des valeurs de la n + 1 nième ligne de carre4. Le résultat est 34 (6 + 3 + 13 + 12).

3/ Un exemple de programme

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

def compareLigne(carre) :
    somme = sommeLigne(carre,0) # On récupère la somme de la première
    for i in range(1,len(carre)) :
        if sommeLigne(carre,i) != somme : # On la compare à celle des autres lignes
            return False
    return True

carre3 = [ [2,7,6] , [9,5,1] , [4,3,8] ]
print(compareLigne(carre3))
```

```
True
```