

## Exercice 1

1. Parmi les extraits de programme suivants, lesquels permettent de construire la liste des cinq premiers nombres impairs ?

☐ a.

```
impairs = [1, 3, 5, 7, 9]
```

☐ b.

```
impairs = []  
for n in range(5):  
    impairs.append(2 * n + 1)
```

☐ c.

```
impairs = [2 * n + 1 for n in range(5)]
```

☐ d.

```
impairs = [n for n in range(1, 11, 2)]
```

☐ e.

```
impairs = []  
n = 0  
while len(impairs) != 5:  
    if n % 2 == 1:  
        impairs.append(n)  
    n = n + 1
```

Tous les programmes permettent de construire la liste des cinq premiers nombres impairs.

## Exercice 2

2. Donner plusieurs programmes permettant de construire la liste des 25 premiers nombres pairs.

Exercice n°2:

```
2) a) nbImpairs = []  
for i in range(1, 50, 2):  
    nbImpairs += [i]  
print(nbImpairs)
```

```
b) nbImpairs = []  
for i in range(1, 50):  
    if i % 2 == 1:  
        nbImpairs += [i]  
print(nbImpairs)
```

Exercice n° 3:

1)

```
def minimum(listes):  
    mini = listes[0]  
    nb = 0  
    for i in listes:  
        nb = i  
        if nb < mini:  
            mini = nb  
    return mini  
print(minimum(test))
```

2)

```
def somme(listes):  
    total = 0  
    for i in range (len(listes)):  
        total += listes[i]  
    return total  
print(somme(test))
```

3)

```
def moyenne(listes):  
    total = 0  
    moyenne = 0  
    for i in range(len(listes)):  
        total += listes[i]  
    moyenne = total / len(listes)  
    return(moyenne)  
print(moyenne(test))
```

## Exercice 5

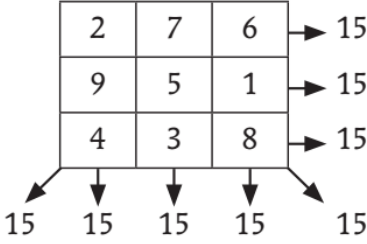
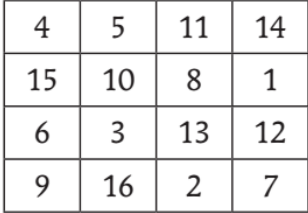
Un carré d'ordre  $n$  est un tableau carré contenant  $n^2$  entiers strictement positifs.

On dit qu'un carré d'ordre  $n$  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 liste de listes de nombres.

Exemples :

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. a. Quelle est la valeur de `len(carre4)` ?
- b. Quelle est la valeur de `carre3[1]` ?
- c. Quelle est la valeur de `carre3[0][2]` ?
- d. Quelle instruction permet de récupérer la valeur 3 de `carre4` ?
2. a. On propose le code suivant :

```
def somme_ligne(carre, n):
    """
    carre est un tableau carré de nombres
```

- 1) a) `len(carre4) = 4`
- b) `carre3[1] = [9, 5, 1]`
- c) `carre3[0][2] = 6`
- d) `x = carre4[3]`

```

n est un nombre entier
"""
somme = 0
for nombre in carre[n]:
    somme = somme + nombre
return somme

```

Que vaut `somme_ligne(carre4, 2)` ?

À quoi sert cette fonction ?

**b.** Écrire le code d'une fonction qui prend un carré en paramètre et qui vérifie que les sommes des nombres de chaque ligne sont égales.

**c.** Proposer le code d'une fonction qui prend un carré en paramètre, ainsi que le numéro d'une colonne, et qui renvoie la somme des nombres de cette colonne.



#### À NOTER

Pour aller plus loin, écrivez une fonction `est_magique()` qui prend un carré en paramètre et renvoie `True` si le carré est bien magique et `False` sinon.

2) a) somme = 34

Elle sert à calculer la somme des nombres d'une ligne

**b)** `print("Exercice 5")`

`print("Question b")`

```
def somme_ligne(carre, n):
```

```
    somme = 0
```

```
    for nombre in carre[n]:
```

```
        somme += nombre
```

```
    return somme
```

```
carre3 = [[2, 7, 6], [9, 5, 1], [4, 3, 8]]
```

```
carre4 = [[4, 5, 11, 14], [15, 10, 8, 1], [6, 3, 13, 12], [9, 16, 2, 7]]
```

```
total = []
```

```
for n in range (len(carre4)):
```

```
    total += [somme_ligne(carre4,n)]
```

```
if total[0] == total[1] and total[1] == total[2] and total[2] == total[3]:
```

```
    print("C'est un carré parfait")
```

```
else:
```

```
    print("Ce n'est pas un carré parfait")
```

```
c) print("Question c")
```

```
def somme_colonne(carre, n, c):
```

```
    somme = carre[n][c]
```

```
    return somme
```

```
sommeColonne = 0
```

```
total = []
```

```
for n in range (len(carre4)):
```

```
    for i in range (len(carre4[n])):
```

```
        sommeColonne += somme_colonne(carre4, i, n)
```

```
    total += [sommeColonne]
```

```
    sommeColonne = 0
```

```
if total[0] == total[1] and total[1] == total[2] and total[2] == total[3]:
```

```
    print("C'est un carré parfait")
```

```
else:
```

```
    print("Ce n'est pas un carré parfait")
```