# INFO-H-100 - Programmation
# TP 6 — Boucles `for` et listes

Lorsque l'exercice demande d'écrire une fonction, écrivez la fonction demandée et testez-la avec plusieurs valeurs pertinentes.

**Ex. 1.** Ecrire une fonction qui calcule la somme des carrés d'une liste de nombres.

```
print sommeCarres([2, 3, 5, 6])          #affiche 74
print sommeCarres([4, 6, 7, 2, 3])       #affiche 114
```

**Ex. 2.** Ecrire une fonction qui trouve le minimum dans une liste de nombres. Si la liste est vide, votre fonction exécutera le code suivant qui déclenche une exception :

```
raise Exception("empty list")
```

```
print minimumListe([4, 3, 5, 6, 7])      #affiche 3
print minimumListe([3, 12, 3, 2, 10])    #affiche 2
```

**Ex. 3.** Ecrire une fonction qui trouve le nom de la personne la plus âgée dans une liste de tuples (nom, âge).

```
print doyen([('Thomas', 20), ('Philippe', 24), ('Axel', 22)])     #affiche Philippe
print doyen([('Jean', 34), ('Laurent', 27), ('Michel', 33)])      #affiche Jean
```

**Ex. 4.** Ecrire une fonction qui remplace tous les nombres d'une liste par leur valeur absolue.

```
li1 = [3, -4, -5, 6, -4]
remplaceAbs(li1)
print li1                    #affiche [3, 4, 5, 6, 4]
```

**Ex. 5.** Ecrire une fonction qui échange deux éléments d'une liste en fonction de leurs indices.

```
li1 = [3, -4, -5, 6, -4]
echange(li1, 0, 3)
print li1                #affiche [6, -4, -5, 3, -4]
```

**Ex. 6.** Ecrire une fonction qui mélange les éléments d'une liste : pour chaque élément de la liste, l'échanger avec lui même ou un autre élément suivant choisi au hasard. Pour rappel, la fonction `randint(a,b)` du module `random` renvoie un entier choisi aléatoirement dans $[a, b]$.

```
li1 = [7, -8, 12, -44, 5]
melange(li1)
print li1   #affiche par exemple [5, -44, 7, -8, 12]
```

**Ex. 7.** Ecrire une fonction qui produit une liste ordonnée contenant les $n$ premières puissances de 2.

```
print powerOf2(4)   #affiche [2, 4, 8, 16]
print powerOf2(10)  #affiche [2, 4, 8, 16, 32, 64, 128, 256, 512, 1024]
```

**Ex. 8.** Ecrire une fonction qui produit une liste ordonnée contenant les nombres premiers inférieurs à un maximum donné.

```
print nombresPremiers(17)   #affiche [2, 3, 5, 7, 11, 13]
print nombresPremiers(50)   #affiche [2, 3, 5, 7, 11, 13, 17, 19, 23, 29, 31, 37, 41, 43, 47]
```

**Ex. 9.** Ecrire une fonction qui affiche une liste `[1,2,3]` de la manière suivante : `[1 -> 2 -> 3]`.

```
myprint([11, -2, 35, -46, 7])          #affiche [11 -> -2 -> 35 -> -46 -> 7]
myprint([1, 2, 3, 5, 7, 11, 13, 17])   #affiche [1 -> 2 -> 3 -> 5 -> 7 -> 11 -> 13 -> 17]
```

**Ex. 10.** Ecrire une fonction qui renvoie la plus longue liste d'une liste de listes.

```
print plusLongueListe([[11, -2, 35, -46, 7],[2,1],[14, -2, 3, 1]])   #affiche [11, -2, 35, -46, 7]
print plusLongueListe([[4, -45, 1, 3], [1, 4], [7, -8, 12, -44, 5]]) #affiche [7, -8, 12, -44, 5]
```

**Ex. 11.** Ecrire une fonction qui génère la liste suivante en fonction de $n$. Exemple pour $n = 3$ :

```
[[1], [1, 2], [1, 2, 3]]
```

```
print listeTriangle(2)  #affiche [[1], [1, 2]]
print listeTriangle(4)  #affiche [[1], [1, 2], [1, 2, 3], [1, 2, 3, 4]]
print listeTriangle(0)  #affiche []
```

# INFO-H-100 - Programmation
# TP 6 — Boucles `for` et listes
Corrections

---

## Solution de l'exercice 1:

```
def sommeCarres(listeNombres):
        somme = 0
        for nombre in listeNombres:
                somme += nombre ** 2
        return somme

print sommeCarres([2, 3, 5, 6])          #affiche 74
print sommeCarres([4, 6, 7, 2, 3])       #affiche 114
```

## Solution de l'exercice 2:

```
def minimumListe(listeNombres):
        if (len(listeNombres) == 0):
                raise Exception("empty list")

        minimum = listeNombres[0]
        for nombre in listeNombres:
                if nombre < minimum :
                        minimum = nombre
        return minimum

print minimumListe([4, 3, 5, 6, 7])       #affiche 3
print minimumListe([3, 12, 3, 2, 10])     #affiche 2
```

Solution qui permet de gagner un tour de boucle :

```
def minimumListe(listeNombres):
        if (len(listeNombres) == 0):
                raise Exception("empty list")

        minimum = listeNombres[0]
        for nombre in listeNombres[1:]:
                if nombre < minimum :
                        minimum = nombre
        return minimum

print minimumListe([4, 3, 5, 6, 7])       #affiche 3
print minimumListe([3, 12, 3, 2, 10])     #affiche 2
```

## Solution de l'exercice 3:

```
def doyen(listeNoms):
    if (len(listeNoms) == 0):
        raise Exception("empty list")

    (nomDoyen, ageDoyen) = listeNoms[0]
    for (nom,age) in listeNoms[1:]:
        if age > ageDoyen:
            (nomDoyen, ageDoyen) = (nom, age)

    return nomDoyen

print doyen([('Thomas', 20), ('Philippe', 24), ('Axel', 22)])      #affiche Philippe
print doyen([('Jean', 34), ('Laurent', 27), ('Michel', 33)])       #affiche Jean
```

Autre solution :

```
def doyen(listeNoms):
    if (len(listeNoms) == 0):
        raise Exception("empty list")

    iMax = listeNoms[0]
    for i in range(1,len(listeNoms)):
        if listeNoms[i][1] > iMax[1]:
```

```
            iMax = listeNoms[i]

    return iMax[0]
```

## Solution de l'exercice 4:

```
def remplaceAbs(listeNombres):
        for i in range(len(listeNombres)):
                if listeNombres[i] < 0:
                        listeNombres[i] = -listeNombres[i]

li1 = [3, -4, -5, 6, -4]
remplaceAbs(li1)
print li1                   #affiche [3, 4, 5, 6, 4]
```

## Solution de l'exercice 5:

```
def echange(liste, i1, i2):
    liste[i1], liste[i2] = liste[i2], liste[i1]

li1 = [3, -4, -5, 6, -4]
echange(li1, 0, 3)
print li1               #affiche [6, -4, -5, 3, -4]
```

## Solution de l'exercice 6:

```
import random

def echange(liste, i1, i2):
    liste[i1], liste[i2] = liste[i2], liste[i1]

def melange(listeNombres):
    for i in range(len(listeNombres)-1):
        place = random.randint(i,len(listeNombres)-1)
        echange(listeNombres,i,place)


li1 = [7, -8, 12, -44, 5]
melange(li1)
print li1   #affiche par exemple [5, -44, 7, -8, 12]
```

## Solution de l'exercice 7:

```
def powerOf2(n):
    li = []
    power = 2
    for i in range(n):
        li.append(power)
        power = power*2

    return li

print powerOf2(4)   #affiche [2, 4, 8, 16]
print powerOf2(10)  #affiche [2, 4, 8, 16, 32, 64, 128, 256, 512, 1024]
```

## Solution de l'exercice 8:

```
import math

def estPremier(nombre):
    if(nombre < 2):
        return False
    for i in range(2, nombre):
        if (nombre % i == 0):
            return False
    return True

def nombresPremiers(max):
    listePremiers = []
    for i in range(max):
        if (estPremier(i)):
            listePremiers.append(i)
```

```
    return listePremiers

print nombresPremiers(17)    #affiche [2, 3, 5, 7, 11, 13]
print nombresPremiers(50)    #affiche [2, 3, 5, 7, 11, 13, 17, 19, 23, 29, 31, 37, 41, 43, 47]
```

Autre version :

```
def estPremier(nombre):
    if(nombre < 2):
        return False
    for i in range(2, int(math.sqrt(nombre))+1):
        if (nombre % i == 0):
            return False
    return True
```

## Solution de l'exercice 9:

```
def myprint(li):
        myString = '[' + str(li[0])
        for i in range(1,len(li)):
                myString += ' -> ' + str(li[i])

        myString = myString + ']'
        print myString

myprint([11, -2, 35, -46, 7])                #affiche [11 -> -2 -> 35 -> -46 -> 7]
myprint([1, 2, 3, 5, 7, 11, 13, 17])         #affiche [1 -> 2 -> 3 -> 5 -> 7 -> 11 -> 13 -> 17]
```

## Solution de l'exercice 10:

```
def plusLongueListe(listes):
    if (len(listes) == 0):
        raise Exception("empty list")

    longueListe = listes[0]
    for liste in listes[1:]:
        if len(liste) > len(longueListe):
            longueListe = liste

    return longueListe

print plusLongueListe([[11, -2, 35, -46, 7],[2,1],[14, -2, 3, 1]])   #affiche [11, -2, 35, -46, 7]
print plusLongueListe([[4, -45, 1, 3], [1, 4], [7, -8, 12, -44, 5]]) #affiche [7, -8, 12, -44, 5]
```

## Solution de l'exercice 11:

```
def listeTriangle(n):
        liste = []
        for i in range(n):
                sousListe = []
                for j in range(i+1):
                        sousListe.append(j+1)
                liste.append(sousListe)

        return liste

print listeTriangle(2)    #affiche [[1], [1, 2]]
print listeTriangle(4)    #affiche [[1], [1, 2], [1, 2, 3], [1, 2, 3, 4]]
print listeTriangle(0)    #affiche []
```

```
def listeTriangle(n):
    liste = []
    sousListe = []
    for i in range(n):
        sousListe.append(i+1)
        liste.append(sousListe[:])
    return liste

print listeTriangle(2)  #affiche [[1], [1, 2]]
print listeTriangle(4)  #affiche [[1], [1, 2], [1, 2, 3], [1, 2, 3, 4]]
print listeTriangle(0)  #affiche []
```