INFO-H-100 - Programmation
TP 5 — 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. Écrire une fonction qui calcule la somme des carrés d'une liste de nombres.

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

Ex. 2. Écrire 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(min_list([4, 3, 5, 6, 7]))      #prints 3
print(min_list([3, 12, 3, 2, 10]))    #prints 2
```

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

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

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

```
li1 = [3, -4, -5, 6, -4]
replace_abs(li1)
print(li1)                    #prints [3, 4, 5, 6, 4]
```

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

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

Ex. 6. Écrire 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]
shuffle(li1)
print(li1)   #prints for instance [5, -44, 7, -8, 12]
```

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

```
print(powers_of_2(4))   #prints [1, 2, 4, 8]
print(powers_of_2(10))  #prints [1, 2, 4, 8, 16, 32, 64, 128, 256, 512]
```

Ex. 8. Écrire une fonction qui produit une liste ordonnée contenant les nombres premiers inférieurs à un maximum donné. Commencez par écrire une fonction qui détermine si un nombre est premier.

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

Ex. 9. Écrire une fonction qui renvoie la plus longue liste d'une liste de listes.

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

Ex. 10. Écrire une fonction qui génère la liste suivante en fonction de n. Exemple pour $n = 3$:

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

```
print(triangle_list(2))   #prints [[1], [1, 2]]
print(triangle_list(4))   #prints [[1], [1, 2], [1, 2, 3], [1, 2, 3, 4]]
print(triangle_list(0))   #prints []
```

INFO-H-100 - Programmation
TP 5 — Boucles `for` et listes
Corrections

Solution de l'exercice 1:

```
def sum_squared(number_list):
    total = 0
    for number in number_list:
        total += number ** 2
    return total

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

Solution de l'exercice 2:

```
def min_list(number_list):
    if (len(number_list) == 0):
        raise Exception("empty list")

    minimum = number_list[0]
    for number in number_list:
        if number < minimum :
            minimum = number
    return minimum

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

Solution de l'exercice 3:

```
def elder(name_list):
    if (len(name_list) == 0):
        raise Exception("empty list")

    (elder_name, elder_age) = name_list[0]
    for (name,age) in name_list:
        if age > elder_age:
            (elder_name, elder_age) = (name, age)

    return elder_name

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

Autre solution :

```
def elder(name_list):
    if (len(name_list) == 0):
        raise Exception("empty list")

    i_max = 0                   # index of the elder
    for i in range(1,len(name_list)):
        if name_list[i][1] > name_list[i_max][1]:
            i_max = i

    return name_list[i_max][0]
```

Solution de l'exercice 4:

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

li1 = [3, -4, -5, 6, -4]
replace_abs(li1)
print(li1)                  #prints [3, 4, 5, 6, 4]
```

Solution de l'exercice 5:

```
def swap(_list, i1, i2):
    _list[i1], _list[i2] = _list[i2], _list[i1]

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

Solution de l'exercice 6:

```
import random

def swap(_list, i1, i2):
    _list[i1], _list[i2] = _list[i2], _list[i1]

def shuffle(number_list):
    for i in range(len(number_list)-1):
        position = random.randint(i,len(number_list)-1)
        swap(number_list, i, position)


li1 = [7, -8, 12, -44, 5]
shuffle(li1)
print(li1)   #prints for instance [5, -44, 7, -8, 12]
```

Solution de l'exercice 7:

```
def powers_of_2(n):
    li = []
    power = 1
    for i in range(n):
        li.append(power)
        power = power*2

    return li

print(powers_of_2(4))   #prints [1, 2, 4, 8]
print(powers_of_2(10))  #prints [1, 2, 4, 8, 16, 32, 64, 128, 256, 512]
```

Solution de l'exercice 8:

```

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

def prime_numbers(max):
    prime_number_list = []
    for i in range(max):
        if (is_prime(i)):
            prime_number_list.append(i)

    return prime_number_list

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

Solution de l'exercice 9:

```
def longest_list(lists):
    if (len(lists) == 0):
        raise Exception("empty list")

    longuest_list = lists[0]
    for _list in lists:
        if len(_list) > len(longuest_list):
            longuest_list = _list
```

```
    return longuest_list

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

Solution de l'exercice 10:

```
def triangle_list(n):
    _list = []
    for i in range(n):
        sub_list = []
        for j in range(i+1):
            sub_list.append(j+1)
        _list.append(sub_list)

    return _list

print(triangle_list(2))   #prints [[1], [1, 2]]
print(triangle_list(4))   #prints [[1], [1, 2], [1, 2, 3], [1, 2, 3, 4]]
print(triangle_list(0))   #prints []
```

Autre solution :

```
def triangle_list(n):
    _list = []
    sub_list = []
    for i in range(n):
        sub_list.append(i+1)
        _list.append(sub_list[:])
    return _list

print(triangle_list(2))  #prints [[1], [1, 2]]
print(triangle_list(4))  #prints [[1], [1, 2], [1, 2, 3], [1, 2, 3, 4]]
print(triangle_list(0))  #prints []
```