-Correction-

2| Applications de base avec Python

1. <u>Les Booléens</u> : Déterminer la valeur des expressions suivantes, en corrigeant la syntaxe si nécessaire

1. 3 * 3.5 > 10. — True
2. 3 *7 == 21 — True
3. 3 - 1 => 1. — True
4. 0<10**-300 == 100*-150 — False (se lit : True and False, donc False)
5. (2 - 1) == 1 == 4 + 3 — False (se lit : True and False, donc False)
6. not(True and False) — True
7. not True and False — False
8. not(not True) — True
9. (5.5 *2 == 11. or 1/2 != .5) and (3 *2 == 6) — True

2. <u>Opération sur des entiers et booléens</u> : Soient n, m et p trois entiers naturels. Écrire sur papier des expressions booléennes traduisant les propriétés suivantes :

1. n est divisible par 5. — n%5==0 (est-ce que le reste de la division de n par 5 vaut 0 ?)
2. l'un des entiers m ou n est multiple de l'autre. — m%n == 0 or n%m==0
3. les entiers m et n sont de même signe. — m*n >= 0
4. les entiers m, n et p sont de même signe. — m*n > 0 and m*p > 0
5. les entiers m, n, p sont distincts deux à deux. — m != n and n != p and p!= m

3. <u>Affectation de variables</u> : Sans taper les lignes suivantes, donner la valeur de x, y et z à l'issue des codes suivants :

```
>>> x = 2
>>> y = x
>>> x = 6
>>> y = x **2
>>> z = x + y
```

x = 6 , y = 36 , z = 42

```
>>> x = 2
>>> y = 3
>>> z = x
>>> x = y
>>> y = z
```

x = 3 , y =2 , z=2

Expliquer et donner la valeur de s à l'issue des codes suivants :

```
>>> s = 0
>>> for k in range (0, 10):
>>>     s = s + k
```

s : somme des entiers de 0 à 9, qui se calcule en maths avec la formule : 9*10/2=45

```
>>> s=1
>>> for i in range (1, 18):
>>>     s*= i
```

produit des entiers de 1 à 17 cad 17! Ne pas chercher à l'expliciter…

Donner les états successifs des variables p et c dans le code suivant :

```
>>> p = 5
>>> c= 0
>>> while p>0:
>>>     if c = = 0:
>>>         p = p-2
>>>         c = 1
>>>     else :
>>>         p += 1
>>>         c = 0
```

7 passages successifs dans la boucle

(p,c) initialisé en (5,0), prend les valeurs successives suivante : (3,1) , (4,0) , (2,1), (3,0), (1,1), (2,0), (0,1)

On dispose de trois variables x, y et z. Écrire quatre lignes de calcul permettant d'affecter à y la valeur de x, à z la valeur de y et à x la valeur de z (Il peut être nécessaire de créer une nouvelle variable…).

La difficulté est de ne pas « perdre » la valeur d'une des variables, par exemple si on commence par y=x, la valeur initiale de y a été perdue !! On crée pour cela une variable intermédiaire…

```
>>> memoire = x
>>> x = z
>>> z = y
>>> y = memoire
```

ou sur une ligne : `>>> x, y,z = z, x, y`

4. <u>Les listes</u> : Extraire à l'aide d'une suite d'instructions :

1. le 2 de la liste L = [4, 5, 3, 2, 1] `>>> L[3]`
2. le 3 de la liste L = [4, "soleil", [1, 3], 0] `>>> L[2][1]`
3. le "m" de la liste L = [5.1, 3, ' informatique ', 89.5] `>>> L[2][5]`

Construire à l'aide d'une suite d'instructions :

1. la liste des entiers de 0 à 15 inclus. >>> L=[i for i in range (0,16)]
2. la liste des nombres pairs entre 10 et 100.

```
>>>L=[]                          # création de la liste vide L
>>> for i in range (10,101):     # i va de 10 à 100 inclus
>>>     if i%2 == 0 :            # si i est pair
>>>         L.append (i)         # on ajoute i à L
```

Ou plus rapide :

```
>>> L=[i for i in range (10,101,2) ]    # i va de 10 à 100 inclus avec un pas de 2.
```

3. la liste des nombres inférieurs strictement à 150 de la forme 3k + 2, avec k entier naturel.

```
>>> L=[]
>>> k=0
>>> while 3*k+2 < 150 :          # Boucle "Tant que " + condition
>>>     L.append(3*k+2)
>>>     k+=1
```

Ecrire une suite d'instructions qui affiche True si x est dans la liste L et False sinon.

La solution astucieuse et pratique, déjà codée en python :

```
>>> x in L
```

Ou les solutions "manuelles" :

```
>>> for i in L:
>>>     if i == x:
>>>         print(True)
>>> print (False)
```

ou

```
>>> for i in range (len(L)):
>>>     if L[i] ==x:
>>>         print (True)
>>> print (False)
```

Proposer une suite d'instruction qui fait la somme terme à terme de deux listes M et L de même longueurs. (ex: [4,5,6] + [2,3,4] -> [6,8,10])

```
>>> N = []
>>> for i in range(len(L)):  La commande len(L) donne le nombre d'éléments d'un itérable (avec la convention qu'on part de 0), pratique pour fabriquer un compteur avec le bon nombre d'éléments.
>>>     N.append(M[i] + L[i])
```

5. <u>Les fonctions</u> : Identifier le rôle des fonctions définies ci-dessous :

```
>>> def f1(x):                    # x valeur réelle quelconque
>>>     if x >=0 :
>>>         return x
>>>     else :
>>>         return –x              valeur absolue de x
```

```
>>> def f2(n):                    # n, entier naturel
>>>     s,i = 0,0
>>>     while i < n :
>>>         i+=1
>>>         s+ = i**2
>>>     return s                   somme des carrés (il y a aussi une formule issue du cours de maths…)
```

```
>>> def f3(n):                    #n, entier naturel
>>>     p=1
>>>     for i in range (1, n+1):
>>>         p=p*i
>>>     return p                   produit factoriel de n autrement dit n!
```

```
>>> def f4(M):                    # M valeur réelle positive
>>>     s,i = 0,0
>>>     while s < M :
>>>         i+=1
>>>         s+= i**2
>>>     return i                   plus petit i tel que la somme des carrées entre 1 et i soit supérieure ou égale à M, autrement dit, le premier entier i tel que 1+…+i² dépasse M. Pour l'avoir théoriquement, on peut résoudre $\frac{i(i+1)}{2} \geq M$.
```

Ecrire les fonctions suivantes :

1. carre(x) qui prend en argument un réel x et qui renvoie x^2.

```
>>> def carre(x):
>>>     return x**2
```

2. somme(L) qui prend en argument une liste L et qui renvoie la somme de ses valeurs.

```
>>> def somme(L):
>>>     s=0
>>>     for i in L :
>>>         s+=i
>>>     return s
```

3. moyenne(L) qui prend en argument une liste L et qui renvoie la moyenne de ses valeurs.

```
>>> def moyenne(L):
```

```
>>>     s=0
>>>     compteur=0
>>>     for i in L :
>>>             compteur+=1
>>>             s+=i
>>>     return s/compteur
```

Ou

```
>>> def moyenne(L):
>>>     s= somme(L)                    #appel de la fonction précédente
>>>     return somme(L) / len (L)      #len(L) renvoie la longueur de la liste = nombre de terme
```

4. maximum(L) qui prend en argument une liste L et qui renvoie la valeur maximale de la liste.

Il faut parcourir la liste, en stockant la valeur maximale, et on l'actualisant si on en trouve une plus grande. Classique !

```
>>> def maximum(L):
>>>     maxi = L[0]                    # On suppose le premier terme comme étant le max
>>>     for i in range (1, len(L)):
>>>             if L[i] > maxi:        # Si on trouve un nouveau terme max
>>>                     maxi = L[i]    # On écrase l'ancien et enregistre le nouveau max
>>>     return maxi
```

5. maximumS(L) qui prend en argument une liste L et qui renvoie une liste avec la valeur maximale de la liste L et le nombre de fois qu'elle apparaît (exemple : maximumS([4,5,6,1,6]) -> [6,2]).

```
>>> def maximumS(L):
>>>     maxi = maximum(L)              # On récupère la valeur de maxi de la liste
>>>     s=0
>>>     for i in L:                    # On parcours la liste L
>>>             if i == maxi :         # Si la valeur vaut le maxi
>>>                     s+=1           # On ajoute 1 au compteur s
>>>     return [maxi, s]
```

Algorithme d'Euclide

Pour a et b entiers naturels, avec b≠0, on veut calculer l'unique couple d'entiers (q,r) (quotient et reste) tels que a = bq+r et 0≤ r < b. La construction de q et r est un algorithme « récursif », qui consiste à procèder ainsi :

si a ≥ b , on remplace a par a-b et tant que a ≥ b on recommence.

si a < b, le reste cherché est a et le quotient est le nombre d'itérations.

Vérifiez avec un cas concret que cela fonctionne !

Donner une fonction python appelée euclide(a,b) qui renvoie le couple (q,r) de la division euclidienne de a par b.

```
>>> def euclide(a,b):
>>>     q=0                          # initialisation du compteur
>>>     while a>=b:  Une boucle while est idéale car on ne sait pas combien d'itérations on va faire.
>>>             a=a-b
>>>             q+=1
>>>     return q,a
```