2| Applications de base avec Python

1. **<u>Les Booléens</u> : Déterminer la valeur des expressions suivantes, et, pour les questions 1 à 5, dans le cas où la réponse est False, corriger afin d'obtenir True.**

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

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

    1. n est divisible par 5.
    2. l'un des entiers m ou n est multiple de l'autre.
    3. les entiers m et n sont de même signe.
    4. les entiers m, n et p sont de même signe.
    5. les entiers m, n, p sont distincts deux à deux.

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

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

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
```

On dispose de trois variables x, y et z. Écrire quatre lignes de code 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…).

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

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

1. la liste des entiers de 0 à 15 inclus.
2. la liste des nombres pairs entre 10 et 100.
3. la liste des nombres inférieurs strictement à 150 de la forme 3k + 2, avec k entier naturel.

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

Proposer une suite d'instructions 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])

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
```

```
>>> def f2(n):                    # n, entier naturel
>>>     s,i = 0,0
>>>     while i < n :
>>>             i+=1
>>>             s+ = i**2
>>>     return s
```

```
>>> def f3(n):                    # n, entier naturel
>>>     p=1
>>>     for i in range (1, n+1):
>>>             p=p*i
>>>     return p
```

```
>>> def f4(M):                    # M valeur réelle positive
>>>     s,i = 0,0
>>>     while s < M :
>>>             i+=1
>>>             s+= i**2
>>>     return i
```

Ecrire les fonctions suivantes :

1. carre(x) qui prend en argument un réel x et qui renvoie x^2.
2. somme(L) qui prend en argument une liste L et qui renvoie la somme de ses valeurs.
3. moyenne(L) qui prend en argument une liste L et qui renvoie la moyenne de ses valeurs.
4. maximum(L) qui prend en argument une liste L et qui renvoie la valeur maximale de la liste.
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]).

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. On procède ainsi :

si a ≥ b , à a on affecte 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.

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

On verra qu'on peut *prouver* qu'un tel algorithme aboutit à quelque chose. Cette algorithme sera également au cœur de la preuve en maths de l'existence de la division euclidienne.