Python
Les fonctions

1. Pourquoi définir des fonctions

❑ **Rendre le programme plus lisible**

Le programmeur peut inclure dans une fonction une séquence d'instructions ayant un rôle précis.
Une règle en programmation est de ne pas écrire plusieurs fois une même séquence d'instructions.

❑ **Ne pas écrire deux fois le même code**

Si une séquence d'instructions doit être exécutée plusieurs fois au cours du programme, il faut l'inclure dans une fonction de manière à l'écrire une seule fois. Dans le programme, la fonction peut être exécutée autant de fois que nécessaire.

❑ **Représenter une fonction mathématique**

Une fonction logicielle peut servir à déclarer une fonction mathématique, par exemple une fonction polynomiale.

2. Comment déclarer une fonction

2.1. Une fonction qui fait toujours la même chose :

Un exemple très basique est celui d'une fonction qui affiche toujours la même chaîne de caractères.

```
def Message():
    print("Veuillez saisir un nombre entier compris entre 0 et 100.")
```

Pour afficher le message, il suffit d'appeler la fonction par son nom suivi des parenthèses.

```
Message()
...
Message()
```

Exemple d'une fonction qui affiche la date et l'heure :

```
from time import *
def AfficherDateHeure():                    # définition de la fonction
    print(strftime('%d/%m/%y %H:%M',localtime()))

AfficherDateHeure()                         # appel de la fonction
```

Dans le script, la définition de la fonction doit précéder l'utilisation de la fonction.

2.2. Fonction avec passage de paramètres :

La fonction peut recevoir des variables du programme grâce au passage de paramètres.

```
def Bonjour(nom):                           # définition
    print("Bonjour " + nom)

Bonjour("Bastien")                          # appel
```

Les paramètres reçus par la fonction sont appelés arguments.

Lors de l'appel de la fonction, le paramètre peut être passé sous la forme de valeurs ou sous la forme de noms de variables.

```
def Message(min, max):
   print("Saisir un nombre compris entre {} et {}".format(min , max)

Message(5, 25)                    # 1er appel de la fonction
...
a = 0.0, b = 9.9
Message(a, b)                     # 2ème appel de la fonction
```

Les paramètres ne portent pas obligatoirement le même nom dans la définition de la fonction (ici min et max) et lors de l'appel de la fonction.

Dans le cas du 2ème appel, **min est une copie de la variable a et max est une copie de b.**

2.3. Fonctions qui retournent une valeur :

Les fonctions précédentes avaient pour rôle d'afficher une chaîne de caractères.

On utilise très souvent les fonctions pour réaliser un traitement ou un calcul et restituer le résultat.

```
from math import sqrt
def Module(a, b):
    m = sqrt(a**2 + b**2)
    return m
```

Le mot clé *return* permet à la fonction de renvoyer une valeur exploitable par le programme.

return peut être suivi du nom d'une variable ou d'une expression.

```
def Module(a, b):
    return sqrt(a**2 + b**2)
```

Le programme récupère la valeur renvoyée par la fonction en l'affectant à une variable.

```
resultat = Module(3, 4)
print("Le module est égal à ", resultat)
```

2.4. Définir une fonction mathématique f(x) :

On peut définir une fonction mathématique f(x) en choisissant une expression parmi plusieurs.

```
from math import exp
def f(x):                              # définition de la fonction
     #return x**2 - 1
     return 1 - exp(-x)

y = f(0)                               # appel de la fonction
```

2.5. Passage d'une fonction en argument d'une autre fonction :

Une fonction mathématique *f(x)* définie dans le programme peut être fournie à une autre fonction.

Un exemple pratique est le calcul de la somme : $\sum_{i=0}^{n} f(i)$

Cette technique peut être utilisée pour calculer la valeur approchée d'une intégrale par la méthode des rectangles ou par la méthode des trapèzes.

Il faut définir deux fonctions : par exemple une fonction *f()* et une *Somme()*.

2.6. Fonction qui renvoie une fonction :

Une fonction qui reçoit en argument les deux fonctions f et g peut par exemple renvoyer la composée f o g.

```
def Composee(f, g):
    def h(x):
        return f(g(x))
    return h
```

2.7. Exercices :

Codez la fonction *Moyenne()* qui calcule la moyenne des éléments d'une liste et renvoie le résultat.

```
def Moyenne(liste):
    return sum(liste)/len(liste)
```

Donnez un exemple d'appel de cette fonction. Le résultat doit être stocké dans la variable *moy*.

```
l = [7, 4, 9, 2]
moy = Moyenne(l)
```

Écrivez la définition de la fonction *Equation()* qui renvoie la liste des racines d'une équation du second degré. Les coefficients a, b et c de l'équation $a.x^2 + b.x + c = 0$ sont passés en paramètres.

```
def Equation(a,  b,  c):
    d = b**2 - 4*a*c
    if d > 0:
        r1= (-b + sqrt(d))/(2*a)
        r2= (-b - sqrt(d))/(2*a)
        racines = [r1,  r2]
    elif d < 0:
        racines = []
    else:
        racines = [-b/(2*a)]
    return racines
```

On donne un exemple d'appel de cette fonction.

```
r = Equation(1, 1, -2)
```

Comment connaître le nombre de racines ? Comment afficher leurs valeurs ?

Nombre de racines : longueur de la liste r.
Afficher les éléments de la liste r.

Complétez la définition de la fonction *Somme()* pour le paragraphe 2.5.

```
def Somme(f, n):
    s = 0
    for i in range(n+1):
        s = s + f(i)
    return s
```