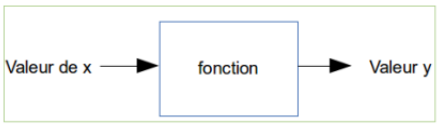
Python : les fonctions:

1. **Les fonctions:**
   Les fonctions permettent de décomposer un programme complexe en une série de sous-programmes plus simples. De plus, les fonctions sont réutilisables : si nous disposons d'une fonction capable de calculer une racine carrée, par exemple, nous pouvons l'utiliser un peu partout dans notre programme sans avoir à la réécrire à chaque fois (on parle de factorisation du code)

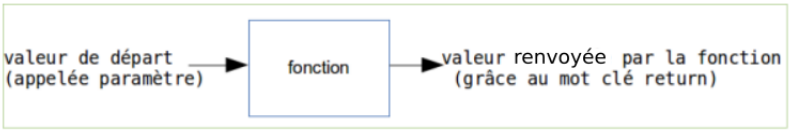
   La notion de fonction en informatique est comparable à la notion de fonction en mathématiques.



   Si nous avons y = 3x+2, pour une valeur donnée de x, nous aurons une valeur de y.

   Exemple : x=4 donc y= 14 (y = 3.4+2=14, attention ici le point correspond au signe "multiplié").

   La fonction en informatique est basée sur la même idée :



   Voici la syntaxe employée en Python pour définir une fonction :

```
0 def nom_fonction(x):
      instruction 1
2     instruction 2
      return y
```

prgm.py

   La fonction renvoie la valeur contenue dans la variable y.

   ATTENTION : Notez bien la présence du décalage entre la première ligne et les lignes suivantes. Ce décalage est appelé indentation, l'indentation permet de définir un bloc de code. Dans l'exemple ci-dessus, l'indentation nous permet de savoir que "instruction_1", "instruction_2" et "return y" constituent un bloc de code, ce bloc correspond au contenu de la fonction. "suite programme" ne fait pas partie de la fonction, car il n'est pas indenté. Pour indenter du code, il

y a 2 solutions : mettre 4 espaces ou utiliser une tabulation. En Python il est conseillé d'utiliser les 4 espaces, mais ce n'est pas une obligation. Une chose est sûre, une fois que vous avez choisi une méthode, n'en changé surtout pas au cours d'un même programme !

Codons notre exemple (y=3x+2) en créant une fonction ma_fonction :

```
0 def ma_fonction(x):
      y = 3*x+2
2     return y
```

prgm.py

Pour "utiliser" la fonction ma_fonction, il suffit d'écrire : ma_fonction (4) (dans ce cas précis, notre fonction renverra le nombre 14).

Exercice 16: Testez le programme suivant (quelle est la valeur référencée par la variable solution après l'exécution du programme) :

```
0 def ma_fonction(x):
      y = 3*x+2
2     return y
solution = ma_fonction(4)
```

prgm.py

Il faut savoir qu'au moment de l'exécution de votre programme le code ma_fonction (4) sera systématiquement remplacé par la valeur renvoyée par la fonction (toujours dans notre exemple le ma_fonction (4) sera remplacé par le nombre 14).

Exercice 17: Codez en Python la fonction $y = x^2 + 2x + 10$. Testez ensuite cette fonction.

Il est possible de faire passer plusieurs paramètres à une fonction.

Exercice 18: Quel est le résultat renvoyé par la fonction ci-dessous si l'on saisit dans la console une_autre_fonction(5, 3)

```
0 def une_autre_fonction(x,b):
      y = 3*x+b
2     return y
```

prgm.py

Les paramètres peuvent être des chaînes de caractères (ainsi que la valeur retournée)

Exercice 19: Quel est le résultat attendu après l'exécution du programme ci-dessous et la saisie dans la console de "dit_bonjour("toto", 14)" ?

```
0 def une_autre_fonction(x,b):
      y = 3*x+b
2     return y
```

prgm.py

Attention : remarquez bien les guillemets autour du paramètre "toto" (c'est une chaîne de caractères)

Les paramètres ne sont pas obligatoires.

Exercice 20: Testez la fonction suivante :

```
0 def ma_fon():
      return 'cette fonction ne sert a rien'
```

prgm.py

Il faut aussi savoir qu'une fonction ne renvoie pas forcément de valeur (le mot clé return n'est pas obligatoire). Mais si elle ne renvoie pas de valeur, que fait-elle ? Elle peut faire plein de choses, par exemple elle peut tout simplement afficher une chaîne de caractères à l'aide d'un "print". Sachez que dans certains langages, on utilise les termes méthode ou procédure pour qualifier une fonction "qui ne renvoie rien".

Exercice 21: Soit le programme suivant :

```
0 def dit_bonjour(nom,age):
      phrase = f'Bonjour {nom}, vous avez {age} ans.'
2     return phrase
```

prgm.py

Testez la fonction dit_bonjour à l'aide de la console (avec par exemple un dit_bonjour("toto", 14))