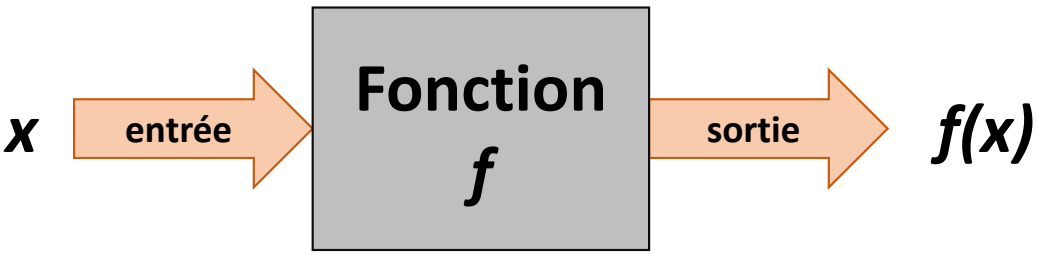
PYTHON 05 : les fonctions

En mathématiques, une fonction est une espèce de « boîte noire », qui reçoit un nombre en entrée, et qui fournit un nombre en sortie :



Par exemple, si on a : $f(x) = 4x^2 + 5x + 3$, alors : $f(4) = 4 \times 4^2 + 5 \times 4 + 3 = 64 + 20 + 3 = 87$.
On dit que 87 est l'image de 4 par la fonction f.

On peut définir des fonctions avec Python grâce à l'instruction « **`def`** ».

* Tapez l'exemple suivant dans l'éditeur Thonny. Attention : on rappelle que l'indentation d'un programme Python (c'est-à-dire le décalage des lignes par rapport au bord gauche de l'écran) est fondamentale.

```
def f(x):
    return 4 * x ** 2 + 5 * x + 3
```

* Lancez le programme en cliquant sur la flèche verte en haut de la fenêtre Thonny. Il ne se passe rien à première vue.

* Placez vous dans la console, et tapez : **`f(4)`**. Que remarque-t-on ? Essayez de calculer l'image d'autres nombres par la fonction « **`f`** ». (Vous pourrez vérifier les résultats donnés par l'ordinateur à l'aide de votre calculatrice).

Le programme a donc créé une fonction dont le nom est « **`f`** ».

La lettre « **`x`** » entre parenthèses derrière « **`f`** » est un argument de la fonction : il s'agit d'un paramètre qui correspond à l'entrée de la fonction.

L'instruction « **`def`** » permet de définir la fonction.

L'instruction « **`return`** » renvoie le résultat de la fonction. Elle correspond à la sortie de la fonction.

Une fois que l'on a définit la fonction, on l'utilise simplement en l'appelant par son nom. On peut aussi faire des calculs avec le résultat d'une fonction.

Essayez de taper dans la console : **`f(1)`**, puis **`3 * f(1)`**. Que remarque-t-on ?

Une fonction Python peut recevoir plusieurs arguments en entrée.

* Pour illustrer ce dernier point, tapez ensuite le programme suivant :

```
def addition(a, b):
    return a + b
```

* Lancez le programme en cliquant sur la flèche verte en haut de la fenêtre Thonny. Il ne se passe toujours rien à première vue.

* Placez vous dans la console, puis tapez : **`addition(12, 27)`**. Le résultat était est-il prévisible ?

* Complétez le programme en créant trois nouvelles fonctions appelées **`soustraction`**, **`multiplication`** et **`division`**, acceptant chacune deux arguments, et réalisant les opérations correspondantes.

Une fonction Python peut aussi renvoyer plusieurs résultats :

* Tapez le programme suivant :

```
def puissances(n):
    # renvoie le carré et le cube du nombre n
    return n ** 2, n ** 3
```

* Placez-vous dans la console et testez la fonction « puissances » avec plusieurs valeurs que vous choisirez.

Les exemples précédents sont des fonctions très simples, réalisant des calculs basiques. Mais en Python une fonction peut être composée de plusieurs instructions, comme celles que nous avons vues précédemment, notamment les tests et les boucles.

* Tapez l'exemple suivant dans l'éditeur Thonny :

```
def somme(n):
    resultat = 0
    for i in range(1, n+1):
        resultat = resultat + i
    return resultat
```

* Lancez le programme en cliquant sur la flèche verte en haut de la fenêtre Thonny, puis placez-vous dans la console.

* Tapez : **`somme(1)`**, **`somme(2)`**, **`somme(3)`**, **`somme(4)`**, puis **`somme(5)`**. Que remarque-t-on ? Que fait exactement la fonction « **`somme`** » ? (On pourra s'aider des résultats précédents, ainsi que du texte du programme).

* Complétez le programme en créant une nouvelle fonction appelée **`produit`** acceptant un argument **`n`**, et réalisant le produit des nombres compris entre 1 et **`n`**.

* A l'aide de tout ce que vous avez vu précédemment, réaliser une fonction appelée « **`test`** » qui attend un argument **`n`**. Cette fonction doit renvoyer la somme des carrés des nombres compris entre 1 et **`n`** :

$$f(n) = 1^2 + 2^2 + 3^2 + \cdots + n^2$$

* Réaliser une fonction « **`test2`** » qui attend un argument **`n`**, et qui renvoie la somme des inverses des carrés des nombres compris entre 1 et **`n`** :

$$f(n) = \frac{1}{1^2} + \frac{1}{2^2} + \frac{1}{3^2} + \cdots + \frac{1}{n^2}$$

Note : les fonctions Python peuvent aussi opérer sur d'autres objets que des nombres, comme par exemple les chaînes de caractères.

* Tapez et exécuter le programme suivant :

```
def repetition(mot, nombre):
    # mot est une chaîne de caractères
    # nombre est un nombre entier
    resultat = mot * nombre
    return resultat
```

* Placez-vous dans la console et tapez : « **`repetition("HELLO", 4)`** ». Le résultat était-il prévisible ? Testez la fonction avec d'autres valeurs pour les arguments.