Les fonctions

Les fonctions : Introduction

Une fonction est un concept fondamental en programmation et en mathématiques. Elle représente une relation entre un ensemble de valeurs en entrée, appelées **arguments**, et un **ensemble de valeurs en sortie**, appelées résultats.

Les fonctions permettent de regrouper un **ensemble d'instructions ou de calculs qui peuvent être réutilisés** à plusieurs endroits dans un programme, favorisant ainsi la modularité, la lisibilité et la réduction de la redondance de code.

En mathématiques, une fonction est définie comme une correspondance qui associe à chaque élément d'un ensemble, appelé domaine, un unique élément d'un autre ensemble, appelé codomaine ou ensemble d'arrivée.

Les fonctions mathématiques sont généralement représentées par des expressions algébriques, des équations ou des graphiques.

Les fonctions : Introduction

En programmation, une fonction est une séquence d'instructions regroupées sous un **nom spécifique** et qui peut être appelée depuis d'autres parties du programme.

Les fonctions **peuvent prendre des arguments en entrée**, qui sont des valeurs fournies lors de l'appel de la fonction, et elles **peuvent renvoyer un résultat en sortie**. Les fonctions permettent d'encapsuler des portions de code dans des unités logiques et autonomes, facilitant ainsi la maintenance, la réutilisation et la compréhension du code.

Une fonction est généralement **définie par une en-tête qui spécifie son nom**, **ses paramètres d'entrée** et **son type de retour**, ainsi qu'**un corps qui contient les instructions à exécuter** lorsque la fonction est appelée.

Les paramètres d'entrée permettent de transmettre des valeurs à la fonction, et le type de retour indique le type de valeur renvoyé par la fonction. Les fonctions peuvent être utilisées pour effectuer des opérations simples ou complexes, comme effectuer des calculs mathématiques, manipuler des chaînes de caractères, accéder à des données dans une base de données, interagir avec l'utilisateur, etc.

Elles peuvent également être utilisées pour décomposer un problème complexe en sous-problèmes plus simples, en adoptant une approche de développement modulaire.

Les fonctions : Algobox

Voici un exemple simple de déclaration de fonction en pseudo-code :

```
FONCTIONS_UTILISEES
 FONCTION somme(a, b)
   VARIABLES_FONCTION
   DEBUT_FONCTION
     RENVOYER a + b
   FIN_FONCTION
VARIABLES
DEBUT_ALGORITHME
 AFFICHERCALCUL somme(2, 2)
FIN_ALGORITHME
```

Les fonctions : les arguments

Une fonction est un bloc de code organisé utilisé pour effectuer une seule tâche. Elle offre une meilleure modularité dans un script et facilite la réutilisation.

Une fonction peut prendre des arguments et renvoyer **des valeurs**.
Dans l'exemple suivant, la fonction **say_hello** reçoit l'argument « name » et imprime un message d'accueil.

```
>>> def say_hello(name):
...     print(f'Hello {name}')
...
>>> say_hello('Carlos')
# Hello Carlos

>>> say_hello('Wanda')
# Hello Wanda

>>> say_hello('Rose')
# Hello Rose
```

Les fonctions : mots clés et arguments

Pour améliorer la lisibilité du code, nous devons être aussi explicites que possible. Nous pouvons y parvenir dans nos fonctions en utilisant des arguments de mots-clés :

```
>>> def say_hi(name, greeting):
...     print(f"{greeting} {name}")
...
>>> # Version sans les mots-clé
>>> say_hi('John', 'Hello')
# Hello John

>>> # avec les mots-clé
>>> say_hi(name='Anna', greeting='Hi')
# Hi Anna
```

Les fonctions : Valeurs de retour

Lors de la création d'une fonction à l'aide de l'instruction **def**, vous pouvez spécifier quelle doit être la valeur de retour avec une instruction **return**.
Une instruction return se compose des éléments suivants :

- Le mot-clé **return**.
- La valeur ou l'expression que la fonction doit renvoyer.

```
>>> def sum_two_numbers(number_1, number_2):
...     return number_1 + number_2
...
>>> result = sum_two_numbers(7, 8)
>>> print(result)
# 15
```

Les fonctions : la portée des variables (local/global)

- Le code dans la portée globale ne peut utiliser aucune variable locale.
- Cependant, une portée locale peut accéder aux variables globales.
- Le code dans la portée locale d'une fonction ne peut pas utiliser de variables dans une autre portée locale.
- Vous pouvez utiliser le même nom pour différentes variables si elles se trouvent dans des portées différentes. Autrement dit, il peut y avoir une variable locale nommée spam et une variable globale également nommée spam.

```
global_variable = 'Je suis accessible de partout'

>>> def some_function():
...     print(global_variable)  # parce qu'elle est globale
...     local_variable = "uniquement accessible dans cette fonction"
...     print(local_variable)
...
>>> # Le code suivant renvoie une erreur car
>>> # 'local_variable' n'existe que dans 'some_function'
>>> print(local_variable)
Traceback (most recent call last):
  File "<stdin>", line 10, in <module>
NameError: name 'local_variable' is not defined
```

Les fonctions : Global Statement

Si vous devez modifier une variable globale depuis une fonction, utilisez l'instruction **global** :

```
>>> scope = 'local'

>>> def change_scope():
...     global scope
...     scope = 'global'
...
>>> change_scope()
>>> print(scope)
```

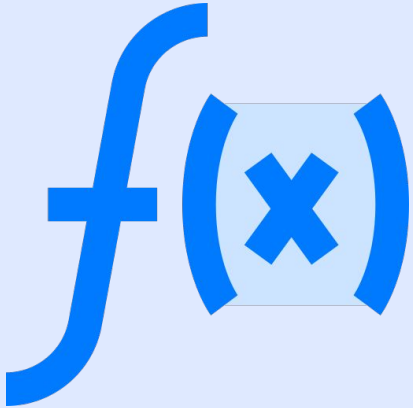
Les fonctions : Lambda

Les fonctions Lambda: Introduction

En Python, une fonction **lambda** est une **fonction anonyme sur une seule ligne**. Elle peut avoir un nombre quelconque d'arguments, mais ne peut avoir qu'une seule expression.

```
>>> def add(x, y):
...     return x + y
...
>>> add(5, 3)
# 8


#un équivalent vers une fonction lambda :


>>> add = lambda x, y: x + y
>>> add(5, 3)
# 8


#Comme les fonctions imbriquées classiques, les lambdas fonctionnent également comme des closures :


>>> def make_adder(n):
...     return lambda x: x + n
...
>>> plus_3 = make_adder(3)
>>> plus_5 = make_adder(5)

>>> plus_3(4)
# 7
>>> plus_5(4)
# 9
```

[Les fonctions]
Workshop



Réaliser les exercices

Réalisez les exercices suivants.

L'intégralité de vos exercices seront à faire avec **Visual Studio Code** dans un dossier variable avec à l'intérieur un fichier par exercice par exemple :

/fonctions
—--- exo1.py
—--- exo2.py
....

Exercices disponible ici.

Récupérer le Cours

Scannez moi