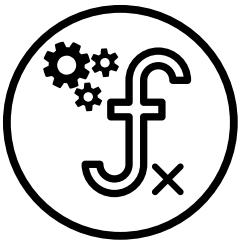
Les fonctions



Les fonctions sont des bouts de code d'un programme qui sont exécutés à chaque fois qu'ils sont appelés. Elles permettent de structurer le code, d'éviter les redondances, de le rendre plus lisible dans le but d'en faciliter la maintenance. L'utilisation de fonctions fait partie des bonnes pratiques de la programmation.

1. Définition d'une fonction

```
# Code hors fonction
def nom_fonction(argument1, argument2, ...):
    # Code de la fonction
      return valeur de retour
# Code hors fonction
```

Une fonction se définit avec l'instruction **def** suivie du nom de la fonction et de la liste de ses arguments.
La valeur de retour est indiquée par l'instruction (optionnelle) **return**. Si celle ci n'est pas présente, la valeur par défaut est retournée

2.Appel d'une fonction

Considérons une fonction double qui multiplie par 2 la somme de deux nombres passés en paramètre. Une telle fonction s'écrira :

```
def double(a,b) :
      resultat=2*(a+b)
      return resultat
```

L'appel suivant de la fonction renvoie donc la valeur 24 (2*(4+8) = 24)

```
>> double(4,8)
24
```

3.Spécification d'une fonction

Une fonction prend des arguments en paramètre, les manipule et renvoie un résultat. Cette organisation d'un programme permet la réutilisation et le partage d'algorithmes plus ou moins complexes. Cette notion de partage oblige les concepteurs de fonctions à décrire la fonction afin de spécifier :

- ✗ Le nom de l'algorithme,
- ✗ La description des paramètres : quelles variables, quels types ...
- ✗ La description du résultat

Cette spécification est réalisée au début de la fonction par l'utilisation de **docstrings**.

```
def puissance(n,p) :
    ''' Calcul la puissance de p du nombre n
    arguments :    n nombre reel
                   p nombre reel
    retour : renvoie le resultat de n puissance p
    '''
    return n**p
```

Remarque : Les docstrings sont créés en apposant trois simples cotes au début du commentaires et trois autres décalés d'une tabulation.

4.Portée des variables

Les variables locales

Les variables définies à l'intérieur de la fonction ont une portée locale c'est à dire qu'elles sont créées à l'appel de la fonction et détruites à la sortie de la fonction.

```
def ajoute_1(x) :
    resultat = x + 1
    return resultat
```

Ici resultat, est une variable locale à la fonction. Celle-ci n'existe qu'au sein de la fonction ajoute_1 et n'est pas visible par le programme principal.

Variable globale

Par opposition aux variables locales, une variable globale est une variable créée dans le programme principal (hors de toute fonction). Une variable globale est accessible partout : dans les fonctions et dans le programme principal.
Par exemple, ce programme affiche la valeur **84**

```
def double() :
    return 2 * v
v = 42
w = double()
print(w)
```

On recommande cependant de ne pas définir de fonctions dont le code dépend de variables globales dont la valeur peut changer. A l'inverse, il est tout à fait légitime d'utiliser des variables globales pour stocker des constantes utilisées dans des fonctions.

5.Erreurs fréquentes lors de la programmation de fonctions

1. La définition de la fonction se termine par ':'
2. La fonction doit être documentée par l'utilisation de docstrings. Cette documentation est indentée par rapport à la définition de la fonction.
3. Le corps de la fonction doit être indenté
4. Les variables locales ne doivent pas être appelées à l'extérieur de la fonction
5. L'instruction print n'équivaut pas à un return. En effet print permet d'afficher un résultat dans la console alors que return permet de sortir de la fonction et de renvoyer un résultat si besoin.