L. n°4

Les fonctions en Python

Jusqu'à présent, vous avez utilisé des fonctions proposées par Python ou un de ses modules (par exemple, la fonction `print()` ou encore `random.randint()` du module `random` …).
Cependant, il est recommandé de décomposer un code source en fonctions renfermant les traitements des scripts. On y gagne en lisibilité, modularité, réutilisabilité, maintenance de code.

I. Définition

Une fonction est un **ensemble d'instructions** (ou **portion de code indépendante**) que l'on peut appeler au besoin (c'est une sorte de **sous-programme**).

Une fonction peut recevoir 0,1 ou plusieurs arguments, qui peuvent être des valeurs, variables ou fonctions, et renvoie le contenu d'une ou plusieurs variables.

On peut ainsi écrire un sous-programme au début du programme principal. En appelant plusieurs fois cette fonction, obtient ainsi un programme principal plus court et plus lisible (on ne fait pas de copier-coller d'un même processus)

II. Syntaxe

Pour définir une fonction avec Python, on utilise la syntaxe suivante :

- On saisit le mot-clé `def` suivi du nom de la fonction et de parenthèses ;
- On indique les paramètres, si besoin, dans les parenthèses, séparés par des virgules (Ces paramètres correspondent à des informations dont la fonction a besoin pour réaliser le traitement dont elle a la charge) ;
- On saisit les instructions à effectuer dans un bloc avec une indentation ;
- On utilise le mot-clé `return` pour renvoyer le(s) résultat(s) du traitement

Exemple :

```
def nom_de_la_fonction(parametre1, parametre2, parametre3, ...):
    """ Documentation
qu'on peut écrire
sur plusieurs lignes """        # docstring entouré de 3 guillemets (ou apostrophes)

    bloc d'instructions         # attention à l'indentation

    return resultat             # la fonction retourne le contenu de la variable resultat
```

⚠ **Attention à l'indentation !**

III. Fonctions SANS paramètre

Exemple 1 : La fonction suivante simule le comportement d'un dé à 6 faces. Pour cela, on utilise la fonction `randint()` de la librairie (appelée aussi bibliothèque ou module) `random`.

```
def tirage_de():
    """ Retourne un nombre entier aléatoire entre 1 et 6 """
    import random
    valeur = random.randint(1, 6)
    return valeur
```

Dans la console, on appelle ensuite la fonction de la manière suivante :

```
>>> print(tirage_de())
3
>>> print(tirage_de())
6
>>> resultat = tirage_de()
>>> print(resultat)
1
```

Exemple 2 : une autre façon de faire appel à la fonction tirage_de :

```
# définition des fonctions
def info():
    """ Informations """
    print("Touche q pour quitter")
    print("Touche Enter pour continuer")

def tirage_de():
    """ Retourne un nombre entier aléatoire entre 1 et 6 """
    import random
    valeur = random.randint(1, 6)
    return valeur

# début du programme
info()
while True:
    choix = input()
    if choix == 'q':
        break
    print("Tirage :", tirage_de())
```

IV. Fonctions AVEC paramètre(s)

Exemple 1 : Une fonction qui affiche la parité d'un nombre entier :

```
# définition de fonction
def parite(nombre):
    """ Affiche la parité d'un nombre entier """
    if nombre%2 == 1:   # L'opérateur % donne le reste d'une division
        print(nombre, 'est impair')
        return
    if nombre%2 == 0:
        print(nombre, 'est pair')
        return
```

Remarque :
l'instruction `return` quitte la fonction et continue l'exécution du programme principal.

Dans cet exemple, on appelle la fonction dans la console et on obtient l'affichage suivant :

```
>>> parite(13)
13 est impair
>>> parite(24)
24 est pair
```

Exemple 2 : Une fonction qui affiche 5 tirages aléatoires successifs de nombres :

```
# définition de fonction
def tirage_de2(valeur_min, valeur_max):
    """ Retourne un nombre entier aléatoire entre valeur_min et valeur_max """
    import random
    return random.randint(valeur_min, valeur_max)

# début du programme
for i in range(5):
    print(tirage_de2(1, 10))    # appel de la fonction avec les arguments 1 et 10
```

Remarque : la fonction est appelée directement dans le programme et pas dans la console.

Dans la console, on obtient l'affichage suivant :

```
>>>
6
7
1
10
2
>>>
```

Exemple 3 : Une fonction qui retourne une liste :

```
# définition de fonction
def tirage_multiple_de(nombretirage):
    """ Retourne une liste de nombres entiers aléatoires entre 1 et 6 """
    import random
    resultat = [random.randint(1, 6) for i in range(nombretirage)]      #
    return resultat

# début du programme
print(tirage_multiple_de(10))
```

Dans la console, on obtient l'affichage suivant :

```
>>>
[4, 1, 3, 3, 2, 1, 6, 6, 2, 5]
```