ISN – Informatique et Sciences du Numérique

LANGAGE PYTHON : FONCTIONS

1 – UTILITE DES FONCTIONS

Une fonction est une **portion de code** (sorte de sous-programme) que l'on peut appeler au besoin. L'utilisation des fonctions permet :

- **d'éviter la répétition** ;
- de **mettre en relief les données et les résultats** : entrées et sorties de la fonction ;
- la **réutilisation dans d'autres scripts** par l'intermédiaire du mécanisme de l'import ;
- de **décomposer une tâche complexe** en tâches plus simples.

On obtient ainsi des **programmes plus courts et plus lisibles**.

2 – SYNTAXE

```
def nomFonction(parametres1,parametre2,parametre3):
    # Documentation de la fonction.
    bloc_instructions>
    return resultat
```

Exemple 1 : fonction « Conversion degrés Celsius en degrés Kelvin »

```
def conv_celsius_kelvin(degres_celsius) :
    # cette fonction permet de convertir des
    # degrés Celsius en degrés Kelvin
    degres_kelvin = degres_celsius + 273
    return degres_kelvin
```

```
>>> conv_celsius_kelvin(0)
273
>>> conv_celsius_kelvin(-273)
0
>>> conv_celsius_kelvin(30)
303
```

3 – PASSAGE DE PARAMETRES

Le **passage de paramètres** permet de **fournir les données** utiles à la fonction. Ce passage s'effectue **lors de l'appel** de la fonction. Il est possible de fournir **plusieurs paramètres** à la fonction. Dans l'exemple précédant, il faut **fournir le paramètre** « **degres_celsius** » à la fonction pour son exécution.

Exemple 2 : fonction « Portion de table de multiplication quelconque »

```
def Table_Mul(table,debut,fin) :
    # ------------------------------------
    # cette fonction permet d'afficher une portion
    # d'une table de mulitiplication quelconque
    # table : table de multiplication attendue
    # debut : à partir de quelle valeur
    # fin : jusqu quelle valeur
    # ------------------------------------
    n = debut
    while n <= fin :
        print(n,"*",table,"=",n*table)
        n = n + 1

# programme principal
num_table = int(input("Quelle table voulez-vous ?"))
num_debut = int(input("A partir de quelle valeur ?"))
num_fin = int(input("Jusqu'à quelle valeur ?"))
print("Table de multiplication par",num_table,"de",num_debut,"à",num_fin)
Table_Mul(num_table,num_debut,num_fin)

>>>
Table de multiplication par 8 de 5 à 9
5 * 8 = 40
6 * 8 = 48
7 * 8 = 56
8 * 8 = 64
9 * 8 = 72
```

Dans l'exemple ci-dessus, il fournir les paramètres « table », « debut » et « fin » à la fonction « Table_Mul ». Par contre le corps d'instruction de la fonction « Table_Mul » **ne contient pas de return**, c'est-à-dire qu'elle ne **retourne pas de résultat**. Il s'agit d'une **procédure**.

Les paramètres passés en arguments peuvent de **types simples** (**int**, **float**, **str**…) mais également de **types plus complexes** (**tuple**, **list**, **dict**…). Il est également possible de passer en argument **d'autres fonctions**.

4 – RETOUR DE RESULTATS

L'instruction **return stoppe l'exécution** de la fonction et **retourne une ou plusieurs données**.

Exemple 3 : fonction « Calcul de la surface et du volume d'une sphère »

```
import math
def surface_volume_sphere(R) :
    # ---------------------------------------
    # cette fonction calcule et retourne à
    # partir du rayon R, la surface S et le
    # volume V d'une sphère
    # ---------------------------------------
    S = 4.0 * math.pi * R**2
    V = S * R / 3
    return S,V

# programme principal
rayon = float (input("Rayon (en cm):"))
s,v = surface_volume_sphere(rayon)
print("Sphère de rayon",rayon,"cm")
print("Sphère de surface",s,"cm²")
print("Sphère de volume",v,"cm3")

>>>
Sphère de rayon 25.0 cm
Sphère de surface 7853.981633974483 cm²
Sphère de volume 65449.84694978735 cm3
```

5 – PORTEE DES VARIABLES : VARIABLES LOCALES ET VARIABLES GLOBALES

La portée d'une variable dépend de **l'endroit du programme où on peut accéder à la variable**. Une **variable globale** est visible et utilisable dans **tout le programme**. Une **variable locale** est crée par une fonction et **n'est visible que par cette fonction**. Lors de la **sortie de la fonction**, la variable est **détruite**.

Exemple 4 : variable globale, variable locale

```
x = 10              # variable globale
def ma_fonction() :
    x = 20          # variable locale
    print("La variable locale est",x)

# programme principal
print("La variable globale est",x)
ma_fonction()

>>>
La variable globale est 10
La variable locale est 20
```

Bien que possédant, le même identifiant, les **deux variables x sont distinctes**.