1 Définition d'une fonction informatique

Fonctions

Une **fonction informatique** est un ensemble d'instructions regroupées sous un *nom* et s'exécutant à la demande (*l'appel* de la fonction). En Python, une fonction se compose :

- du mot clé **`def`**, suivi de l'identificateur de la fonction, de parenthèses entourant les paramètres de la fonction séparés par des virgules, et du caractère « **:** » qui termine *l'en-tête* de la fonction ;
- d'une **chaîne de documentation** (ou *docstring*), facultative, indentée comme le corps de la fonction ;
- du **bloc d'instruction indenté** par rapport à la ligne de définition et qui constitue le corps de la fonction.

Pseudo-code

```
1 Fonction nom_fonction(param1,param2,...):
2     instructions
3     renvoie result1, result2,...
```

Code Python :

```
1 def nom_fonction(param1,pa
  ram2,...):
2     """documentation de la
      fonction"""
3     instructions
4     return result1, result2
```

Remarque : Les fonctions sont des éléments structurants de base de tout langage procédural. Elles offrent différents avantages :

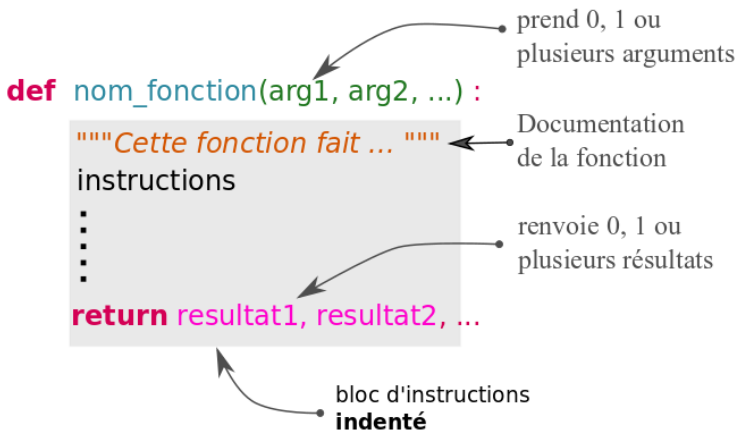
- elles évitent la répétition (factorisation de code) : on écrit le code d'une fonction une seule fois mais on peut appeler cette fonction plusieurs fois ;
- elles mettent en relief les données et les résultats : entrées et sorties de la fonction ;
- elles permettent la réutilisation : on peut importer et utiliser une fonction définie dans un autre module sans connaître les détails internes de sa programmation (avec l'instruction **`import`**)
- elles décomposent une tâche complexe en tâches plus simples : leur usage améliore la conception, l'écriture, la lecture, la correction et la modification de programmes

Les instructions sont regroupées dans un bloc **indenté** (décalé). Le mot **`return`** (optionnel) indique la fin de la fonction et fait immédiatement sortir du flux des instructions de la fonction. Les instructions ne sont exécutées que si la fonction est appelée. Une fonction acquiert tout son potentiel avec :

- une **entrée** qui regroupe des variables qui servent de **paramètres (ou arguments)** ;
- une **sortie** qui est un résultat renvoyé par la fonction ;

Il peut y avoir plusieurs fois l'instruction **`return`** dans une fonction mais une seule sera exécutée.
Une fonction peut avoir plusieurs paramètres (ou aucun) et renvoyer plusieurs résultats.

les valeurs renvoyées par une fonction peuvent être réutilisées ailleurs dans le programme : **on peut donc faire appel à une fonction dans une autre fonction**.



2 Simuler le hasard avec un programme informatique

L'informatique permet de simuler le hasard à l'aide de nombres pseudo-aléatoires. En langage Python, il faut importer la bibliothèque `random`, puis utiliser la fonction `random.randint` :

Code Python :

```
import random
lancer = random.randint(1,6)
```

La **variable** `lancer` contient un entier aléatoire entre 1 et 6.
On peut alors construire une **fonction** `lancer_un_de` qui va simuler le lancer d'un dé :

Code Python :

```
import random
def lancer_un_de():
    """simule le lancer d'un dé"""
    lancer = random.randint(1,6)
    return lancer
```

On pourra ensuite appeler cette fonction dans la console :

Console Python :

```
>>> lancer_un_de()
3
```

On peut construire une fonction `lancer_deux_des` qui simule le lancer de deux dés et renvoie la somme des faces obtenues. Il restera ensuite à simuler un grand nombre de lancers pour former un échantillon et calculer la fréquence des issues :

Code Python :

```
import random
def lancer_deux_des():
    """simule le lancer de deux dés et renvoie la somme des faces"""
    lancer = ..................................
    return lancer

def frequence_deux_des(issue, nb_tirages):
    """calcule la fréquence d'une issue dans un échantillon de taille
    nb_tirages"""
    effectif_issue = 0 # cette variable va contenir le nombre de fois où l'issue
    apparaît
    for i in range(nb_tirages): # boucle Pour qui répète nb_tirages fois les
    mêmes instructions
        tirage = ........................... # on lance les dés
        if ..................... :
            effectif_issue = .....................
    frequence_issue = .........................
    return ...............
```