Les fonctions

Lorsqu'on écrit un programme, on a besoin de fonctions diverses et variées.

- Les fonctions mathématiques arithmétiques (multiplication, addition,...) ou logiques (OU, ET, ...) classiques sont accessibles dans la bibliothèque de "base" de python et ne recquièrent aucune ligne de code supplémentaire pour y avoir accès.
- Les fonctions mathématiques plus sophistiquées sont inclues dans des **bibliothèques** spécifiques disponibles qu'il faut importer avant de pouvoir utiliser ces fonctions.
- On peut aussi avoir besoin de créer ses propres fonctions personnalisées.

Une bibliothèque est donc un ensemble de fonctions prédéfinies.

Bibliothèques externes disponibles

Une des grandes forces du langage Python réside dans le nombre important de bibliothèques logicielles externes disponibles. Celles-ci sont mises à disposition afin de pouvoir être utilisées sans avoir à les réécrire.

Quelques exemples :

- la librairie *math* contient entre autres les fonctions trigonométriques, la racine carrée, les recherches de PGCD, les factorielles ...
- la librairie *random* permet d'avoir accès à de nombreuses fonctions en rapport avec la génération de nombres aléatoires,
- la librairie *matplotlib* contient toutes les fonctions permettant de générer et de gérer l'affichage de graphiques,
- la librairie *numpy* contient de nombreux outils mathématiques (trigonométriques, tableaux,...) et permet entre autres de modéliser des ensembles de valeurs.

1. L'importation des bibliothèques doit se faire en tête de programme
2. Pour importer une bibliothèque, il suffit d'écrire la ligne : **import nom_de_la_bibliothèque**
3. L'accès à la fonction s'effectue ainsi : **nom_de_la_bibliothèque.nom_de_la_fonction**
4. On peut aussi donner un petit surnom à la bibliothèque par souci de simplification : **import nom_de_la_bibliothèque as surnom**
5. On peut aussi parfois ne vouloir importer qu'une fonction spécifique et non pas la totalité d'une bibliothèque. Il suffit alors d'écrire :
   from nom_de_la_bibliothèque import fonction. Attention dans ce cas, l'accés à la fonction s'effectue ainsi : **nom_de_la_fonction**

Quelques liens utiles...
https://docs.python.org/fr/3/library/math.html

https://docs.python.org/fr/3/library/random.html

In [1]:

```
# Importation de la bibliothèque math sans surnom
import math
print(math.cos(math.pi))  # permet d'avoir accès à la fonction cosinus ainsi qu'à la valeur␣
 ↪de pi et d'afficher le résultat
```

```
-1.0
```

In [2]:

```
# Importation de la bibliothèque random avec surnom
import random as rd
print(rd.randint(1,10)) # la fonction randint permet d'afficher à l'écran un nombre␣
 ↪aléatoire entier compris entre 1 et 10
```

```
7
```

In [3]:

```
# Importation de la seule fonction randint à partir de la bibliothèque random
from random import randint
print(randint(1,10))   # ne pas écrire random.randint() mais uniquement randint()
```

```
2
```

Fonctions personnalisées

Un programme écrit ń linéairement ż est peu lisible . On préfère en général le décomposer en plusieurs sous-programmes, nommés fonctions.

Une fonction est donc un ensemble d'instructions !

Les avantages de définir des fonctions sont multiples :

- le programme est plus lisible car architecturé, et se comprend plus facilement ;
- le code est réutilisable ;
- le programme est moins long : il suffit de définir une fonction pour effectuer une tâche précise, puis appeler cette fonction plusieurs fois si nécessaire ;
- le programme est plus facile à corriger et à améliorer.

Création d'une fonction

1. Les fonctions sont souvent écrites en tête de programme, après les imports de bibliothèques
2. La définition d'une fonction commence toujours par la ligne **def nom_de_la_fonction():** (ne pas oublier les : à la fin, erreur classique)
3. Les lignes de codes de cette fonction sont ensuite écrites en-dessous avec une indentation (décalage vers la droite).
4. Lorsqu'il a besoin de cette fonction, le programme principal (PP) doit l'appeler : **nom_de_la_fonction()**
5. Le PP peut avoir besoin d'échanger des informations avec la fonction.
6. Il est conseillé de documenter la fonction en utilisant la syntaxe `""" (...) """` juste après la déclaration de fonction

Les exemples ci-après sont progressifs. Les fonctions étudiées n'ont pas d'autre intérêt que d'illustrer les différents échanges possibles entre le PP et la fonction.

In [4]:

```
#Dans ce premier exemple, il n'y a pas d'échange d'informations entre le PP et la fonction
# Définition de la fonction nommée félicitations

def félicitations() :                # déclaration de la fonction
```