

Les fonctions

COMMENT MIEUX ORGANISER SON CODE!

Les fonctions

Les fonctions permettent de décomposer un programme en petits blocs qui pourront être réutilisés dans diverses circonstances.

Chaque petit bloc représente habituellement une tâche qui doit être accomplie pour arriver à l'objectif du programme.

Par exemple:

- Calculer la moyenne
- Déterminer la pente d'une droite
- Calculer la factorielle d'un nombre.

Vous avez utilisé plusieurs fonctions déjà fournies par python! Par exemple:

- `print(message)`
- `round(nombre, décimales)`
- `len(string)`

Les fonctions

- Une fonction ne doit être définie qu'une seule fois : on écrit ses instructions à un seul endroit dans un programme, et on peut par la suite l'appeler à toute occasion où il sera jugé nécessaire de le faire
- Les fonctions sont la base de la **réutilisabilité des programmes**, on définit une fonctionnalité une fois et on la réutilise à plusieurs endroits, ce qui permet de **minimiser la duplication** du code
- Il est même tout à fait possible de construire des "**bibliothèques**" : un **regroupement de fonctions** que l'on peut utiliser dans n'importe quel autre programme par la suite.

Les indications de types

LE MODE STRICT

Les indications de types

- Les indications de types sont un mécanisme dans le langage Python qui permet de renforcer les validations de types qui sont effectuées durant l'exécution d'un programme.
- On parle ici de vérifier, à chaque affectation, que le type de la valeur affecté à une variable correspond à celui qui a été choisi au moment de la définition de la variable
- On applique aussi les mêmes vérifications lors des appels de fonctions, au niveau des paramètres de la fonction appelée et aussi de la valeur retournée par les fonctions. (voir diapo 7)

Exemples d'indications de types

```
# variable de type str (chaîne de caractères)
```

```
nom : str = "Julie"
```

```
# Avertissement que la chaîne "pipo" est un type incompatible avec la variable nombre_entier
```

```
nombre_entier : int = "pipo"
```

```
# variable de type float
```

```
moyenne : float = 0.0
```

```
# nouvelle liste de float
```

```
liste_notes : list[float] = []
```

```
# Avertissement que la chaîne "papi" est un type incompatible avec la liste (float)
```

```
liste_notes.append("papi")
```

Il est important de noter ici que les indications de types ne sont que des **indications**. Elles permettent aux outils de validation des environnements de développement de **détecter certaines erreurs** de programmation, mais **n'empêcheront en rien l'exécution** du programme lorsque ces erreurs sont détectées

::::

Définition d'une fonction

les fonctions sont composées de plusieurs parties, regardons un exemple:

```
def bonjour(nom : str, age : int) -> None:  
    print(f"Bonjour {nom}, tu as {age} ans!")
```

La fonction `bonjour()` permet d'afficher un message de bienvenue à l'utilisateur indiquant son **nom** et son **âge**

Partie	définition
<code>def</code>	mot-clé qui indique le début de la définition d'une nouvelle fonction
<code>bonjour()</code>	Est le nom de la fonction , c'est lui que l'on va utiliser pour l'appeler
<code>nom : str,</code> <code>age : int</code>	Ce sont les paramètres de la fonction, ce sont les données qui seront utilisées par les instructions à l'intérieur de la fonction. Ils sont composés comme ceci: <code>nom_parametre : type</code>
<code>-> None</code>	Définit le type de retour, si la fonction doit retourner une valeur! Ici None indique qu'il n'y aura pas de valeur de retour. (À noter que ceci est une indication!)

Exemple : addition

Regardons un exemple de fonction qui permet d'additionner deux entiers et qui retourne le résultat de l'opération:

```
def addition(a : int, b : int) -> int:
    resultat : int = a + b # on additionne a et b et insère le résultat dans la variable "resultat"
    return resultat # on retourne la valeur de resultat

# collecte des données
premier_nombre : int = int(input("Premier nombre: "))
deuxieme_nombre : int = int(input("Deuxième nombre: "))

# Appel de la fonction "addition"
somme : int = addition(premier_nombre, deuxieme_nombre)

# sortie en console
print(f"La somme de {premier_nombre} et {deuxieme_nombre} est {somme}!")
```

Notez qu'en Python, les **fonctions** doivent être **définies avant** leur utilisation

(les définitions des fonctions se trouvent avant leur utilisation dans le fichier)