

[{m.  aly}]

# Python Niveau 2

Aly Tall NIANG

# Les Fonctions

# Les fonctions

Les fonctions sont très utiles en programmation. Elles nous permettent de :

- Faire une même opération plusieurs fois sans pour autant réécrire le code plusieurs fois
- Réduire le nombre de lignes de codes de nos programmes
- Organiser notre script
- Réutiliser des blocs de codes
- ...

On a déjà utilisé des fonctions prédéfinies comme : ***print()***, ***type()***, ***is()***, ***input()***,...

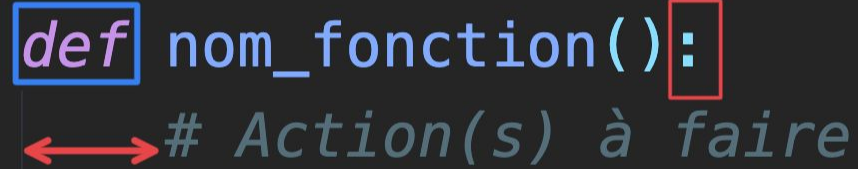
On peut **définir** sa propre fonction. Il existe deux types de fonctions:

- **Les fonctions sans retour de valeur** : *elles exécutent le code qu'elles contiennent.*
- **Les fonctions avec retour de valeur** : *elles exécutent le code qu'elles contiennent puis retourne un **résultat**.*

# Les fonctions

Définir une fonction (Syntaxe) :

```
def nom_fonction():  
    # Action(s) à faire
```

A diagram illustrating the syntax for defining a function in Python. The code is shown on a dark background. The keyword 'def' is enclosed in a blue box. The function name 'nom\_fonction()' is in a light blue font. The colon ':' at the end of the first line is enclosed in a red box. A red double-headed arrow points from the space between the function name and the colon to the comment '# Action(s) à faire' on the second line, indicating that the function body follows the colon.

# Les fonctions

Définir une fonction (Syntaxe) :

```
def nom_fonction():  
    # Action(s) à faire
```

*Remarque : les parenthèses sont obligatoires même si c'est vide (pas de parametres pour le moment)*

# Les fonctions

- **Retourner une valeur :**

- Pour retourner une valeur dans une fonction, on utilise le mot-clé **return**.
- l'instruction return termine l'exécution de la fonction (les lignes après ne seront pas exécutées)
- Par défaut toutes les fonctions retournent une valeur. En effet python retourne la valeur **None** pour les fonctions sans retour.

- **Appel de fonction :**

- Pour appeler une fonction (exécuter son code), on écrit le nom de la fonction avec les parenthèses
- Si la fonction a une valeur de retour, on peut l'appeler dans les cas suivants:
  - Expression de calcul
  - Expression d'affichage
  - Condition
  - Expression d'affectation

**Exemple :** Écrire un programme qui permet d'afficher "Bonjour" avec une fonction sans retour puis avec une fonction avec retour.

# Les fonctions

```
1 # Definition de la fonction sans retour
2 def dire_bonjour_1():
3     print("Bonjour tout le monde")
4
5 # Appel de la fonction sans retour
6 dire_bonjour_1()
7 # -----
8 # Definition de la fonction avec retour
9 def dire_bonjour_2():
10     return "Bonjour tout le monde"
11
12 # Appel de la fonction avec retour
13 print(dire_bonjour_2())
```

# Les fonctions

- **Paramètres (arguments) de fonction:**

- Les paramètres de fonction nous donnent la possibilité de donner nos propre valeur a une fonction.
- Les paramètres sont les valeurs dont la fonction à besoin pour faire son travail.

**Exemple :** Écrire une fonction qui affiche “Bonjour” suivi d’un nom passé en paramètre.

```
1
2 def dire_bonjour(nom):
3     print(f"Bonjour {nom}")
4
5
6 name = input("Entrer votre nom :\n")
7 dire_bonjour(name)
8
```

*Paramètre* (pointing to `nom` in line 2)

*Argument* (pointing to `name` in line 7)



# Les fonctions

- Paramètres par défaut:

- Dans l'exemple précédent, on ne peut pas appeler la fonction ***dire\_bonjour()*** sans lui donner d'argument (le paramètre est dit **obligatoire**).
- Un paramètre peut être optionnel, dans ce cas, lors de l'appel sans argument, une valeur par défaut est prise en compte.

## Exemple :

```
1
2 def dire_bonjour(nom="anonyme"):
3     print(f"Bonjour {nom}")
4
5
6 name = input("Entrer votre nom :\n")
7 dire_bonjour(name)
8
```

Paramètre par défaut

# Les fonctions

- **Ordre des paramètres :**

- Une fonction peut avoir plusieurs paramètres. Lors de l'appel d'une fonction, on peut respecter ou pas l'ordre des arguments.
- Si les paramètres ne sont pas nommés alors l'ordre doit être respecté.

Exemple :

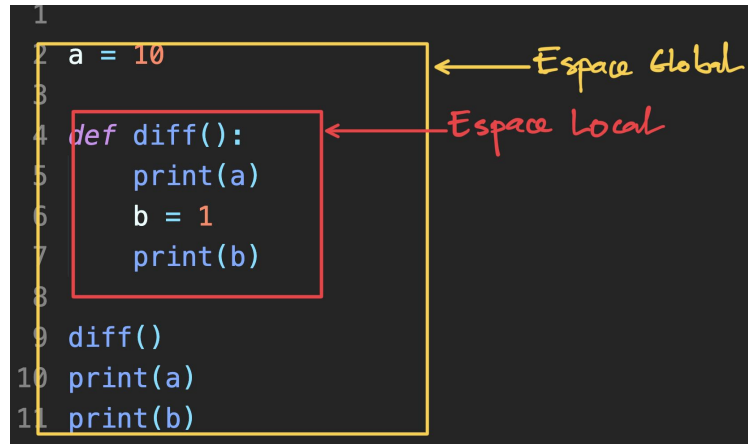
```
1
2
3
4 def diff(a,b):
5     return a-b
6
7 resultat = diff(12,2)
8 print(resultat) # affiche : 10
9 resultat = diff(b=12,a=2) ← Paramètres nommés
10 print(resultat) #affiche : -10
```

# Les fonctions

- **Variable locale et globale :**

- On distingue deux types d'espaces en python :
  - Espace global : toutes les lignes de codes non indentées. les variables de cet espace sont accessibles dans les fonction mais sont pas modifiées
  - Espace local : les lignes de codes écrites dans des fonctions. les variables de cet espace sont accessibles uniquement dans ce espace

Exemple :



```
1
2 a = 10
3
4 def diff():
5     print(a)
6     b = 1
7     print(b)
8
9 diff()
10 print(a)
11 print(b)
```

← Espace Global

← Espace Local