D-CLIC, FORMEZ-VOUS AU NUMÉRIQUE AVEC L'**OIF**

« Programme D-CLIC de l'Organisation internationale de la Francophonie »

Semaine 2 – Variables, opérations et fonctions en Python

Objectifs pédagogiques

À la fin de cette semaine, les apprenants sauront :

1. Identifier et utiliser les différents types de variables en Python.
2. Manipuler les opérations arithmétiques, de comparaison et logiques.
3. Créer et utiliser des fonctions, y compris les fonctions anonymes (*lambda*).
4. Appliquer ces notions dans de petits scripts d'analyse.

1. Variables et opérations

1.1. Types de variables en Python

Python reconnaît plusieurs types de données de base :

Type	Description	Exemple
int	Nombre entier	`3`, `-12`, `0`
float	Nombre décimal	`2.5`, `-0.1`
string	Chaîne de caractères	`"Pierre"`, `'Bonjour'`
bool	Booléen (Vrai/Faux)	`True`, `False`

Exemple en code :

```
x = 3                    # type int
y = 2.5                  # type float
prenom = "Pierre"        # type string
z = True                 # type bool
```

💡 **Astuce** : Python déduit le type de variable automatiquement selon la valeur que vous lui assignez.

1.2. Opérations arithmétiques

En Python, on peut effectuer toutes les opérations de base :

```
print("x + y =", x + y)       # addition
print("x - y =", x - y)         # soustraction
print("x / y =", x / y)         # division
print("x // y =", x // y)       # division entière
print("x * y =", x * y)        # multiplication
print("x ** y =", x ** y)      # puissance
```

📌 **Remarque** :

- `//` (modulo) retourne uniquement la partie entière de la division.

- `**` permet de calculer une puissance (`x**y` = x puissance y).

1.3. Opérations de comparaison

Ces opérations renvoient un booléen (`True` ou `False`) :

```
print("égalité :", x == y)
```
Vérifie si x est exactement égal à y.
Renvoie True si c'est le cas, sinon False.

```
print("inégalité :", x != y)
```
Vérifie si x est différent de y.
Renvoie True si les deux valeurs ne sont pas égales.

```
print("inférieur ou égal :", x <= y)
```
Vérifie si x est plus petit ou égal à y.
Renvoie True si c'est vrai.

```
print("supérieur ou égal :", x >= y)
```
Vérifie si x est plus grand ou égal à y.
Renvoie True si c'est vrai.

1.4. Opérations logiques

Permettent de combiner des conditions :

```
print("ET :", False and True)     # renvoie True si les deux sont vrais
print("OU :", False or True)      # renvoie True si au moins un est vrai
print("OU exclusif :", False ^ True)  # True si un seul est vrai
```

Utilité : Les opérations de comparaison et de logique utilisées ensemble permettent de construire des structures algorithmiques de bases (if/esle, while, ...)

2. Fonctions

2.1. Définir une fonction

Une fonction regroupe un bloc d'instructions que l'on peut réutiliser :

```
def ma_fonction(a, b):
    return a + b

resultat = ma_fonction(3, 5)
print(resultat)  # Affiche 8
```

2.2. Fonctions anonymes (*lambda*)

Une **fonction lambda** en Python est une **petite fonction anonyme** (c'est-à-dire **sans nom**) qu'on écrit sur **une seule ligne**. Ces fonctions courtes sont souvent utilisées pour des opérations simples :

Syntaxe :

lambda arguments: expression

```
carre = lambda x: x ** 2
print(carre(4))  # Affiche 16

addition = lambda a, b: a + b
print(addition(3, 4))  # Affiche 7

est_pair = lambda x: x % 2 == 0
print(est_pair(4))  # True
print(est_pair(7))  # False
```

Quelques fonctions simples

1- Calculer le carré d'un nombre

```
def carre(x):
    return x ** 2

print(carre(5))  # 25
```

2- Additionner deux nombres

```
def addition(a, b):
    return a + b

print(addition(3, 7))  # 10
```

3- Vérifier si un nombre est pair

```
def est_pair(n):
    return n % 2 == 0

print(est_pair(4))  # True
print(est_pair(5))  # False
```

4- Trouver la longueur d'un mot

```
def longueur_mot(mot):
    return len(mot)

print(longueur_mot("Python"))  # 6
```

💡 **Exemple pratique** : Trier une liste de tuples selon un critère.

```
eleves = [("Pierre", 12), ("Marie", 15), ("Jean", 10)]
# Trier par note
eleves_tries = sorted(eleves, key=lambda x: x[1])
print(eleves_tries)
```

3. Exercices pratiques

Exercice 1 – Calcul simple

Demandez à l'utilisateur deux nombres et affichez :

- leur somme
- leur produit
- le plus grand des deux

Exercice 2 – Analyse de texte simple

Écrire une fonction qui prend une phrase et retourne le nombre de mots.

4. Mini-projet – Calculateur d'IMC

Objectif : écrire une fonction qui calcule l'IMC (*Indice de Masse Corporelle*).

```
def imc(poids, taille):
    return poids / (taille ** 2)

p = float(input("Entrez votre poids en kg : "))
t = float(input("Entrez votre taille en m : "))
print("Votre IMC est :", imc(p, t))
```

Erreurs fréquentes à éviter

1. **Confondre `=` et `==`**
    - `=` sert à **affecter** une valeur à une variable.
    - `==` sert à **comparer** deux valeurs.

```
x = 5      # affectation
print(x == 5)  # comparaison → True
```

2. **Oublier les parenthèses pour appeler une fonction**

- `ma_fonction` → référence à la fonction
- `ma_fonction()` → exécution de la fonction

```
def bonjour():
  print("Salut !")

bonjour()  # OK
bonjour    # Ne s'exécute pas
```

3. **Mettre des majuscules au mauvais endroit pour les booléens**
   - En Python, on écrit `True` et `False` (avec T et F majuscules).

```
est_vrai = True   # OK
est_vrai = true   # Erreur !
```

4. **Oublier le `return` dans une fonction qui doit renvoyer une valeur**

```
def carre(x):
    x ** 2  # Mauvais : pas de return
```

5. **Mélanger types de données sans conversion**
   - Erreur : additionner un entier et une chaîne.

```
age = 25
print("J'ai " + age + " ans")  # Erreur
# Correction :
print("J'ai " + str(age) + " ans")
```

Quiz rapide – 5 questions

Q1. Que va afficher ce code ?

```
x = 3
y = 3
print(x == y)
```

- a) 3

- b) `True`

- c) `False`

Q2. Que fait l'opérateur `//` ?

- a) Division normale

- b) Division entière

- c) Multiplication

Q3. Quelle est la bonne syntaxe pour une fonction qui renvoie le triple d'un nombre ?

- a) `def triple(x): return x * 3`

- b) `def triple(x) -> x * 3`

- c) `triple = lambda x { x * 3 }`

Q4. Vrai ou faux :
En Python, `True` et `False` s'écrivent sans majuscule.

Q5. Quel est le résultat de :
print(5 != 3)

- a) `True`

- b) `False`

Formations aux métiers du numérique avec l'OIF en :
Développement Web - Développement mobile - Marketing numérique
Culture numérique - Cybersécurité - Intelligence Artificielle
Cyber Academy - Gouvernance du numérique
(+228) 719 987 05 dclicoiftogo@gmail.com

ORGANISATION INTERNATIONALE DE la francophonie

D-CLIC
FORMEZ-VOUS AU NUMÉRIQUE !