Variables et Types en Python

Une variable est une boîte étiquetée dans la mémoire de l'ordinateur qui stocke une valeur. Vous donnez un nom à la boîte et y mettez une valeur. Python se souvient de cette association.

Analogie : Imaginez des tiroirs étiquetés dans un bureau :
Étiquette (nom) = age
Contenu (valeur) = 19

```
nom = "Alice"
_age = 25
note1 = 15.5
prenom_utilisateur = "Bob"
```

```
1nom = "Alice"        # Ne peut pas commencer par un chiffre
mon-nom = "Alice"     # Pas de tiret
nom utilisateur = "Bob"   # Pas d'espace
```

Bonnes pratiques :

- ❑ Utilisez des noms descriptifs : age plutôt que a
- ❑ Majuscules/minuscules : Age, AGE et age sont 3 variables différentes !
- ❑ Convention : en Python, on utilise le snake_case : ma_variable

Les 4 types de base essentiels

1. Booléen (bool) - Vrai ou Faux

```
est_actif = True     # Commence par une majuscule !
est_complete = False
# Utile pour les tests : if est_actif: ...
```

2. Entier (int) - Nombre entier

```
age = 25
nombre_etudiants = 150
temperature = -5
```

3. Réel (float) - Nombre à virgule

```
prix = 19.99
taux_tva = 0.20
pi = 3.14159
# Le point pour la virgule (anglais) : 3.14, pas 3,14
```

4. Chaîne (str) - Texte

```
prenom = "Gawonou"
message = 'Bonjour tout le monde'
adresse = """123 rue de Paris
75001 Paris"""
```

 # Guillemets simples ou doubles, mais restez cohérent !

Afficher du texte avec print()

Affichage simple :

```
nom = "Gawonou"
age = 25
print("Bonjour")  # Affiche : Bonjour
print(age)        # Affiche : 25
print(prenom)     # Affiche : Gawonou
```

Affichage avec f-strings (le plus simple !)

```
nom = "Gawonou"
age = 25
ville = "Lome"

# Placez un 'f' avant les guillemets, puis {variable}

print(f"Je m'appelle {nom}")
print(f"J'ai {age} ans et je vis à {ville}")
# Résultat : Je m'appelle Gawonou, j'ai 25 ans et je vis à Lome

# Calcul dans la f-string
print(f"Dans 10 ans, j'aurai {age + 10} ans")
```

À retenir

Type	Exemple	Usage
int	42, -10	Âge, compteur
float	3.14, -0.5	Prix, moyenne
str	"Bonjour"	Noms, messages
bool	True, False	Tests, conditions

- **Une variable = un nom + une valeur**
- **Python devine le type** automatiquement
- **Utilisez f-strings** pour afficher des variables dans du texte
- Pratiquez ! Créez vos propres variables et affichez-les

```
age = "25"      # C'est du TEXTE
age = 25        # C'est un NOMBRE

print(f"J'ai {age + 5} ans")  # Fonctionne avec 25, pas avec "25" !
```

Demo

```
# 1. Création des variables
nom_utilisateur = "Kokou"
age_utilisateur = 30
solde_bancaire = 1250.75
est_inscrit = True

# 2. Affichage des valeurs
print("=== FICHE PERSONNELLE ===")
print(f"Nom : {nom_utilisateur}")
print(f"Âge : {age_utilisateur} ans")
print(f"Solde : {solde_bancaire} €")
print(f"Inscrit : {est_inscrit}")

# 3. Petite démonstration
print("\n=== CALCULS ===")
print(f"Dans 5 ans, {nom_utilisateur} aura {age_utilisateur + 5} ans")
print(f"La moitié de son solde est {solde_bancaire / 2} fcfa")
```

Formatage Scientifique en Python

Syntaxe de base :

```
nombre = 1500000
print(f"{nombre:e}")    # 1.500000e+06
```

Avec précision :

```
print(f"{nombre:.2e}")  # 1.50e+06 (2 chiffres)
print(f"{nombre:.4e}")  # 1.5000e+06 (4 chiffres)
```

FORMATAGE AVANCÉ

<15 = ALIGNEMENT À GAUCHE

```
print(f"{'Nom':<15} {'Âge':<15}")
print(f"{'Marie':<15} {'12 ans':<15}")
print(f"{'Thomas':<15} {'10 ans':<15}")
```



```
Nom             Âge
Marie           12 ans
Thomas           10 ans
```

Tableau récapitulatif

Symbole	Signification	Mnémonique
<	Alignement à GAUCHE	Left (commence par L comme <)
>	Alignement à DROITE	Right (commence par R comme >)
^	Alignement au CENTRE	Center (C'est au milieu)

Saisie utilisateur avec input()

```
# input() demande à l'utilisateur de taper quelque chose
# et retourne ce qu'il a tapé comme TEXTE (str)
score1 = input("Entrez le score 1 : ")  # Exemple : 12
score2 = input("Entrez le score 2 : ")  # Exemple : 36
```

ATTENTION : input() retourne toujours du texte, même si on tape des chiffres !

```
# Avec input(), score1 et score2 sont du TEXTE
score1 = input("Entrez le score 1 : ")  # "12" (texte)
score2 = input("Entrez le score 2 : ")  # "36" (texte)

# Concaténation de texte
total_texte = score1 + score2
print(total_texte)  # "1236" et non 48 !
```

Explication : "12" + "36" = "1236" (comme "Bon" + "jour" = "Bonjour")

Solution : Conversion de types

```
# Étape 1 : Récupérer le texte
score1 = input("Entrez le score 1 : ")   # "12"
score2 = input("Entrez le score 2 : ")   # "36"

# Étape 2 : Convertir en nombres
nombre1 = int(score1)   # Convertit "12" en 12
nombre2 = int(score2)   # Convertit "36" en 36

# Étape 3 : Calculer
total = nombre1 + nombre2   # 12 + 36 = 48
print(f"Total : {total}")   # Total : 48
```

Conversion directe :

```
# Convertir immédiatement après l'input
score1 = int(input("Entrez le score 1 : "))  # Taper 12 → devient 12
score2 = int(input("Entrez le score 2 : "))  # Taper 36 → devient 36

total = score1 + score2  # Addition correcte
```

Récapitulatif des conversions

Fonction	Convertit en	Exemple
int()	Nombre entier	int("12") → 12
float()	Nombre décimal	float("12.5") → 12.5
str()	Texte	str(12) → "12"

Bonnes pratiques

- ❑ Toujours convertir les input() numériques
- ❑ Utiliser f-strings pour le formatage
- ❑ Nommer clairement les variables
- ❑ Afficher des messages clairs dans les input()
- ❑ Tester avec différentes entrées

Les mots clés

False	class	finally	is	return
None	continue	for	lambda	try
True	def	from	nonlocal	while
and	del	global	not	with
as	elif	if	or	yield
assert	else	import	pass	
break	except	in	raise	

Les Listes en Python

Une liste est une collection ordonnée d'éléments qui peut contenir différents types de données.

NomListe = [élément1, élément2, élément3, ...]

DEUX FAÇONS DE CRÉER UNE LISTE VIDE

```
# Méthode 1 : avec des crochets
vide1 = []
print(vide1)  # []

# Méthode 2 : avec la fonction list()
vide2 = list()
print(vide2)  # []
```

Exemple

```
# Liste de nombres
notes = [12, 15, 18, 10, 14]
nombres = [6, 2, 4, 5, 1]

# Liste de textes
prenoms = ["Alice", "Bob", "Charlie"]
mois = ["Janvier", "Février", "Mars", "Avril"]


# Liste mixte
melange = ["Python", 3.9, True, 42]
mixte = ["un", 1, "deux", 2]
```

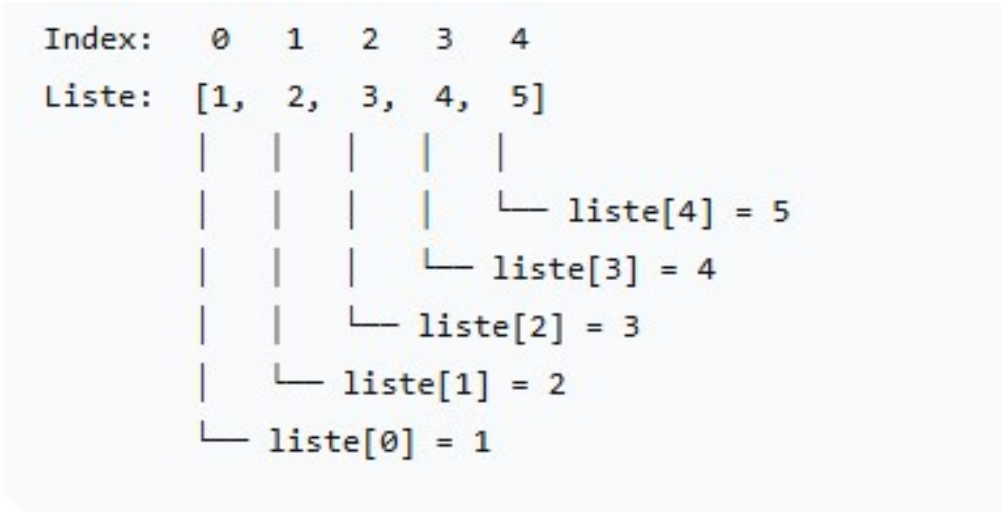
OPÉRATIONS DE BASE SUR LES LISTES

```python
nombres = [10, 20, 30, 40, 50]

print(nombres[0])   # 10 (premier élément)
print(nombres[2])   # 30 (troisième élément)
print(nombres[-1])  # 50 (dernier élément)
print(nombres[-2])  # 40 (avant-dernie
```

```python
nombres = [10, 20, 30, 40, 50]
nombres[2] = 99   # Remplace 30 par 99
print(nombres)    # [10, 20, 99, 40, 50]
```

Opérations de Slicing sur les Listes




```
liste = [1, 2, 3, 4, 5]
partie = liste[1:4]  # [2, 3, 4]
```

SYNTAXE GÉNÉRALE liste[début:fin:pas]

début : index de départ (inclus)
fin : index d'arrêt (exclus)
pas : incrément (par défaut = 1)

Opération	Résultat	Explication
liste[:]	[1, 2, 3, 4, 5]	Copie complète
liste[2:]	[3, 4, 5]	À partir de l'index 2
liste[:3]	[1, 2, 3]	Jusqu'à l'index 3 (exclu)
liste[1:4]	[2, 3, 4]	De l'index 1 à 4 (exclu)
liste[::2]	[1, 3, 5]	Un élément sur deux
liste[::-1]	[5, 4, 3, 2, 1]	Liste inversée

Opération	Résultat	Explication
liste[::-1]	[5, 4, 3, 2, 1]	Inverse la liste
liste[::-2]	[5, 3, 1]	Inverse + un sur deux
liste[3:0:-1]	[4, 3, 2]	De 3 à 0 en sens inverse

Exemple

ma_liste = ["a", "b", "c", "d", "e", "f"]
Qu'obtenez-vous avec ma_liste[1:5:3] ?

OPÉRATEURS SUR LES LISTES

Opération	Exemple	Résultat	Description
in	5 in [1,2,3,4,5]	True	Vérifie si élément présent
not in	10 not in [1,2,3]	True	Vérifie si élément absent
+	[1,2] + [3,4]	[1,2,3,4]	Concaténation
*	[0] * 5	[0,0,0,0,0]	Répétition

```
# Vérification d'appartenance
liste = [1, 2, 3, 4, 5]
print(3 in liste)      # True
print(10 in liste)     # False
print(10 not in liste)# True

# Concaténation
liste1 = [1, 2]
liste2 = [3, 4]
resultat = liste1 + liste2  # [1, 2, 3, 4]

# Répétition
zeros = [0] * 5  # [0, 0, 0, 0, 0]
```

FONCTIONS SUR LES LISTES

Fonction	Exemple	Résultat	Description
len()	len([1,2,3])	3	Nombre d'éléments
max()	max([1,5,2])	5	Plus grand élément
min()	min([1,5,2])	1	Plus petit élément

Méthode	Exemple	Effet	Description
.append(x)	liste.append(5)	Ajoute 5 à la fin	Ajoute un élément
.count(x)	liste.count(2)	Compte les 2	Compte occurrences
.index(x)	liste.index(3)	Position du 3	Trouve l'index
.clear()	liste.clear()	Vide la liste	Supprime tout
.copy()	copie = liste.copy()	Crée une copie	Copie la liste

Demo

```python
nombres = [1, 2, 3, 4, 5]

# Ajouter un élément
nombres.append(6)  # [1, 2, 3, 4, 5, 6]

# Compter les occurrences
nombres.append(3)  # Ajoute un autre 3
print(nombres.count(3))  # 2 (il y a deux 3)

# Trouver la position
print(nombres.index(4))  # 3 (le 4 est à l'index 3)

# Vider la liste
nombres.clear()  # []
```

```
#challenge avec une liste

videoGameChars = ["Mario", "Luigi", "Link", "Zelda", "Samus", "Ness", "Megaman", "Kid Icarus"]

1. #Ajoutez 2 Characters de plus dans la liste

2. #Printez le 4e item de la nouvelle liste

3. #Mettez la nouvelle liste en ordre alphabetique

4. #Printez le 5,6,7 item de la nouvelle liste

5. #Printez la nouvelle liste complete en sautant de 2
```

Les Tuples en Python

Un tuple est une collection ordonnée et immuable d'éléments.

Méthode 1 : Avec parenthèses

```
coordonnees = (48.8566, 2.3522)  # Latitude, Longitude
jours_semaine = ("Lundi", "Mardi", "Mercredi", "Jeudi", "Vendredi")

tuple1 = (1, 2, 3)
tuple2 = ("a", "b", "c")
tuple_vide = ()
```

Méthode 2 : Sans parenthèses (virgule obligatoire)

```
tuple3 = 1, 2, 3          # (1, 2, 3)
singleton = (4,)          # (4,) - NOTE : virgule nécessaire!
pas_un_tuple = (4)        # 4 (c'est un int, pas un tuple!)
```

Méthode 3 : Avec la fonction tuple()

```

tuple4 = tuple([1, 2, 3])     # Convertit une liste en tuple
tuple5 = tuple("abc")         # ('a', 'b', 'c')
```

ACCÈS AUX ÉLÉMENTS

```python
mois = ("Janvier", "Février", "Mars", "Avril", "Mai")

print(mois[0])      # "Janvier" (premier élément)
print(mois[2])      # "Mars" (troisième élément)
print(mois[-1])     # "Mai" (dernier élément)

print(mois[1:4])    # ("Février", "Mars", "Avril")
print(mois[:3])     # ("Janvier", "Février", "Mars")
print(mois[::2])    # ("Janvier", "Mars", "Mai")
```

Les Dictionnaires en Python

Un dictionnaire est une structure qui associe des **clés** à des **valeurs**.

```python
personne = {"nom": "Gawonou", "age": 25}
notes = {"math": 15, "physique": 18}


# Exemple : fiche d'étudiant
etudiant = {
    "nom": "Kokou",
    "age": 25,
    "cours": "Informatique",
    "notes": [15, 18, 12]
}
```

Clé = mot à chercher (ex: "nom")
Valeur = définition (ex: "Gawonou")

Les clés :

- ❑ Uniques : pas de doublons
- ❑ Immuables : chaînes, nombres, tuples
- ❑ Pas mutables : pas de listes

Les valeurs :

- ❑ Tous types autorisés
- ❑ Peuvent être modifiées
- ❑ Peuvent être en double

ACCÈS ET MODIFICATION

Accéder à une valeur :

```
personne = {"nom": "Kokou", "age": 25}
print(personne["nom"])  # "Kokou"
print(personne["age"])   # 25
```

Ajouter ou modifier :

```
personne["ville"] = "Lome"  # Ajoute
personne["age"] = 26         # Modifie
```

VÉRIFICATIONS

```
dico = {"a": 1, "b": 2, "c": 3}

# Vérifier si une clé existe
print("a" in dico)       # True
print("d" in dico)       # False
print("z" not in dico)   # True

# Longueur du dictionnaire
print(len(dico))   # 3
```

MÉTHODES PRINCIPALES

.keys() : Liste des clés

```
dico = {"a": 1, "b": 2}
print(dico.keys())  # dict_keys(['a', 'b'])
```

.values() : Liste des valeurs

```
print(dico.values())  # dict_values([1, 2])
```

.items() : Liste des paires (clé, valeur)

```
print(dico.items())  # dict_items([('a', 1), ('b', 2)])
```

EXEMPLE COMPLET : ÉTUDIANT

```python
# Création d'un dictionnaire
etudiant = dict()

# Ajout de données
etudiant["nom"] = "Gawonou"
etudiant["age"] = 25
etudiant["redoublant"] = False
etudiant["notes"] = {"math": 15, "physique": 18, "francais": 15}

# Affichage
print("Étudiant complet :", etudiant)
print("Nom :", etudiant["nom"])
print("Redoublant :", etudiant["redoublant"])
print("Note en physique :", etudiant["notes"]["physique"])

# Méthodes
print("Clés :", etudiant.keys())
print("Valeurs :", etudiant.values())
print("Items :", etudiant.items())
```

```
Étudiant complet : {'nom': 'Gawonou', 'age': 25, 'redoublant': False, 'notes': {'math': 11, 'physiqu
e': 18, 'francais': 15}}
Nom : Gawonou
Redoublant : False
Note en physique : 18
Clés : dict_keys(['nom', 'age', 'redoublant', 'notes'])
Valeurs : dict_values(['Gawonou', 25, False, {'math': 11, 'physique': 18, 'francais': 15}])
Items : dict_items([('nom', 'Gawonou'), ('age', 25), ('redoublant', False), ('notes', {'math': 11,
'physique': 18, 'francais': 15})])
```