# Le JEU pour REVISER les bases du PYTHON

Récapitulatif des cours présents dans le jeu • **Chapitre 1**



## Table des matières

# 1. Le logiciel Anaconda


Le logiciel **Anaconda** est une distribution d'applications de développement pour les langages de programmation Python et R.

Ce logiciel, libre et open source, vise à simplifier la gestion et l'apprentissage.

Télécharger le logiciel : https://www.anaconda.com/products/distribution

Ce logiciel propose les applications **Spyder** et **Jupyter**.

**Spyder** est un IDE (Environnement de Développement Intégré) est un outil de développement Python.

L'application intègre directement les bibliothèques scientifiques de manipulation de données, telles que NumPy, Matplotlib, IPython, SciPy, etc.



**Jupyter** est une application de Notebook sur ordinateur.

Un notebook permet de combiner, dans un même document, du texte, du code, de la visualisation de données, etc.

# 2. Les noms des variables en Python

Les noms des variables en Python peuvent contenir :

- des lettres minuscules
- des lettres majuscules
- des chiffres
- des tirets bas


Les variables ne doivent pas commencer par un **chiffre** et ne doivent pas être constitué d'**espace** (il est aussi conseillé de ne pas mettre d'espace dans les noms des fichiers de programmation).

Il faut aussi éviter d'utiliser un mot déjà utilisé par Python (print, for, if, min, max, from, import, while, etc.).

**Remarque :** Une variable doit toujours être écrite de la même façon :

- mavariable
- MaVariable
- MaVariable
- maVariable
- etc.  sont toutes des variables différentes.

*On nomme cela la « sensibilité à la casse ».Cette sensibilité ne s'applique pas qu'au Python, mais aussi à la plupart des langages de programmation, aux nommages de fichiers, etc.*

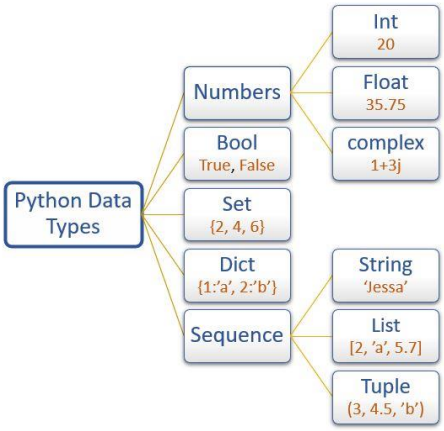
## 3. Les types en Python

Les principaux types de données :

| Les nombres | |
|---|---|
| Entier | int |
| A virgule (flottant) | float |

| Séquences | |
|---|---|
| Chaînes de caractères | str |
| Listes/Tableaux modifiables | list |
| Listes/Tableaux non-modifiables | Tuple |

| Autres | |
|---|---|
| Booléns | Bool |
| Dictionnaires | dict |




En savoir plus : https://courspython.com/types.html

## 4. Les chaînes de caractères

Les chaînes de caractères sont de type **str** en Python.

Les chaînes de caractères sont encadrées de guillemets ou d'apostrophes.

### Exemples :

```
"C'est une chaîne de caractères"

'Là aussi'
```

Pour écrire un guillemet dans une chaîne alors qu'ils encadrent cette chaîne,il faut placer un \ devant le guillemet (de même pour l'apostrophe).

## Exemple :

```
"C'est une chaîne de caractères avec des \"guillemets\" dedans."
```

Les indices dans une chaîne de caractères (de même pour une liste, un tuple, etc.) :

phrase = "NSI et SNT"  len(phrase)  # 10

| N | S | I | | e | t | | S | N | T |
|---|---|---|---|---|---|---|---|---|---|
| 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 |
| -10 | -9 | -8 | -7 | -6 | -5 | -4 | -3 | -2 | -1 |

## Quelques manipulations des chaînes de caractères :

*Pour notre exemple, la variable x est de type* ***str****.*

- `len(x)` → Retourne la taille de la chaîne de caractères.
- `x[i]` → Retourne la lettre à l'indice `i` dans la chaîne.
- `x[i:j]&` → Retourne la sous-chaîne de l'indice `i` à l'indice `j-1`.
- `x + "chaine"` → Retourne la chaîne `x` suivie de `"chaine"`
  Exemple : si `x = "la"`, on obtient `"lachaine"`.
- `x = ""` → Permet d'initialiser une chaîne vide.

## Exemple python :

```
ch = "chaine"          # initialisation
ch = ch + "s"
print(ch)              # Affiche "chaines"

taille = len(ch)
print(taille)          # Affiche 7

print(ch[2])           # Affiche "a"
print(ch[2:])          # Affiche "aines"
```

# 5. Les principales structures de données

| | | |
|---|---|---|
| Listes | int | Collection d'éléments ordonnés et mutable qui peut contenir plusieurs fois la même valeur |
| Tuples | tuple | Collection d'éléments ordonnés et non-mutable qui peut contenir plusieurs fois la même valeur. |
| Dictionnaires | Dict | Collection d'éléments non ordonnés mais indexés avec des clés et mutable qui n'accepte pas de contenir plusieurs fois le même élément |
| Ensembles | set | Collection d'éléments non ordonnés, non indexés et non-mutable qui n'accepte pas de contenir plusieurs fois le même élément |

Source : https://www.pierre-giraud.com/python-apprendre-programmer-cours/ensemble-set/#:~:text=Pr%C3%A9sentation%20des%20ensembles%20ou%20sets%20Python&text=Un%20ensemble%20est%20une%20collection,un%20autre%20type%20de%20donn%C3%A9es.

## Leur structure

| | | |
|---|---|---|
| Listes | Commence et termine par des crochets.<br>Les valeurs sont séparées par des virgules. | `li = [1, 2, 3, 4, 5]` |
| Tuples | Commence et termine par des parenthèses.<br>Les valeurs sont séparées par des virgules. | `tu = (1, 2, 3, 4, 5)` |
| Dictionnaires | Le dictionnaire commerce et termine toujours par des accolades.<br>Les clés et les valeurs sont associées par des deux-points « : ». Les duo clé/valeur sont séparés par des virgules. | `di = {"taille": 170,`<br>`"age": 25,`<br>`"yeux": "verts"}` |
| Ensembles | Commence et termine par des accolades.<br>Les valeurs sont séparées par des virgules. | `en = {1, 2, 3, 4, 5}` |

Principales manipulations et fonctions

| | Listes | Tuples | Dictionnaires | Ensembles |
|---|---|---|---|---|
| Taille de la structure | `len(L)` | `len(T)` | `len(D)` | `len(E)` |
| Sélectionner une valeur | `L[i]` | `T[i]` | `D[cle]` | |
| Sélectionner une partie | `L[i:j]` | `T[i:j]` | | |
| Concaténer deux structures | $L_1$ + $L_2$ | $T_1$ + $T_2$ | | |
| Ajouter une valeur x | `L.append(x)` | | `D[cle] = x` | `E.add(x)` |
| Modifier une valeur | `L[i] = y` | | `D[cle] = y` | |
| Supprimer une valeur | `L.pop(i)` ou `del L[i]` ou `L.remove(x)` | | `D.pop(cle)` ou `del D[cle]` | `E.remove(x)` |

Notes :

- `i` et `j` sont de type `int`
- `cle` est souvent de type `str`

## Exemples de codes

Les listes :

```
L = ["ab", "cd", "ef"]      # initialisation
L = L + ["g", "h"]
print(L)                    # Affiche ["ab", "cd", "ef", "g", "h"]

taille = len(L)
print(taille)               # Affiche 5

print(L[2])                 # Affiche "ef"
print(L[2:])                # Affiche ["ef", "g", "h"]
```

Il est possible de mettre une liste dans une liste:

```
L = [ [1, 2, 3], ["a", "b", "c"] ]
print(L[0])           # Affiche [1, 2, 3]
print(L[1])           # Affiche ["a", "b", "c"]
print(L[1][2])        # Affiche "c"
```

Les dictionnaires :

```
dico = {"taille": 170, "age": 24, "yeux": "verts"}

dico["age"] = dico["age"] + 1
print(dico)                     # Affiche {"taille": 170,
                                #       "age": 25,
                                #       "yeux": "verts"}

dico["cheveux"] = "court"
print(dico)                     # Affiche {"taille": 170,
                                #       "age": 25,
                                #       "yeux": "verts",
                                #       "cheveux": "court"}
```

*On observe que si la clé n'existe pas dans le dictionnaire, celle-ci est ajoutée.*

# 6. Les booléens et les opérations

Le type booléen ne comporte que deux valeurs : **vrai** et **faux** (en python, **True** et **False**).

Certaines opérations permettent d'obtenir une valeur booléenne :

**Les opérateurs :** Les variables `x` et `y` ont des valeurs quelconques.

| Opérateurs | Résultats | |
|---|---|---|
| | True | False |
| `x == y` | `x` et `y` sont égales | `x` et `y` sont différentes |
| `x != y` | `x` et `y` sont différentes | `x` et `y` sont égales |
| `x < y` | `x` est strictement inférieure à `y` | `x` est supérieure ou égale à `y` |
| `x > y` | `x` est strictement supérieure à `y` | `x` est inférieure ou égale à `y` |
| `x <= y` | `x` est inférieure ou égale à `y` | `x` est strictement supérieure à `y` |
| `x >= y` | `x` est supérieure ou égale à `y` | `x` est strictement inférieure à `y` |

**Les expressions booléennes :** Les variables `x` et `y` ont des valeurs booléennes.

| Expressions | Résultats | |
|---|---|---|
| | True | False |
| `x and y` | `x` **et** `y` sont égales à **True** | `x` **et/ou** `y` sont égales à **False** |
| `x or y` | `x` **et/ou** `y` sont égales à **True** | `x` **et** `y` sont égales à **False** |
| `not x` | `x` est égale à **False** | `x` est égale à **True** |

**Exemple python :**

```
print(333 == "333")              # Affiche False
print(True or False)             # Affiche True
print(12 > 15 and 0 < 99)        # Affiche False
```

## 7. Les calculs

Certaines opérations permettent de faire des calculs avec des nombres (int et float).

**Les opérateurs :** Les variables `x` et `y` ont des valeurs int ou float.

| Opérateurs | Résultats |
|---|---|
| `x + y` | Addition |
| `x - y` | Soustraction |
| `x * y` | Multiplication |
| `x ** y` | Exponentiation → $x^y$ |
| `x / y` | Division (float) |
| `x // y` | Division euclidienne (int) |
| `x % y` | Modulo (reste entier de la division euclidienne) |

## 8. Les conversions

Pour convertir le type d'une valeur, on utilise des fonctions nommées avec le nouveau type.

**Exemple python :**

```
print( str(333) )                # Affiche '333'
print( float("257") )            # Affiche 257.0
print( tuple( [1, 2, 3] ) )      # Affiche (1, 2, 3)
```

## 9. En savoir plus sur les variables et les types

- https://python.sdv.univ-paris-diderot.fr/02_variables/#:~:text=la%20variable%20x%20.-,2.2%20Les%20types%20de%20variables,caract%C3%A8res%20(string%20ou%20str)
- http://www.python-simple.com/python-langage/types.php
- Ou demande à ton/ta prof de NSI :)