**VEILLE VARIABLES**

**Qu'est-ce-qu'une variable?**

Une variable permet de stocker un élément de données et d'y accéder facilement dans un programme. Une variable est composée de trois éléments : nom, valeur et type. Les valeurs des variables peuvent être modifiées.

Le nom d'une variable doit refléter la signification de son contenu, comme des étiquettes sur une boîte. La nature de la variable est reflétée par son type : entier, décimal, caractère, chaine.

**Spécificités de Python :**

-typage dynamique

- le nom d'une variable ne dout pas commencer par un nombre et le caractère \_ doit être évité en premier caractère

**Lister les types de variables :**

int : entier

bool : booléen (True ou False)

float : nombre flottant qui a la précision d'un double

str : chaîne de caractère (string)

list : liste

complex : nombre complexe

**Lister les opérateurs de Python :**

**Opérateurs arithmétiques**

+ Addition

– Soustraction

\* Multiplication

/ Division

% Modulo

\*\* Puissance

// Division entière

**Lister les fonctions utilisées pour traiter les chaînes de caractères :**

+ Concaténation

\* Duplication

len() Longueur

.upper Mettre en majuscules

.lower Mettre en minuscules

.find() Rechercher la position d'une sous-chaîne

**VEILLE LISTES**

**Qu’est-ce qu'une liste en Python**

Les listes (ou list / array ) en python permettent d’organiser plusieurs variables dans une structure ordonnée et indexée dont la longueur peut varier.

Exemples :

[] (liste vide), [2, 4, 6] (liste d’entiers), ['ab', 2, 'cd', 5] (liste hétérogène), ['a', ['b', 'c'], 'd'] (liste contenant une autre liste)

**Lister les fonctions utiles**

L’indexation commence à 0 et les éléments de la liste sont accessibles à travers leur index.

Si l = [2, 4, 6], l[1] fait référence au 2ème élément de la liste, à savoir 4.

Concaténation de deux listes par l’opérateur +

l.append(x) : ajoute un élément x.

l.pop() : enlève le dernier élément et le renvoie.

l.pop(0) : enlève le premier élément et le renvoie.

l.pop(i) : enlève l'élément à la position i et le renvoie.

l.index(x) : renvoie l'index du 1er élément identique à x (erreur si non trouvé).

l.count(x) : compte combien de fois l'élément x est dans la liste.

l.insert(i, x) : insert l'élément i à la position x, et décale le reste (comme l[i:i] = x).

l.remove(x) : enlève le premier élément identique à x (erreur si non trouvé).

l.reverse() : renverse la liste en place.

l.sort() : trie en place les éléments de la liste.

l.sort(reverse = True) : effectue le tri dans l'autre sens.

sorted(l) : renvoie la liste triée, mais sans affecter la liste de départ (sorted peut prendre les mêmes options key et reverse que sort()).

sorted(l, myOrder) : quand on veut utiliser une fonction de tri personnalisée (comme avec sort)

reversed(l) : renvoie un itérateur qui permet de boucler sur les éléments de la liste en sens inverse.

**Comment parcourir une liste**

Une liste est un objet “itérable” que l’on peut donc utiliser dans une boucle.

Ici, je donner l’exemple d’une boucle de type for :

liste = ["a","d","m"]

**for** lettre **in** liste:

print lettre

**Explorer les listes de compréhension**

Une liste en compréhension permet de définir une liste à l’aide d’une boucle, l’itération aura lieu lors de l’affectation de la liste.

Exemple :

[x \*\* 2 for x in [1, 2, 3, 4]] : renvoie [1, 4, 9, 16] (liste de tous les carrés)

**List vs Tuples**

Un tuple est une liste qui ne peut plus être modifiée.

Les tuples peuvent être utilisés comme clefs dans les dictionnaires et prennent aussi un peu moins de mémoire que les listes.

Exemple :

MonTuple = ("a","d","m")

MonTuple[0] renvoie "a"

MonTuple[0] = "ok" déclenchera une erreur

**VEILLE DICTIONNAIRE**

**Qu’est-ce qu'un dictionnaire en Python**

Un dictionnaire en Python pourrait aussi s’appeler tableau associatif. Il est constitué d’un ensemble d’éléments sous forme de paires clé-valeur. La clé sert d’index pour retrouver la valeur.

**Comment créer un dictionnaire en Python**

Pour créer un nouveau dictionnaire, on utilise des accolades { } et on définit les paires clef : valeur à l’intérieur des accolades comme ceci :

mon\_dictionnaire = {"voiture": "véhicule à quatre roues", "vélo": "véhicule à deux roues"}

**Lister les opérations communes pour manipuler les dictionnaires**

Accéder à une valeur par l’index : mon\_dictionnaire["voiture"]

Créer une entrée : mon\_dictionnaire["tricycle"] = "véhicule à trois roues"

Créer un dictionnaire :

nombre\_de\_pneus = {}nombre\_de\_pneus["voiture"] = 4nombre\_de\_pneus["vélo"] = 2

[10]:

nombre\_de\_pneus

[10]:

{'voiture': 4, 'vélo': 2}

**Comment parcourir un dictionnaire**

On utilise ma méthode objet items()

Exemple pour une boucle for avec un indice i :

nombre\_de\_roues = {"voiture": 4, "vélo": 2, "tricycle": 3}

for i in nombre\_de\_roues.items(): print(i)

Ou encore

for cle, valeur in nombre\_de\_roues.items():

print("l'élément de clé", cle, "vaut", valeur)