2025

Créer par : RAWAA M’CHAABAT

HIBA MARZAK

Formateur : OTHMAN HAQQAY

03/05/2025



Rapport de python

**Définition du Collection en python :**

En Python, **une collection** (ou conteneur) est un objet qui permet de regrouper plusieurs éléments (valeurs, objets, etc.) sous une même variable. Les collections sont utilisées pour stocker, organiser et manipuler des données de manière efficace.

Principaux types de collections en Python :

1. **Listes (**list**) :**
   * Ordonnées et modifiables.
   * Permettent les doublons.
   * Exemple : [1, 2, 3]
2. **Tuples (**tuple**) :**
   * Ordonnés mais **non modifiables** (immutables).
   * Permettent les doublons.
   * Exemple : (1, 2, 3)
3. **Ensembles (**set**) :**
   * Non ordonnés, modifiables et **sans doublons**.
   * Exemple : {1, 2, 3}
4. **Dictionnaires (**dict**) :**
   * Stockent des paires **clé-valeur**.
   * Non ordonnés (avant Python 3.7), modifiables.
   * Clés uniques.
   * Exemple : {"clé": "valeur"}
5. **Chaînes de caractères (str) :**
   * Collections **immutables** de caractères.
   * Exemple : "Bonjour"

Définition de **Named Tuple** en python :

Un **namedtuple** est une sous-classe de tuple qui permet d'accéder à ses champs via des noms attribués, en plus des indices traditionnels.

Exemples d’exécution :

from collections import namedtuple

Personne = namedtuple('Personne', ['nom', 'age'])

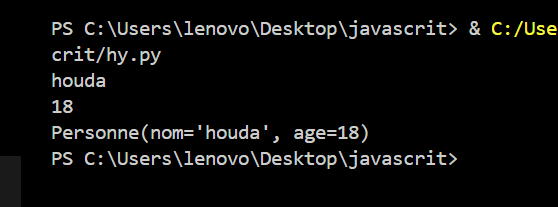
p = Personne(nom="houda", age=18)

print(p.nom)

print(p[1])

print(p)

L’affichage :



Définition de deque :

Une deque (double-ended queue) est une structure optimisée pour les ajouts/suppressions rapides aux deux extrémités. Plus efficace qu'une liste pour les opérations en début de séquence.

Exemples d’exécution :

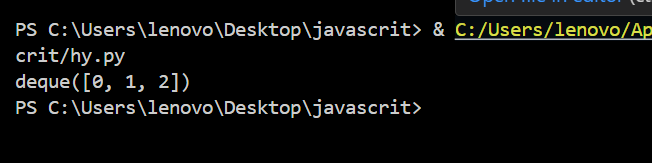
from collections import deque

d = deque([1, 2, 3])

d.appendleft(0)

d.pop()

print(d)

L’affichage : 

Définition de ChainMap :

Un**ChainMap**agrège plusieurs dictionnaires en une seule vue. Les recherches sont effectuées dans chaque dictionnaire jusqu'à trouver une clé correspondante.

Exemples d’exécution :

from collections import ChainMap

dict1 = {'a': 1, 'b': 2}

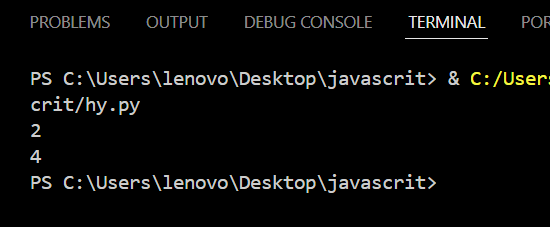
dict2 = {'b': 3, 'c': 4}

chain = ChainMap(dict1, dict2)

print(chain['b'])

print(chain['c'])

L’affichage :



Définition de OrderDict :

Un **OrderedDict** est un dictionnaire qui conserve l'ordre d'insertion des clés. Utile pour les versions de Python < 3.7 (où les dicts normaux ne le garantissaient pas).

from collections import OrderedDict

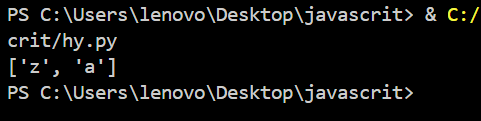
od = OrderedDict()

od['z'] = 1

od['a'] = 2

print(list(od.keys()))

L’affichage :



Définition de DefaultDict :

Un**defaultdict** est un dictionnaire qui fournit une valeur par défaut pour les clés absentes, évitant les erreurs KeyError .

Exemples d’exécution :

from collections import defaultdict

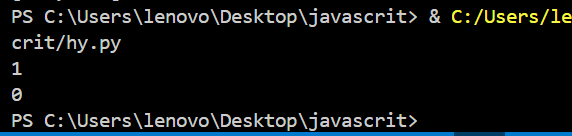
dd = defaultdict(int)

dd['a'] += 1

print(dd['a'])

print(dd['b'])

L’affichage :



Définition de JSON :

En Python, le module **json** permet de travailler facilement avec des données JSON .

Fonctions principales

1. json.dumps(): Convertit un objet Python en chaîne JSON
2. json.dump(): Écrit un objet Python dans un fichier au format JSON
3. json.loads(): Convertit une chaîne JSON en objet Python
4. json.load(): Lit un fichier JSON et le convertit en objet Python

Création d’un dictionnaire python :

import json

personne = {

    "nom": "Houda",

    "age": 18,

    "ville": "casablanca"

}

*Convertir en JSON (chaîne de caractères)*

json\_str = json.dumps(personne)

print("JSON string:", json\_str)

*Convertir JSON en dictionnaire Python*

python\_dict = json.loads(json\_str)

print("Dictionnaire Python:", python\_dict)

l’affichage :

