TD - Fonctions et Structures de données en Python

ali.zainoul.az@gmail.com

18 juin 2025

1. Fonctions en Python

Objectif : Savoir définir, appeler et exploiter les fonctions en Python, y compris les fonctions anonymes, générateurs et les fonctions avec arguments complexes.

Exemple :

```python
def greet(name="user"):
    return f"Hello, {name}!"
print(greet("Alice"))
```

Exercice 1 : *Fonctions simples avec paramètres et valeurs par défaut.*

— Créez une fonction qui retourne la somme, la moyenne et le produit de trois nombres.

— Utilisez des valeurs par défaut.

Exercice 2 : *Fonctions avec *args et **kwargs.*

— Créez une fonction qui accepte un nombre variable d'entiers et retourne leur max et min.

— Ajoutez une option nommée pour trier ou non.

Exercice 3 : *Fonctions anonymes et fonctions d'ordre supérieur.*

Utilisez `map()`, `filter()`, `lambda` pour filtrer les mots de plus de 5 lettres dans une liste de mots donnée par l'utilisateur.

Exercice 4 : *Générateurs.*

— Ècrivez un générateur qui produit les puissances successives d'un entier jusqu'à une borne.

— Testez avec `next()` et une boucle `for`.

Exercice 5 : *Espace de noms et fonction `dir()`.*

Créez un script qui définit une variable locale, une globale et une interne à une fonction. Utilisez `dir()` et `globals()` pour explorer les espaces de noms.

Exercice 6 : *Fonctions `eval()` et `exec()`.*

Demandez une opération mathématique à l'utilisateur, exécutez-la dynamiquement avec `eval()`, puis stockez la trace de l'exécution avec `exec()` dans une variable.

2. Types de données non-modifiables

Objectif : Comprendre les types immuables : `str`, `tuple`, `bytes`, `None`.

Exemple :

```
t = (1, 2, 3)
print(t[1])
print(len(t))
```

Exercice 7 : *Analyse de structure.*

— Déclarez un tuple contenant des str, float, et int.
— Testez `id()`, `hash()`, et l'opérateur `is`.

Exercice 8 : *Dépacking.*

— Créez une fonction qui retourne 4 valeurs.
— Dépackez les dans des variables individuelles et utilisez-les dans une autre fonction.

3. Types de données modifiables

Objectif : Savoir utiliser les listes, dictionnaires, sets et comprendre les références.

Exercice 9 : *Listes et copies.*

— Modifiez une liste dans une fonction, observez l'effet sur la liste d'origine.
— Testez avec une copie profonde.

Exercice 10 : *Fonction `sorted()`, `reversed()`, `range()`.*

Créez une fonction qui retourne les 10 premiers entiers pairs dans l'ordre inverse, sans utiliser de liste préconstruite.

Exercice 11 : *Manipulation de dictionnaires.*

— Créez un dictionnaire de mots-clés et de définitions.
— Affichez tous les mots triés par ordre alphabétique.
— Permettez la mise à jour ou la suppression d'entrées.

Exercice 12 : *Opérateurs sur les ensembles.*

— Créez deux ensembles.
— Appliquez les opérateurs `union`, `intersection`, `différence symétrique`.

4. Compréhension de collections

Objectif : Manipuler les intensions de liste, ensemble, dictionnaire et tuple.

Exercice 13 : *List comprehension avancée.*

— Générez une liste des carrés parfaits inférieurs à 100.
— Puis générez une liste contenant uniquement les carrés pairs.

Exercice 14 : *Dict et set comprehension.*

— Déclarez un dictionnaire associant des lettres aux carrés de leurs indices (a=0^2, b=1^2, ...).
— Générez un set des longueurs de mots donnés par un utilisateur.

Exercice 15 : *Tuple en intension (avec `tuple()`).*

— Générez un tuple contenant les puissances de 2 de 0 à 10.