Exercices Pratiques de Python : Types de données non-modifiables

Ali ZAINOUL

Exercices Pratiques de Python : Types de données non-modifiables

1. **Optimisation mémoire** : Écrivez un script Python qui crée une grande liste de chaînes de caractères et une grande liste de tuples. Comparez l'utilisation de la mémoire entre ces deux structures de données et expliquez les résultats.

2. **Fonctions `id()` et `hash()`** : Écrivez un programme qui crée plusieurs objets immuables (chaînes de caractères, tuples) et montre leurs identifiants uniques avec `id()`. Montrez également comment utiliser `hash()` pour créer un dictionnaire avec ces objets comme clés.

3. **Opérateur `is`** : Développez un script qui compare différents objets en utilisant l'opérateur `is`. Expliquez les différences entre `is` et `==` à travers des exemples.

4. **Séquences ordonnées** : Créez un programme qui prend une liste de chaînes de caractères, les trie par ordre alphabétique, puis les convertit en tuples ordonnés. Affichez le résultat et expliquez les avantages de l'utilisation de tuples immuables.

5. **Classe String (str)** : Écrivez un programme qui demande à l'utilisateur de saisir une chaîne de caractères, puis effectue plusieurs opérations sur cette chaîne (comme la recherche de sous-chaînes, la conversion en majuscules/minuscules, et le remplacement de certaines parties de la chaîne).

6. **Classe Tuple** : Développez un script qui crée plusieurs tuples imbriqués et montre comment accéder aux éléments individuels et aux sous-tuples. Expliquez comment les tuples peuvent être utilisés pour retourner plusieurs valeurs depuis une fonction.

7. **Fonction `range`** : Créez un programme qui utilise la fonction `range()` pour générer des séquences de nombres. Montrez comment `range()` peut être utilisé dans des boucles for et des compréhensions de listes.

8. **Slicing des chaînes de caractères** : Écrivez un programme qui demande à l'utilisateur d'entrer une chaîne de caractères et utilise le slicing pour extraire et afficher des sous-chaînes spécifiques (comme les premiers 5 caractères, les derniers 5 caractères, etc.).

9. **Slicing des tuples** : Développez un script qui crée un grand tuple de nombres et utilise le slicing pour extraire des sous-tuples spécifiques. Affichez les résultats et expliquez comment le slicing fonctionne avec les tuples.

10. **Immutable et mutable** : Créez un programme qui compare les performances de la modification d'une liste mutable et d'une chaîne de caractères immuable dans une boucle. Expliquez pourquoi les objets immuables peuvent être plus efficaces dans certains contextes.

11. **Concaténation de chaînes** : Écrivez un programme qui demande à l'utilisateur d'entrer plusieurs chaînes de caractères et les concatène en une seule chaîne en utilisant différentes méthodes (opérateur +, `join()`). Comparez les performances des différentes méthodes.

12. **Concaténation de tuples** : Créez un script qui montre comment concaténer plusieurs tuples en un seul tuple. Utilisez différentes méthodes et comparez leurs performances.

13. **Conversion de types** : Écrivez un programme qui convertit des listes en tuples et vice versa. Demandez à l'utilisateur de saisir une liste, convertissez-la en tuple, puis reconvertissez-la en liste.

14. **Fonctions de chaîne avancées** : Écrivez un script qui utilise des fonctions avancées de chaîne de caractères (comme `split()`, `strip()`, `replace()`) pour analyser et transformer une entrée utilisateur.

15. **Immuabilité et performances** : Développez un programme qui mesure le temps nécessaire pour effectuer des opérations sur des listes mutables et des tuples immuables. Comparez les résultats et expliquez pourquoi les objets immuables peuvent être plus performants dans certains cas.

16. **Recherche dans des séquences** : Créez un script qui prend une liste de tuples représentant des enregistrements (par exemple, des informations sur des étudiants) et permet à l'utilisateur de rechercher un enregistrement spécifique en utilisant des critères de recherche.

17. **Immutabilité et sécurité** : Écrivez un programme qui montre comment l'immutabilité peut améliorer la sécurité des données. Utilisez des tuples immuables pour stocker des informations sensibles et montrez comment ces informations ne peuvent pas être modifiées accidentellement.

18. **Projet final : Mini-processeur de texte** : Développez un mini-processeur de texte en Python qui permet à l'utilisateur de saisir, modifier et analyser des textes. Utilisez des chaînes de caractères pour le stockage et la manipulation des données, et incluez des fonctionnalités comme la recherche, le remplacement, le comptage de mots et de lignes.