Rapport de Projet : Gestion de Données avec les Tuples en Python

Introduction

Ce projet a pour objectif de pratiquer la manipulation des tuples et des listes en Python afin de gérer des données structurées. Nous avons divisé le projet en deux parties :

1. Gestion des informations étudiantes.
2. Gestion des personnages dans un jeu de rôle.

Chaque partie est accompagnée d'explications détaillées sur les résultats obtenus.

Partie A – Gestion des Informations Étudiants

L'objectif est de manipuler les données de plusieurs étudiants et de calculer la moyenne générale.

1. Création des tuples

Trois étudiants sont représentés par des tuples au format *(nom, âge, moyenne)* :

— Alice, 20 ans, moyenne : 15,5

— Bob, 22 ans, moyenne : 14,0

— Clara, 19 ans, moyenne : 16,2

Ces tuples sont stockés dans une liste `etudiants` :

```
etudiants = [('Alice', 20, 15.5), ('Bob', 22, 14.0), ('Clara', 19, 16.2)]
```

Explication : Cette structure permet d'accéder facilement à chaque étudiant via son indice et de manipuler ses informations sans modifier directement les tuples.

2. Récupération d'informations

2.1. Affichage du dernier étudiant : En utilisant l'indice `-1`, on accède à Clara. *Résultat attendu :* ('Clara', 19, 16.2) **Interprétation :** Le dernier élément de la liste est retourné, pratique pour accéder rapidement aux données les plus récentes.

2.2. Affichage de la moyenne du premier étudiant : On accède à l'indice 0 puis à la troisième valeur du tuple : 15,5. **Interprétation :** Permet de consulter rapidement la performance individuelle d'un étudiant.

2.3. Extraction des informations du deuxième étudiant (Bob) : On utilise le décompactage de tuple :

```
nom, age, moyenne = etudiants[1]
```

Résultat attendu : nom = Bob, age = 22, moyenne = 14,0 **Interprétation :** Le décompactage permet de récupérer simultanément toutes les informations dans des variables distinctes.

3. Ajout d'un étudiant

Un nouvel étudiant arrive : David, 21 ans, moyenne : 15,0. On l'ajoute à la liste avec `append()`.

```
etudiants.append(('David', 21, 15.0))
```

Explication : La liste peut croître dynamiquement, tandis que les tuples restent immuables. **Résultat attendu :** la liste contient désormais 4 tuples.

4. Calcul de la moyenne des étudiants

4.1. Création du tuple `Moyennes` : On extrait uniquement la troisième valeur de chaque tuple :

```
Moyennes = (15.5, 14.0, 16.2, 15.0)
```

Interprétation : On dispose ainsi d'un ensemble simplifié pour les calculs statistiques.

4.2. Calcul de la moyenne générale :

$$\text{Moyenne Générale} = \frac{15.5 + 14.0 + 16.2 + 15.0}{4} = 15.175$$

Interprétation : La moyenne générale permet d'évaluer la performance globale du groupe d'étudiants.

Partie B – Gestion des Personnages (Jeu de Rôle)

Dans cette partie, nous manipulons des tuples pour gérer des personnages et leurs points de vie dans un jeu.

1. Création des personnages

Trois personnages sont représentés par des tuples :

— Lila, mage, 100 points de vie

— Torn, guerrier, 150 points de vie

— Eryn, archer, 120 points de vie

Stockage dans la liste `personnages` :

```
personnages = [('Lila','mage',100),('Torn','guerrier',150),('Eryn','archer',120)]
```

Explication : Les tuples permettent de stocker les informations de manière compacte et immuable.

2. Modification des points de vie

Torn perd 20 points de vie. Comme un tuple est immuable, il faut créer un nouveau tuple et remplacer l'ancien :

```
personnages[1] = ('Torn', 'guerrier', 130)
```

Résultat attendu : Torn a désormais 130 points de vie. **Interprétation :** Même si un tuple ne peut pas être modifié, la liste permet de remplacer l'élément complet.

3. Affichage des informations

On affiche les informations de Torn sous un format lisible :

Nom du personnage : Torn, Classe : guerrier, Points de vie : 130

Explication : Le formatage rend la sortie lisible et facilement interprétable par un joueur ou pour un rapport.

Conclusion

Ce projet démontre l'utilité des tuples et des listes en Python pour la gestion de données structurées.

— Les tuples permettent de stocker des informations fixes et immuables.

— Les listes offrent la flexibilité d'ajouter, supprimer ou remplacer des éléments.

— Les opérations sur les listes et tuples permettent de récupérer facilement des informations, d'effectuer des calculs et de maintenir les données organisées.

La manipulation des structures de données présentées dans ce projet constitue une base solide pour des applications plus avancées, telles que la gestion de bases de données, les statistiques ou le développement de jeux.