Exercice : Travailler avec les dictionnaires en Python

Dans cet exercice, tu vas appliquer les notions sur les dictionnaires que tu viens de voir. L'objectif est de manipuler les dictionnaires pour gérer des données, les modifier et les parcourir.

Contexte

Tu travailles pour une entreprise de voyages et tu es chargé de gérer les informations concernant les différentes destinations disponibles, ainsi que les clients intéressés par ces destinations. Toutes les données doivent être stockées dans des dictionnaires.

Objectifs

1. Créer un dictionnaire contenant des informations sur plusieurs destinations (nom de la ville, nombre de réservations).
2. Gérer les réservations pour chaque destination.
3. Afficher les informations sur les destinations disponibles et les clients.
4. Appliquer des modifications sur les dictionnaires en utilisant des boucles et des méthodes propres aux dictionnaires.

Instructions

1. Créer un dictionnaire de destinations

Crée un dictionnaire `destinations` avec au moins 3 villes (clés) et le nombre de réservations associées (valeurs).

2. Ajouter des nouvelles destinations

Ajoute au moins deux nouvelles destinations avec un nombre initial de réservations égal à 0.

3. Mettre à jour les réservations

Écris une fonction `ajoute_reservations(destinations, ville, nombre)` qui prend en paramètre le dictionnaire des destinations, le nom de la ville, et le nombre de nouvelles réservations à ajouter. Mets à jour le dictionnaire en conséquence.

4. Supprimer une destination

Supprime une destination du dictionnaire. Fais en sorte que la destination supprimée soit renvoyée sous forme de tuple (clé, valeur).

5. Afficher les destinations disponibles

Affiche toutes les destinations et le nombre de réservations associées en parcourant le dictionnaire à l'aide d'une boucle `for`.

6. Trier les destinations

Trie les destinations par ordre de nombre de réservations, puis affiche le résultat.

7. Gérer les clients

Crée un dictionnaire `clients` avec pour clés les noms des clients et pour valeurs la destination qu'ils souhaitent réserver. Écris une fonction `associe_client(clients, nom_client, destination)` pour associer un client à une destination.

8. Afficher les clients et leurs destinations

Affiche chaque client ainsi que la destination qu'il a choisie en parcourant le dictionnaire `clients`.

Contraintes

- Utilise au moins une fois chacune des méthodes suivantes sur les dictionnaires : `.update()`, `.pop()`, `.popitem()`, `.keys()`, `.values()`, et `.items()`.
- Assure-toi de gérer les cas où une clé n'existe pas dans le dictionnaire (par exemple, une destination non valide ou un client non inscrit).

Exercice : Manipulation avancée des boucles et générateurs en Python

Exercice :

1. Créez un générateur `gen_carrés(n)` qui génère les carrés des nombres de 1 à `n`.
2. Parcourez les résultats du générateur avec une boucle `for` et affichez chaque carré, mais arrêtez la boucle dès que vous atteignez un carré supérieur à 50 grâce à `break`.
3. Utilisez `continue` pour ignorer les carrés pairs et n'affichez que les carrés impairs.
4. Parcourez une liste d'entiers en sens inverse de deux façons différentes.
5. Afficher à la fois l'index et la valeur de chaque élément d'une liste.

Exercice : Manipulation de chaînes de caractères

Écrire un programme qui prend une chaîne de caractères saisie par l'utilisateur et effectue les tâches suivantes :

1. Afficher la chaîne d'origine.
2. Afficher la longueur de la chaîne.
3. Afficher la chaîne avec tous les caractères en majuscules.

4. Afficher la chaîne avec tous les caractères en minuscules.
5. Afficher la chaîne inversée.
6. Remplacer toutes les occurrences d'un mot spécifique (saisies par l'utilisateur) par un autre mot (saisi par l'utilisateur).
7. Compter le nombre d'occurrences d'un mot spécifique (saisi par l'utilisateur) dans la chaîne.
8. Afficher la chaîne sans espaces en début et en fin.

Résultats attendus

- La chaîne d'origine affichée.
- La longueur de la chaîne.
- La chaîne en majuscules.
- La chaîne en minuscules.
- La chaîne inversée.
- La chaîne avec le mot remplacé.
- Le nombre d'occurrences du mot spécifié.
- La chaîne sans espaces en début et en fin.

Remarque : N'oubliez pas de gérer les cas où le mot à remplacer ou à compter n'est pas trouvé dans la chaîne.

Exercice : Gestion de Contacts

Objectif : Créer une application simple pour gérer une liste de contacts.

Étapes à suivre :

1. **Définir une classe `Contact` :**
    - La classe doit avoir des attributs pour le nom, le numéro de téléphone et l'email.
    - Ajouter une méthode pour afficher les informations du contact.
2. **Créer une fonction `ajouter_contact` :**
    - Cette fonction doit prendre en paramètres une liste de contacts et un contact, puis ajouter ce dernier à la liste.
    - La fonction doit vérifier que le contact n'existe pas déjà dans la liste (vous pouvez définir l'égalité des contacts par le nom ou un autre attribut).
3. **Créer une fonction `supprimer_contact` :**
    - Cette fonction doit prendre en paramètres une liste de contacts et un nom de contact, puis supprimer le contact de la liste.
4. **Créer une fonction `chercher_contact` :**
    - Cette fonction doit prendre en paramètres une liste de contacts et un nom, puis retourner le contact correspondant.
5. **Créer une fonction `afficher_contacts` :**
    - Cette fonction doit afficher tous les contacts dans la liste.

6. **Tests :**
    - Écrire des tests pour chaque fonction pour vérifier leur bon fonctionnement.
    - Utiliser des assertions pour s'assurer que les résultats sont conformes aux attentes.

Exercice : Gestion de la Bibliothèque

Vous devez créer un programme pour gérer une bibliothèque. Votre programme doit permettre à un utilisateur d'ajouter des livres, de les supprimer, de trier la liste des livres, et d'afficher les livres en fonction de certains critères.

Étapes à suivre :

1. **Créer une liste vide** pour stocker les titres des livres.
2. **Ajouter des livres** à la liste (demandez à l'utilisateur d'entrer le titre d'un livre). Assurez-vous que le titre n'est pas déjà dans la liste.
3. **Afficher la liste des livres**.
4. **Supprimer un livre** de la liste en demandant à l'utilisateur d'entrer le titre du livre à supprimer. Assurez-vous que le livre existe dans la liste avant de le supprimer.
5. **Trier la liste des livres** par ordre alphabétique.
6. **Afficher le nombre total de livres** dans la bibliothèque.
7. **Afficher les livres qui contiennent un mot spécifique** (demandez à l'utilisateur d'entrer un mot).
8. **Créer une fonction** qui prend en paramètre une liste de livres et qui renvoie une nouvelle liste contenant uniquement les livres qui commencent par une lettre spécifique (demandez à l'utilisateur d'entrer la lettre).
9. **Terminer le programme** lorsque l'utilisateur entre "exit".

Exercice : Gestion des Fichiers et JSON

Partie 1 : Gestion des Chemins

1. **Définissez un chemin de fichier pour votre système d'exploitation :**
    - Pour Windows : `chemin = "C:\Users\VotreNom\Documents\mon_fichier.txt"`
    - Pour macOS/Linux : `chemin = "/Users/VotreNom/Documents/mon_fichier.txt"`
    - Utilisez un préfixe `r` pour éviter les erreurs liées aux slashs.
2. **Affichez le chemin du fichier.**

Partie 2 : Lecture et Écriture de Fichiers

1. **Créez un fichier texte et écrivez-y trois lignes de texte.**
    - Utilisez le mode `ecriture` pour écrire dans le fichier.
2. **Ouvrez le fichier en mode lecture et lisez son contenu :**
    - Affichez le contenu à l'écran.
3. **Récupérez chaque ligne dans une liste et affichez cette liste.**

Partie 3 : Utilisation des Curseurs

1. **Après avoir lu le contenu du fichier, remettez le curseur au début du fichier.**
2. **Lisez de nouveau le fichier et affichez le contenu.**
3. **Affichez uniquement les 10 premiers caractères du fichier.**

Partie 4 : JSON

Créez un fichier JSON (`settings.json`) avec le contenu suivant :
json
Copier le code

```
{
  "fontsize": 20,
  "theme": "light"
}
```

1.
2. **Récupérez les données du fichier JSON et affichez la valeur de `fontsize`.**
3. **Modifiez `fontsize` à 15 et sauvegardez les modifications dans le fichier JSON.**
4. **Ajoutez un nouvel attribut `language` avec la valeur `"French"` à votre fichier JSON, puis sauvegardez-le.**
5. **Pour vérifier, rouvrez le fichier JSON et affichez son contenu.**

Exercice : Pratique des Modules os et pathlib

Objectif

Cet exercice a pour but de vous familiariser avec les opérations sur les fichiers et répertoires en utilisant les modules `os` et `pathlib`. Vous allez créer, lire, renommer, et supprimer des fichiers et répertoires.

Partie 1 : Création de fichiers et de répertoires

1. **Créer un fichier texte :**

    - Créez un fichier nommé `notes.txt` et écrivez-y trois lignes de texte.
2. **Créer un répertoire :**
    - Créez un répertoire nommé `mon_dossier`.

Partie 2 : Lecture et suppression

1. **Lire le contenu du fichier :**
    - Ouvrez `notes.txt` et lisez son contenu. Affichez chaque ligne.
2. **Supprimer le fichier :**
    - Supprimez le fichier `notes.txt`.
3. **Supprimer le répertoire :**
    - Supprimez le répertoire `mon_dossier`.

Partie 3 : Renommage et informations sur les fichiers

1. **Créer un nouveau fichier :**
    - Créez un fichier nommé `ancien_nom.txt` et écrivez-y une ligne de texte.
2. **Renommer le fichier :**
    - Renommez `ancien_nom.txt` en `nouveau_nom.txt`.
3. **Obtenir des informations sur le fichier :**
    - Affichez la taille et la date de dernière modification de `nouveau_nom.txt`.

Partie 4 : Utilisation de pathlib

1. **Créer un nouveau répertoire :**
    - Créez un répertoire `images`.
2. **Créer un fichier dans le répertoire :**
    - Créez un fichier `image.jpg` dans le répertoire `images`.
3. **Modifier l'extension du fichier :**
    - Changez l'extension de `image.jpg` en `image.png`.
4. **Lister les fichiers dans le répertoire :**
    - Listez tous les fichiers dans le répertoire `images` et affichez leurs noms.
5. **Vérifier si le fichier existe :**
    - Vérifiez si `nouveau_nom.txt` existe dans le répertoire courant et affichez un message approprié.

Partie 5 : Itération sur les fichiers

1. **Itérer sur tous les fichiers `.txt` dans un répertoire (vous pouvez créer un répertoire `documents` pour cet exercice) :**
    - Créez le répertoire `documents` et ajoutez-y quelques fichiers `.txt`.
    - Écrivez un code qui liste tous les fichiers `.txt` dans le répertoire `documents`.

Exercice sur le module `datetime`

Partie 1: Théorie

1. **Définitions**:
   - Que signifie "epoch" en informatique? Quel est l'epoch UNIX?
   - Qu'est-ce que UTC et quelle est sa différence par rapport aux fuseaux horaires?
   - Qu'est-ce qu'une date "naive" et une date "aware" en termes de fuseaux horaires?
2. **Questions**:
   - Quelles sont les classes principales dans le module `datetime` et à quoi servent-elles?
   - Quelle est la norme ISO 8601 et pourquoi est-elle importante pour le formatage des dates?

Partie 2: Pratique

1. **Création de dates**:
   - Créez une instance de `date` représentant le 1er janvier 2024.
   - Récupérez la date d'aujourd'hui et affichez-la.
2. **Manipulation des temps**:
   - Créez une instance de `time` représentant 14h30 et 45 secondes.
   - Récupérez l'heure actuelle et affichez les heures, minutes et secondes.
3. **Conversion de chaînes en dates**:
   - Convertissez la chaîne "2024-10-14" en un objet `date` à l'aide de la méthode appropriée.
   - Essayez de convertir la chaîne "31-12-2024" (au format DD-MM-YYYY). Affichez l'objet résultant.
4. **Gestion des fuseaux horaires**:
   - Créez un objet `datetime` pour l'heure actuelle à Paris.
   - Créez un objet `datetime` pour l'heure actuelle à Tokyo et affichez les deux.
5. **Calcul de deltas**:
   - Calculez la date qui sera 45 jours après aujourd'hui et affichez-la.
   - Créez une date représentant le 1er février 2024, puis ajoutez 2 mois à cette date.
6. **Différence entre deux dates**:
   - Créez deux objets `datetime` pour le 1er mars 2020 et le 10 mars 2020 à 15h00 dans le fuseau horaire de Montréal.
   - Calculez et affichez le nombre de jours entre ces deux dates, en tenant compte du fuseau horaire.

Partie 3: Questions de réflexion

1. En utilisant les concepts de `timedelta`, comment pourriez-vous implémenter un système qui envoie un rappel par e-mail 3 jours avant un événement ?

2. Pourquoi est-il important de travailler avec des objets `datetime` aware lors de la manipulation de dates et heures dans des applications internationales ?

Exercice : Traitement de Texte avec des Expressions Régulières

Vous êtes chargé de développer un petit script qui analyse un texte et effectue les tâches suivantes :

1. **Recherche d'un Mot :**
   - Trouvez la première occurrence du mot "chat" dans la chaîne donnée.
2. **Validation du Début de la Chaîne :**
   - Vérifiez si la chaîne commence par le mot "Python".
3. **Extraction de Nombres :**
   - Récupérez tous les nombres présents dans la chaîne suivante : `"Il y a 3 pommes et 2 oranges."`.
4. **Extraction de Mots en Majuscules :**
   - Trouvez tous les mots qui commencent par une majuscule dans la chaîne : `"Ceci Est Un Test."`.
5. **Remplacement de Nombres :**
   - Remplacez tous les nombres dans la chaîne suivante par le mot "beaucoup" : `"Il fait 25 degrés aujourd'hui."`.
6. **Remplacement de Voyelles :**
   - Remplacez toutes les voyelles dans la chaîne suivante par des 'X' : `"Bonjour le monde!"`.
7. **Validation de Mot de Passe :**
   - Vérifiez si le mot de passe suivant est valide. Un mot de passe est valide s'il contient au moins une majuscule, une minuscule, un chiffre, et a une longueur d'au moins 8 caractères. Testez le mot de passe : `"MonMotdepasse123"`.

Exercice : Gestion d'un système de réservation de jeux vidéo (fonctions, args, kwargs, tests)

Vous devez développer un système de réservation pour un magasin de jeux vidéo. Le système doit inclure les fonctionnalités suivantes :

1. **Ajouter un jeu :** Créez une fonction `ajouter_jeu(liste_de_jeux, *jeux, **options)` qui permet d'ajouter un ou plusieurs jeux à une liste. Les jeux doivent être ajoutés sous forme de chaînes de caractères. Les options doivent inclure des informations telles que `genre`, `plateforme`, et `prix`.

2. **Afficher les jeux :** Créez une fonction `afficher_jeux(liste_de_jeux)` qui affiche tous les jeux disponibles dans le magasin avec leurs détails.
3. **Rechercher un jeu :** Créez une fonction `rechercher_jeu(liste_de_jeux, nom_jeu)` qui retourne un message indiquant si un jeu est disponible ou non.
4. **Réserver un jeu :** Créez une fonction `reserver_jeu(liste_de_jeux, nom_jeu, nom_client)` qui permet de réserver un jeu pour un client. Si le jeu n'est pas disponible, un message doit être affiché.
5. **Tester les fonctionnalités :** Écrivez des tests pour vérifier le bon fonctionnement de vos fonctions. Assurez-vous de tester les cas de succès et d'erreur.