TD9 : Manipulation de dictionnaires : Terra Aventura

Préambule

— Dans votre dossier PythonOpenData créez un sous-dossier TD9.
— Lancer l'éditeur Visual Studio Code.
— Dans Visual Studio Code, ouvrir le dossier PythonOpenData/TD9 et créer un fichier TD9.py dans lequel vous écrirez votre code.

Problème posé

La région Nouvelle Aquitaine propose une chasse aux trésors touristique, nommée Terra Aventura : `https://www.terra-aventura.fr/`. Ainsi, sur tout le territoire de Nouvelle Aquitaine, environ 600 trésors ont été dissimulés. Chaque *trésor* (aussi appelé *cache* ou *aventure*) peut être trouvé en collectant des indices, lors de la découverte d'un lieu. Les aventures sont triées par thématique, et associées à un personnage. Par exemple, le personnage de Zabeth correspond à des aventures de type "Histoire".

Deux amis explorateurs, Marco et Polo, ont pour objectif de dénicher le maximum de trésors possibles. Terra Aventura propose plus de 600 trésors, mais Marco et Polo en ont déjà trouvé un certain nombre !

Deux fichiers sont à votre disposition sur Cursus :

— `tresors.json` est un export du site web Terra Aventura. Il contient la liste de tous les trésors disponibles dans la région Nouvelle Aquitaine.
— `trouvailles.csv` contient la liste des trésors déjà trouvés par Marco et Polo (identifiant du trésor et date de découverte).

Exercice 1 : Etiquetage des trésors déjà trouvés

Q.1.1. Écrire une fonction `charge_tresors`, qui ne prend pas de paramètre. Cette fonction doit ouvrir le fichier tresors.json et renvoyer la liste de trésors sous forme d'une liste de dictionnaires.
Indice : la liste renvoyée doit contenir 589 trésors.

Q.1.2. Écrire une fonction `charge_trouvailles`, qui ne prend pas de paramètre. Cette fonction doit ouvrir le fichier trouvailles.csv et renvoyer un dictionnaire qui associe à chaque identifiant de trésor trouvé, la date de trouvaille au format datetime.
Indice : le dictionnaire renvoyé doit contenir 51 clés, et commence par :

```
{'1367': datetime.datetime(2024, 10, 30, 0, 0),
 '1566': datetime.datetime(2024, 8, 14, 0, 0),
 '167': datetime.datetime(2023, 5, 2, 0, 0),
 ...
```

Q.1.3. Écrire une fonction `enrichit_dico` qui prend en entrée : une liste de trésors, telle que renvoyée par la question 1.1, et un dictionnaire de trouvailles, tel que renvoyé par la question 1.2. Cette fonction doit compléter les dictionnaires contenus dans la liste de trésors en ajoutant :

- un champ `"trouvé"`, qui vaut True si le trésor a été trouvé, False sinon.
- si le trésor a été trouvé, un champ `date` qui contient la date de la trouvaille au format datetime.

Voici les résultats attendus pour les deux premiers trésors :

```
 [{'difficulte': '3',
 'dpt': '16',
 'lat': '45.6808667',
 'lng': '0.4190667',
 'nid': '1980',
 'nom': 'À la pierre et au moulin...',
 'saison': '14',
 'terrain': '3',
 'trouvé': False,
 'type': 'Zectonic',
 'ville': 'Vilhonneur'},
{'date': datetime.datetime(2022, 7, 15, 0, 0),
 'difficulte': '2',
 'dpt': '16',
 'lat': '46.0151',
 'lng': '0.6730333',
 'nid': '72',
 'nom': "Remontez le temps, au fil de l'eau",
 'saison': '5',
 'terrain': '2',
 'trouvé': True,
 'type': 'Zabeth',
 'ville': 'Confolens'}
 ...]
```

Exercice 2 : Trésor de prédilection

Marco et Polo ont un type de trésor de prédilection... Lequel est-ce ?

Q.2.1. Ecrire une fonction `analyse_donnee` qui prend en entrée une liste de trésors, enrichie des trouvailles. Cette fonction doit construire et renvoyer un dictionnaire de dictionnaire, permettant de compter pour chaque type de trésor le nombre de trésors associés, ainsi que le nombre de trouvailles. Voici le format attendu :

```
{ 'Zabdo': {'nb': 10, 'trouvé': 0},
 ...
 'Zarthus': {'nb': 49, 'trouvé': 1},
 ...
 'Zenight': {'nb': 11, 'trouvé': 1},
 ...
}
```

Q.2.2. Écrire une fonction `type_preferé` qui prend en entrée un dictionnaire de trouvaille par type (tel que renvoyé par la question 2.1) et qui renvoie le nom du type préféré, c'est à dire celui pour lequel le nombre de "trouvé" est le plus grand.

Q.2.3. Écrire une fonction `type_plus_avancé` qui prend n entrée un dictionnaire de trouvaille par type (tel que renvoyé par la question 2.1) et qui renvoie le nom du type le plus avancé, c'est à dire celui dont le pourcentage de trésors trouvés est le plus grand.

Exercice 3 : Dates de découverte

Marco et Polo sont nostalgiques... Ils voudraient visualiser leurs trouvailles dans l'ordre des découvertes.

Q.3.1. Ecrire une fonction `affiche_trouvailles_ordonnees` qui prend en entrée une liste de trésors, enrichie des trouvailles. Cette fonction doit conserver uniquement les trésors trouvés, les ordonner selon la date de trouvaille, et les afficher au format suivant :

```
-------------------------------------------
Liste des trouvailles par ordre de découverte
-------------------------------------------
Le 01/05/2022 à Pau : Pau, c'est royal !
Le 13/07/2022 à Chef-Boutonne : Le vent tourne !
Le 13/07/2022 à Saint-Maixent-l'Ecole : Les Poï'z prennent du galon
Le 14/07/2022 à Chalais : Chalais, inimitable !
```

Q.3.2. Marco et Polo ne peuvent pas aller régulièrement trouver des trésors. Certains mois de l'année sont plus propices aux découvertes que d'autres. Écrire une fonction `analyse_mois_trouvaille` qui prend un dictionnaire de trouvailles, tel que renvoyé par la question 1.2. Cette fonction doit renvoyer un dictionnaire associant à chaque mois de l'année le nombre de trouvailles.

On doit obtenir : `{1: 0, 2: 0, 3: 0, 4: 8, 5: 7, 6: 0, 7: 22, 8: 8, 9: 0, 10: 5, 11: 0, 12: 1}`

Exercice 4 : Pour aller plus loin : cartographie

En utilisant le module de cartographie vue dans un TD précédent, on souhaite cartographier les trésors.

Q.4.1. Ecrire une fonction `cartographie_tresors` qui prend en entrée une liste de trésors, enrichie des trouvailles. Cette fonction doit créer puis cartographier les trésors. Les trésors trouvés seront affichés dans une couleur différente des trésors non trouvés. On pourra afficher le nom du trésor au survol.