TD1 : exploitation de données au format JSON

Télécharger l'archive *td1.zip* depuis l'espace-cours Cursus et décompresser cette archive dans un répertoire personnel. Le fichier de données au format JSON s'appelle *travaux.json*, et le fichier de code Python à modifier est *td1.py* .Pour une introduction au format JSON, consulter la page Wikipédia https://fr.wikipedia.org/wiki/JavaScript_Object_Notation.

Le fichier de données *travaux.json* contient des descriptions de travaux en cours sur la commune de Rennes et ses alentours, au format GeoJSON[1], avec pour chaque chantier, des données géométriques (liste de coordonnées géographiques notamment) et des caractéristiques du chantier (quartiers, localisation, date de début et de fin, etc.). Les données proviennent de l'API « Travaux » de Rennes Métropole[2].

Exercice 1 – Visualisation et importation des données JSON

Pour diminuer au maximum la taille des fichiers au format JSON (souvent destinés à être transférés par Internet), les espaces, retours à la ligne, et autres marques d'espacement, sont souvent retirés des données brutes. Cela rend ce type de fichiers difficilement lisibles tels quels car l'arborescence des données n'apparaît pas de prime abord (vous pouvez par exemple voir ce que ça donne en ouvrant le fichier *travaux.json* dans un éditeur de texte).

Il existe cependant des logiciels ou plugins qui permettent de faire réapparaître cette arborescence. C'est le cas du navigateur internet Firefox.

1. Ouvrir le fichier *travaux.json* dans Firefox (vous pouvez glisser/déposer le fichier dans un nouvel onglet depuis l'explorateur de fichiers).
2. Observer la structure des données en repérant notamment où se trouvent les caractéristiques des chantiers référencés.
3. À l'aide du module `json` de Python, récupérer dans le corps principal de votre programme un dictionnaire contenant l'intégralité des données du fichier *travaux.json* et afficher ce dictionnaire. La documentation complète du module `json` est disponible en ligne à l'adresse : https://docs.python.org/fr/3/library/json.html
   Indication : pour l'ouverture du fichier on utilisera la fonction `open()` à laquelle on ajoutera un paramètre `encoding='utf-8'` afin que les caractères accentués soient correctement gérés.

Exercice 2 – Création d'une classe *Chantier* et d'une liste de chantiers

1. Créer une classe (et son initialisateur) `Chantier` ayant les attributs suivants :

- attribut de classe : un dictionnaire `quartiers`. Ce dictionnaire devra être mis à jour à chaque fois qu'un objet `Chantier` dont le quartier n'a pas encore été rencontré, est créé. Il contiendra comme clés, les identifiants des quartiers (de type `int`) apparaissant dans les données avant le nom des quartiers, et comme valeurs les noms du ou des secteurs du quartier séparés par des « / » sans espace.

1 http://geojson.org/
2 http://travaux.data.rennesmetropole.fr/. Les données JSON sont directement accessibles à l'URL http://travaux.data.rennesmetropole.fr/api/roadworks

  - Exemple :
    `{1:'Centre', 2:'Thabor/Saint Hélier/Alphonse Guérin', … }`
- attributs d'instances : les valeurs des attributs d'instance d'un objet `Chantier` seront initialisées par l'initialisateur qui recevra en paramètre le dictionnaire correspondant à la partie `'properties'` d'un chantier dans les données JSON. Les correspondances sont les suivantes :

attribut d'instance `Chantier`	clé JSON (dans `'properties'`)
`id` (type `string`)	`'id'`
`quartierID` (type `int`)	début de l'attribut `'quartier'`
`localisation` (type `string`)	`'localisation'`
`type` (type `string`)	`'type'`
`libelle` (type `string`)	`'libelle'`
`perturbation` (type `string`)	`'niv_perturbation'`

2. Dans la classe `Chantier`, surcharger la méthode spéciale `__repr( )__` pour que l'affichage d'un objet `Chantier` donne le texte suivant : `nom(s) du quartier, localisation : libellé (type, niv_perturbation)`
   - Ex :
     `Centre, Rue du Puits Mauger : Métro ligne b - Construction de la station Colombier (Route barrée, Secteur à éviter)`
3. Dans la partie « Définition locale des fonctions », créer une fonction `initListeChantiers(data)` qui prend en argument le dictionnaire des données JSON et renvoie une liste d'objets `Chantier` initialisés à partir des données du dictionnaire. Pour tester, afficher cette liste dans le programme principal.

Exercice 3 – Gestion des dates

Les dates de début et de fin des chantiers seront représentées par des objets de type `datetime` du module Python `datetime` (déjà importé dans le fichier *td1.py*) qui permettent de manipuler des données représentant une date et une heure. Les objets `datetime` peuvent être comparés avec les opérateurs classiques de comparaison.

La création d'un objet `datetime` se fait par l'appel de l'initialisateur `datetime()` avec en arguments, l'année, le mois, le jour, l'heure, les minutes et les secondes, tous de type `int` (les arguments heure, minutes et secondes sont facultatifs) :

```
monDateTime = datetime(year, month, day, hour, minute, second)
```

On peut obtenir une chaîne de caractères représentant un objet `datetime` en appelant la méthode `strftime()` sur cet objet, à laquelle on passe une chaîne de caractères représentant le modèle de formatage souhaité.

Exemple d'utilisation :

```
monDateTime.strftime("%Y-%m-%d %H:%M:%S")
```

Les indicateurs de format utilisés ici sont :`%Y` pour l'année sur 4 chiffres, `%m` pour le numéro du mois, `%d` pour le numéro du jour, `%H` pour l'heure sur 24h, `%M` pour les minutes, et `%S` pour les secondes.

Inversement, il est possible de créer un objet datetime à partir d'une chaîne de caractères représentant une date et une heure grâce à la méthode de classe `datetime.strptime()` à laquelle on spécifie le formatage utilisé pour représenter la date.

Exemple d'utilisation :

```
datetime.strptime(chaine, "%Y-%m-%d %H:%M:%S")
```

Le module fournit également des objets de type `timedelta` qui représentent des durées (différences) entre deux `datetime`. Cette durée est représentée par les attributs d'instance `days`, `seconds` et `microseconds`.

La documentation complète du module `datetime` est disponible à l'adresse :

https://docs.python.org/fr/3/library/datetime.html

1. Dans l'initialisateur de la classe `Chantier`, rajouter la création de deux attributs d'instance, `debut` et `fin` de type `datetime`, correspondant aux dates de début et de fin de chantier (les chaînes de caractères donnant ces dates sont déjà présentes dans le dictionnaire de données passé en paramètre de l'initialisateur de `Chantier`).
2. Modifier le code de la méthode `__repr()__` pour rajouter l'affichage des dates de début et de fin de chantier. Les dates devront être affichées au format `jour/mois/année heure:minute:seconde`

   Exemple d'affichage attendu :

   ```
   Le Blosne, Boulevard de Bulgarie, du 11/01/2017 00:00:00 au 22/01/2017 00:00:00 : Travaux de renouvellement de la conduite et branchements (Chaussée rétrécie, Impact limité)
   ```

3. Ajouter à la classe `Chantier` trois méthodes d'instance, `enCours()`, `termine()` et `aVenir()` prenant en argument une chaîne de caractères représentant une date et une heure (au format "`année-mois-jour heure:minute:seconde`"). Ces méthodes renverront un booléen permettant de tester l'état du chantier à cette date : en cours, terminé ou à venir, respectivement.

Exercice 4 – Utilisation de la classe Chantier et écriture de données JSON

1. Créer une fonction `listeChantiersEnCours()` qui prend en paramètre la liste de tous les chantiers et une chaîne de caractères (optionnelle) représentant une date et une heure au format "`année-mois-jour heure:minute:seconde`", et qui renvoie une liste des chantiers en cours à cette date. Si la date n'est pas fournie, c'est la date courante qui sera utilisée (l'objet `datetime` correspondant à la date courante peut être obtenu par un appel à la fonction `datetime.now()`).
2. Créer une fonction `affichePlanningChantiers()` qui prend en paramètre la liste des chantiers, un identifiant de quartier (type `int`) et une chaîne de caractères (optionnelle) représentant une date et une heure au format "`année-mois-jour heure:minute:seconde`". Si la date n'est pas fournie, c'est la date courante qui sera utilisée.

   La fonction devra afficher, en plus du nom du quartier, la liste des localisations et types des chantiers du secteur avec leur état et le délai avant le début ou la fin du chantier.

   Exemples d'affichage :

   ```
   Planning des chantiers pour le quartier Maurepas/Bellangerais

   Rue Jack Kerouac (Route barrée) : chantier en cours (fin dans 4 jours et 1 heures)

   Rue Guy Ropartz (Route barrée) : chantier à venir (début dans 0 jours et 1 heures)

   Rue Claude Bernard au n° 5 (Chaussée rétrécie) : chantier terminé depuis 361 jours et 1 heures

   Boulevard E. Mounier + Rue de la Marbaudais (Chaussée rétrécie) : chantier en cours (fin dans 1259 jours et 1 heures)
   ```

3. Créer dans la classe Chantier une méthode d'instance `jsonDictionnary()` retournant un dictionnaire représentant les attributs de l'instance au format JSON, avec les correspondances

suivantes :

clé du dictionnaire	valeur
`'id'`	valeur de l'attribut d'instance correspondant
`'quartier'`	nom du quartier
`'localisation'`	valeur de l'attribut d'instance correspondant
`'libelle'`	valeur de l'attribut d'instance correspondant
`'type'`	valeur de l'attribut d'instance correspondant
`'perturbation'`	valeur de l'attribut d'instance correspondant
`'debut'`	Date de début du chantier au format `jour/mois/année heure:minute:seconde`
`'fin'`	Date de fin du chantier au format `jour/mois/année heure:minute:seconde`

4. Créer une fonction `dumpListeChantiersJSON()` qui permet d'écrire dans un fichier les données au format JSON représentant une liste d'objets de type `Chantier`. Le nom du fichier de sortie sera passé en premier paramètre et la liste de chantiers en second paramètre.

   Indication : pour que l'affichage des caractères accentués se passe correctement, on ajoutera un paramètre `encoding='utf-8'` à la fonction `open()` lors de l'ouverture du fichier de sortie, et un paramètre `ensure_ascii=False` à la fonction `json.dump()` qui permet l'écriture des données au format JSON dans le fichier.

   Exemple d'affichage du fichier obtenu :

```
{
    "chantiers":
[
{
    "id": 44675,
    "type": "Chaussée rétrécie",
    "debut": "24/11/2015 00:00:00",
    "quartier": "Centre",
    "localisation": "Rue de Saint-Malo",
    "perturbation": "Impact limité",
    "libelle": "Travaux dans le cadre du chantier de construction du Centre des Congrès",
    "fin": "26/02/2017 00:00:00"
},
{
    "id": 47145,
    "type": "Route barrée",
    "debut": "15/01/2017 00:00:00",
    "quartier": "Jeanne d'Arc/Longs Champs/Beaulieu",
    "localisation": "Square Francis Pellerin",
    "perturbation": "Circulation difficile",
    "libelle": "Travaux de pompage de chape le 15 et le 18 janvier",
    "fin": "18/01/2017 00:00:00"
},
{
…
…
]
}
```