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* .

Préambule : le format JSON

JSON (*JavaScript Object Notation*) est un format de données structurées (i.e. avec imbrication d'éléments les uns dans les autres) qui permet de représenter sous forme textuelle les structures de données du langage JavaScript :

- les *Object* de JavaScript (qui sont similaires aux **dictionnaires** Python) entre accolades `{…}` ;
- les *Array* de JavaScript (qui sont similaires aux **listes** Python) entre crochets droits `[…]`.

Dans un fichier JSON, ces structures de données sont souvent imbriquées les unes dans les autres. On trouve ainsi fréquemment des listes de dictionnaires, des dictionnaires de dictionnaires, des dictionnaires de listes, etc. , avec possiblement plusieurs niveaux d'imbrication.
En JSON, les clés des dictionnaires sont obligatoirement des chaînes de caractères (i.e. écrites entre guillemets), les valeurs peuvent être des listes, des dictionnaires ou des données de type simple (nombres entiers ou décimaux, chaînes de caractères, booléens (`true` ou `false`) ou valeur `null` (absence de valeur équivalente au `None` de Python).

Pour une introduction plus détaillée au format JSON, consulter la page Wikipédia https://fr.wikipedia.org/wiki/JavaScript_Object_Notation.

Exercice 1 – Visualisation et importation des données JSON

Le fichier de données *travaux.json* contient des descriptions de travaux en cours sur la commune de Rennes et ses alentours, 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 du jeu de données « *travaux_30_jours* » de Rennes Métropole[1].

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 tel quel 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 extension (plugins) qui permettent de faire réapparaître cette arborescence. C'est le cas du navigateur internet Firefox (ou Chrome avec l'extension JSON Viewer).

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 une liste de dictionnaires 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*

1 https://data.rennesmetropole.fr/explore/dataset/travaux_30_jours/information/

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

Informations sur les noms des quartiers

Dans un bloc de données JSON décrivant un chantier, la clé `'quartier'` indique l'identifiant et le nom du ou des quartiers impacté(s) par le chantier. Le nom d'un quartier est parfois formé de plusieurs noms de secteurs séparés par la chaîne `' / '` (un slash entouré de 2 espaces), par exemple `"Bourg l'Evesque / La Touche / Moulin du Comte"`. On gardera ces noms composés tels quels.

Dans le cas où plusieurs quartiers sont impactés par le chantier, les informations des différents quartiers impactés sont séparées par la chaîne `', '` (une virgule suivie d'un espace).

Exemple où deux quartiers sont impactés :

```
"quartier" : "8 - Sud gare, 35281 - Saint-Jacques-de-la-Lande"
```

Pour un quartier donné, l'identifiant est la partie qui précède le nom du quartier (dans l'exemple ci-dessus : `"8"` pour le premier quartier, `"35281"` pour le deuxième quartier) séparée du nom par la chaîne `' - '` (un tiret entouré de 2 espaces). Attention, la chaîne `' - '` peut parfois se trouver aussi dans le nom (composé) du quartier comme dans l'exemple suivant : `"9 - Cleunay / Arsenal - Redon / La Courrouze"`.

Enfin, dans quelques cas rares où le chantier est entièrement en dehors de Rennes, l'information de quartier n'est pas présente (i.e. la valeur associée à la clé `"quartier"` est `null`). Dans ce cas particulier on utilisera les informations de la clé `"commune"` qui fonctionnent exactement comme celles des quartiers (cf règles énoncées ci-dessus), l'identifiant étant dans ce cas le code postal de la commune.

Questions :

1. Créer une classe `Chantier` et son initialisateur qui prendra en paramètre un dictionnaire représentant les données JSON d'**un** des chantiers. La classe `Chantier` aura les attributs suivants :
   - *Attribut de classe* : un dictionnaire `quartiers`. **Attention : ce dictionnaire devra être mis à jour à chaque fois qu'un objet `Chantier` dont un des quartiers n'a pas encore été rencontré, est créé.** Il contiendra comme clés, les identifiants des quartiers (de type `str`) et comme valeurs les noms des quartiers. Exemple :

```
{'5': 'Maurepas / Bellangerais', '35281': 'Saint-Jacques-de-la-Lande', … }
```

   - *Attributs d'instances* : les valeurs des attributs d'instance d'un objet `Chantier` seront initialisées grâce aux données contenues dans le dictionnaire passé à l'initialisateur. Les correspondances sont les suivantes :

attribut d'instance `Chantier`	clé JSON dans le bloc de données d'un chantier
`id` (type `string`)	`'id'`
`quartier_ids` (type `list` de `str`)	Dans la clé `'quartier'` (ou `'commune'` si pas d'information de quartier), parties qui précèdent les noms de quartier
`localisation` (type `string`)	`'localisation'`
`type` (type `string`)	`'type'`
`libelle` (type `string`)	`"travaux"` suivi d'un espace et de la clé `'libelle'` si cette clé est non nulle, sinon juste `"travaux"`
`perturbation` (type `string`)	`'niv_perturbation'`

2. Dans la classe `Chantier`, définir une méthode spéciale pour que l'affichage d'un objet `Chantier` avec la fonction `print()` donne le texte suivant :

```
nom du quartier ou des quartiers, localisation : libellé (type, perturbation)
```

Lorsque plusieurs quartiers sont impactés, les noms des quartiers seront concaténés avec la chaîne ' + ' (signe plus entouré de deux espaces).
Exemple d'affichage pour un chantier impactant les quartiers Sud gare et Le Blosne :

```
Sud gare + Le Blosne, 28 Rue Raymond Hermer : travaux sur réseaux ou ouvrages d'eau potable (Circulation interdite,Impact limité)
```

Pour tester, repérer visuellement dans les données JSON (i.e. dans Firefox) un chantier impactant plusieurs quartier (ou communes) et utiliser le bloc de données correspondante pour créer un objet Chantier puis afficher cet objet.

3. Dans la partie « Définition locale des fonctions », créer une fonction `init_liste_chantiers(data)` qui prend en argument la liste de dictionnaires **complète** des données JSON lues depuis le fichier *travaux.json* et renvoie une liste Python d'objets `Chantier` initialisés à partir de ces données. 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*). La documentation complète du module `datetime` est disponible à l'adresse : https://docs.python.org/fr/3/library/datetime.html.

Questions :

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. Dans le dictionnaire de données passé en paramètre de l'initialisateur de `Chantier` les chaînes de caractères donnant ces dates sont au format ISO 8601 avec une information de fuseau horaire. On utilisera la fonction appropriée du module `datetime` pour récupérer des objets `datetime` à partir de ce format.

   Remarque : en Python on ne peut pas comparer un objet `datetime` contenant une information de fuseau horaire (objet `datetime` "offset-aware") avec un autre objet `datetime` ne contenant pas d'information de fuseau horaire (objet `datetime` "offset-naive"). Pour simplifier la gestion des dates on retirera l'information de fuseau horaire des objets `datetime` récupéré ici en utilisant la méthode `.replace(tzinfo=None)` sur ces objets. Si on suppose que toutes les dates sont données pour le même fuseau horaire, cette information n'a plus d'utilité.

2. Modifier le code de la méthode spéciale d'affichage 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 pour le premier chantier des données :

   ```
   Bourg l'Evesque / La Touche / Moulin du Comte, du 86 au 88 Rue de Lorient du 09/05/2023 00:00:00 au 15/05/2025 00:00:00 : travaux de construction de bâtiment (Interdiction de stationnement,Impact nul)
   ```

3. Ajouter à la classe `Chantier` trois méthodes d'instance, `en_cours()`, `termine()` et `a_venir()` prenant en argument une chaîne de caractères représentant une date et une heure (au format `"jour/mois/année 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.

   Tester sur le premier chantier avec des dates pertinentes permettant de vérifier les différentes situations.

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

1. Créer une fonction `liste_chantiers_en_cours()` qui prend en paramètre la liste de tous les chantiers et une chaîne de caractères (optionnelle) représentant une date et une heure au format "`jour/mois/année 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()`).

   Tester en affichant (un à un) les chantiers encore en cours le 1er janvier de la prochaine année à minuit (`00:00:00`).

2. Créer une fonction `affiche_planning_chantiers()` qui prend en paramètre la liste des chantiers, un identifiant de quartier (type `str`) et une chaîne de caractères (optionnelle) représentant une date et une heure au format "`jour/mois/année 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 pour le quartier d'identifiant `'1'` :

   ```
   Planning des chantiers du quartier Centre au 06/01/2026 12:38:51 :

   du 47 au 50bis Boulevard de Chézy, dans la contre-allée (Circulation à double sens) : chantier en cours (fin dans 205 jours et 13 heures)

   du 47 au 50bis Boulevard de Chézy, dans la contre-allée (Interdiction de stationnement) : chantier en cours (fin dans 205 jours et 13 heures)

   Rue des Portes Mordelaises, entre la rue de Juillet et le porche (Circulation interdite) : chantier à venir (début dans 6 jours et 12 heures)

   ...
   ```

3. Créer dans la classe Chantier une méthode d'instance `json_dictionnary()` 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(s) du ou des quartier(s) impacté(s) (séparés par un `'+'` s'il y en a plusieurs)
`'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 `dump_liste_chantiers_JSON()` 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. On cherchera à obtenir un affichage « structuré » des données JSON dans le fichier de sortie en utilisant le paramètre d'indentation de la fonction `json.dump()`.

Indication *: pour que le codage/décodage 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.*

Extrait du contenu du fichier à obtenir :


```
[
    {
        "id": 18,
        "quartier": "Maurepas / Bellangerais",
        "localisation": "Avenue de Rochester à gauche de l'entrée du parking face au 2 square
Armand de la Rouerie",
        "libelle": "travaux",
        "type": "Interdiction de stationnement",
        "perturbation": "Impact nul",
        "debut": "01/05/2025 02:00:00",
        "fin": "03/04/2026 02:00:00"
    },
    {
        "id": 86,
        "quartier": "La Pommeraie",
        "localisation": "face au 48 Avenue Monseigneur Mouézy",
        "libelle": "travaux sur réseaux ou ouvrages de télécommunications",
        "type": "Mesure libre circulation",
        "perturbation": "Impact nul",
        "debut": "15/12/2025 01:00:00",
        "fin": "15/01/2026 01:00:00"
    },
    ...
]
```