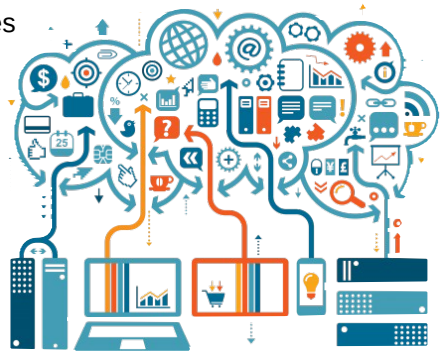
OpenData

Thématique : Langages et programmation **Niveau :** Term NSI **Type d'activité :** TP/Projet **Point(s) du programme traité(s) :** Utiliser des API (Application Programming Interface). **Résumé :** Cette activité peut être utilisée avant d'aborder les bases de données. Elle permet de revoir le traitement des données en tables vu en 1ère et de découvrir le format JSON et ainsi que l'utilisation d'OpenData. L'activité ce prolonge sur un mini-projet permettant de mettre en œuvre de nombreuses notions (IHM, API, données en tables, POO). **Prérequis :** traitement des données en table, POO et module tkinter	**Durée : 6**h **Matériel / logiciels par binôme :** • PC équipé de l'IDE Python de votre choix • Connexion internet • module python Folium **Documents ressources :** • corrections des travaux préliminaires • tuto_python_matplotlib.pdf • Conseils pour réussir votre présentation.pdf • Exemple_de_présentation_projet.pptx

1) Les données dans notre société

Les données sont devenues un enjeu pour notre société. Elles touchent tous les domaines : la santé, l'éducation, l'industrie, la sécurité, le commerce,… De nouveaux termes sont apparus : ***Big Data***, ***Open Data***, … de nouveaux métiers sont créés : *Architecte Big Data*, *Data Scientist*, … de nouvelles disciplines sont enseignées : *global data analytics*, … de nouvelles technologies sont développées : le ***Cloud Computing***, les bases de données ***NoSQL***, … et de nouveaux algorithmes sont appliqués : ***MapReduce***, ***Spark***, …
L'utilisation et la maîtrise du big data suscite beaucoup d'enthousiasme, mais également des inquiétudes, en particulier sur la protection des données à caractère personnel.



Travail n°1 :
A partir des liens ci-dessous, répondre aux questions suivantes :
https://www.lebigdata.fr/definition-big-data
https://www.lebigdata.fr/top-metiers-du-big-data-cloud
https://www.lebigdata.fr/open-data-definition

1) Etablir une définition du big data.
2) Définir la règle des 3v qui définit les caractéristiques des outils big data.
3) Expliquer le métier d'ingénieur big data.
4) Lister les 3 critères fondateurs de l'opendata

2) Mise en forme des données

Les données sont principalement représentées sous la forme de tableaux. On parle de données tabulaires.
Exemple :

Nom	Prenom	Age
Perrin	Léo	18
Petit	Loïc	32
Leroux	Pierre	27

Il existe trois formats pour représenter un tableau de données : les formats CSV, XML et JSON (le XML est de moins en moins utilisé)
Ces trois formats sont des fichiers composés d'une suite de caractères où l'on distingue deux types d'information :

- Les données.
- Les caractères permettant de structurer ces données.

2.1 Le format CSV

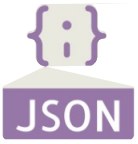
Le format *Comma Separated Values (CSV)* structure les données sous la forme de valeurs séparées par des virgules. Ce format est très facile à générer et à manipuler.
Chaque ligne du fichier CSV correspond à une ligne du tableau et chaque valeur séparée par une virgule correspond à une colonne du tableau.

Exemple du tableau précédent au format CSV :

```
1  Nom,Prenom,Age
2  Perrin,Léo,18
3  Petit,Loic,32
4  Leroux,Pierre,27
```

La première ligne du fichier contient l'entête de la table, à savoir le nom de chacune des colonnes. Les lignes suivantes contiennent les données du tableau, en respectant l'ordre des colonnes. Le séparateur n'est pas forcément une virgule, on peut par exemple utiliser le point-virgule.

2.2 Le format JSON



Le format *JavaScript Object Notation* (JSON) est un format utilisé pour représenter des objets qui dérive de la notation des objets du langage JavaScript.
Un fichier JSON est composé d'objets, de tableaux et de valeurs.

- L'objet : Il contient un membre ou une liste de membres. La syntaxe de l'objet est:
  { membre, membre, ... }
  Chaque membre étant de la forme d'une paire clef-valeur (key-value) : "clef" : "valeur"
- Le tableau : il contient une ou plusieurs valeurs séparées par des virgules.
  [valeur, valeur, ...]
- Les valeurs : une valeur peut être une chaîne de caractères, un nombre, un booléen… mais elle peut être aussi un objet, un tableau, voir même un tableau d'objets.

```
{
  "liste": {
    "personne": [
      {
        "NOM": "Perrin",
        "PRENOM": "Léo",
        "AGE": 18
      },
      {
        "NOM": "Petit",
        "PRENOM": "Loic",
        "AGE": 32
      },
      {
        "NOM": "Leroux",
        "PRENOM": "Pierre",
        "AGE": 27
      }
    ]
  }
}
```

Exemple du tableau au format JSON :
Dans cet exemple, nous avons un fichier JSON composé d'un objet principal constitué d'un membre dont la clef est "liste" et la valeur est un objet {}. Cet objet est constitué d'un membre dont la clef est "personne" et la valeur est un tableau []. Ce tableau est composé lui-même de 3 objets [{…},{…},{…}]. Chacun de ces objets contient 3 membres.

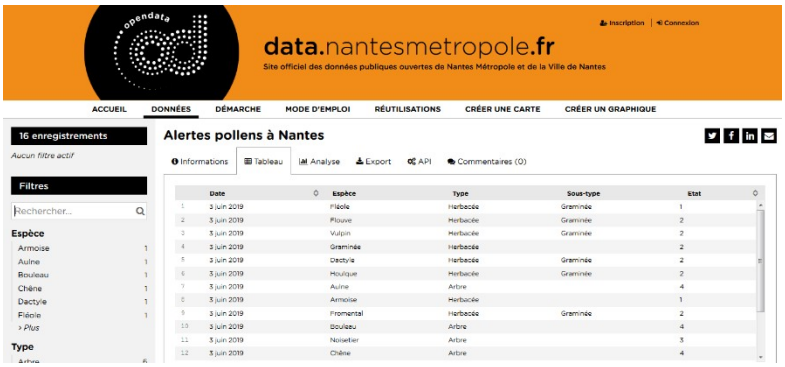
Remarquez que les nombres ne sont pas placés entre guillemets.

Travail n°2 :
A partir du tableau suivant, créer manuellement deux fichiers : l'un au format CSV et l'autre au format JSON.

Nom	Prix	code
Banane	5,99	77
Pomme	2,99	99
Poire	7,99	170

3) Traitement des données avec Python

Python possède de nombreuses instructions dédiées au traitement des données. L'objectif de l'activité est de réaliser un programme pour chaque format de données : CSV et JSON.
Les données, que nous allons utiliser, sont accessibles sur l'open data de Nantes à l'adresse suivante : lien



Ces données représentent le niveau de pollens à Nantes en fonction des essences d'arbres ou de plantes. Ces données sont actualisées tous les jours.

Modèle de données :

Modèle de données
Cliquez pour replier

Date

Date

Nom (identifiant) : date
Type : date
Exemple : 2019-06-03

Espèce

Nom de l'espèce allergisante

Nom (identifiant) : nom
Type : texte
Exemple : Fléole

Type

Type de l'espèce

Nom (identifiant) : type
Type : texte
Exemple : Herbacée

Sous-type

Sous-type de l'espèce

Nom (identifiant) : sous_type
Type : texte
Exemple : Graminée

Etat

Etat de l'alerte (1 pour "pas d'émission", 2 pour 'émission en cours", 3 pour "plus d'émission" et 4 pour "non observable")

Nom (identifiant) : etat
Type : décimal
Exemple : 1

Travail n°3 :
Récupérer les fichiers alertes-pollens-nantes.csv et alertes-pollens-nantes.json mis à votre disposition par votre enseignant (ces fichiers correspondent aux données fournies au mois de juin, en pleine période de pollinisation).

3.1 Traitement d'un fichier CSV

Pour traiter les données au format CSV, python a à sa disposition une bibliothèque dédiée :
https://docs.python.org/fr/3/library/csv.html
Le module csv implémente la méthode reader pour lire des données tabulaires au format CSV. Chaque ligne lue depuis le fichier CSV est alors renvoyée comme une liste de chaînes de caractères. Vous pouvez aussi lire les données dans un dictionnaire en utilisant les méthodes DictReader.

Exemple :

```
import csv

f=open("alertes-pollens-nantes.csv", 'rt', encoding='UTF8')
lecteurCSV=csv.DictReader(f,delimiter=" ;")
for ligne in lecteurCSV:
    print(ligne)
f.close()

print(ligne.items())
print(ligne.keys())
print(ligne.values())
print(ligne['Type'])
if ligne['Etat']=='3.0':
    print(ligne)
```

- La méthode ***DictReader()*** permet de lire les données du fichier et les stocke dans un dictionnaire dont les clés (keys) correspondent aux entêtes du fichier. Dans le cas du fichier "Alertes_pollens_nantes.csv", cela correspond aux noms des différentes colonnes (les attributs : Date, Nom, Type, ….).
- Les méthodes ***items(), keys()*** et **values()** sont des méthodes propres à l'utilisation des dictionnaires. Ces trois méthodes permettent de parcourir soit l'ensemble des paires clés-valeurs `items()`, soit l'ensemble des clés `keys()`, soit l'ensemble des valeurs `values()`.

Résultat du script :

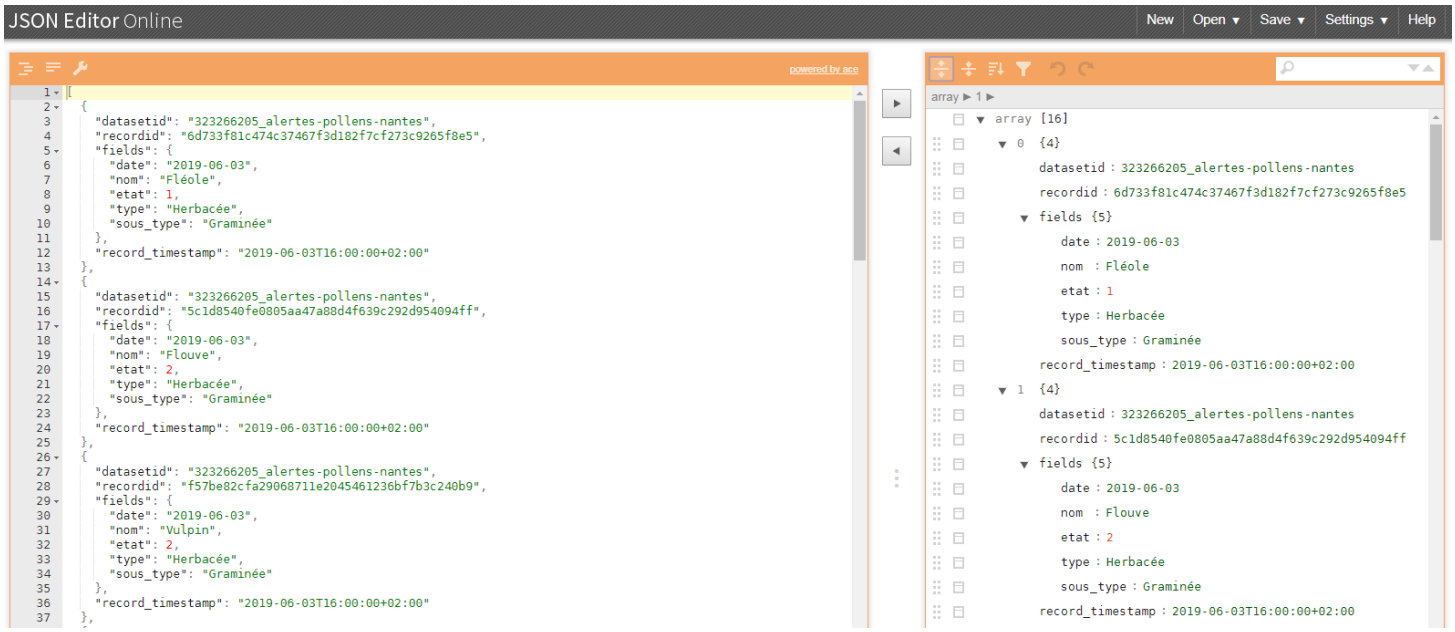
```
>>>
{'Date': '2019-06-03', 'Type': 'Herbacée', 'Espèce': 'Fléole', 'Etat': '1.0', 'Sous-type': 'Graminée'}
{'Date': '2019-06-03', 'Type': 'Herbacée', 'Espèce': 'Flouve', 'Etat': '2.0', 'Sous-type': 'Graminée'}
{'Date': '2019-06-03', 'Type': 'Herbacée', 'Espèce': 'Vulpin', 'Etat': '2.0', 'Sous-type': 'Graminée'}
{'Date': '2019-06-03', 'Type': 'Herbacée', 'Espèce': 'Graminée', 'Etat': '2.0', 'Sous-type': ''}
{'Date': '2019-06-03', 'Type': 'Herbacée', 'Espèce': 'Dactyle', 'Etat': '2.0', 'Sous-type': 'Graminée'}
{'Date': '2019-06-03', 'Type': 'Herbacée', 'Espèce': 'Houlque', 'Etat': '2.0', 'Sous-type': 'Graminée'}
{'Date': '2019-06-03', 'Type': 'Arbre', 'Espèce': 'Aulne', 'Etat': '4.0', 'Sous-type': ''}
{'Date': '2019-06-03', 'Type': 'Herbacée', 'Espèce': 'Armoise', 'Etat': '1.0', 'Sous-type': ''}
{'Date': '2019-06-03', 'Type': 'Herbacée', 'Espèce': 'Fromental', 'Etat': '2.0', 'Sous-type': 'Graminée'}
{'Date': '2019-06-03', 'Type': 'Arbre', 'Espèce': 'Bouleau', 'Etat': '4.0', 'Sous-type': ''}
{'Date': '2019-06-03', 'Type': 'Arbre', 'Espèce': 'Noisetier', 'Etat': '3.0', 'Sous-type': ''}
{'Date': '2019-06-03', 'Type': 'Arbre', 'Espèce': 'Chêne', 'Etat': '4.0', 'Sous-type': ''}
{'Date': '2019-06-03', 'Type': 'Herbacée', 'Espèce': 'Plantain', 'Etat': '2.0', 'Sous-type': ''}
{'Date': '2019-06-03', 'Type': '', 'Espèce': '', 'Etat': '2.0', 'Sous-type': ''}
{'Date': '2019-06-03', 'Type': 'Arbre', 'Espèce': 'Frêne', 'Etat': '4.0', 'Sous-type': ''}
{'Date': '2019-06-03', 'Type': 'Arbre', 'Espèce': 'Saule', 'Etat': '3.0', 'Sous-type': ''}
dict_items([('Date', '2019-06-03'), ('Type', 'Arbre'), ('Espèce', 'Saule'), ('Etat', '3.0'), ('Sous-type', '')])
dict_keys(['Date', 'Type', 'Espèce', 'Etat', 'Sous-type'])
dict_values(['2019-06-03', 'Arbre', 'Saule', '3.0', ''])
Arbre
{'Date': '2019-06-03', 'Type': 'Arbre', 'Espèce': 'Saule', 'Etat': '3.0', 'Sous-type': ''}
>>>
```

Travail n°4 :

Tester et analyser le code ci-dessus afin de comprendre l'architecture des données CSV et son traitement par Python.
Modifier ensuite le code afin d'afficher le nom de toutes les espèces allergisantes dont le niveau d'émission est de 2 (émission en cours)

3.2 Traitement d'un fichier JSON

Le format JSON est un standard du net. Le fichier JSON que nous souhaitons analyser est de la forme :



La structure de base du JSON est une paire clef-valeur (key-value) : `"nom": "Fléole"`
On distingue les valeurs atomiques et les valeurs complexes (construites)

- Valeurs atomiques :
  - chaînes de caractères (entourées par les classiques guillemets) `"type": "Herbacée"`
  - nombres (entiers, flottants) `"etat": 2`
  - valeurs booléennes (true ou false)
- Valeur complexes :
  - Tableau (ou array) : `[{…},{…},{…}]`
  - Objet (ou object) : `"fields": {"date":"2019-06-03", "nom":"Vulpin", … }`

Pour lire un fichier JSON, nous utilisons le module json de Python (voir documentation).

Exemple de code :

```
import json

f = open("alertes-pollens-nantes.json","rt", encoding='UTF8')
lecteurJSON = json.loads(f.read())
print(lecteurJSON)
print("*****************************************************")

for objet in lecteurJSON:
    print(objet)

print("*****************************************************")
print(objet.keys())
print(objet['fields'].keys())
print(objet['fields']['nom'])

print("*****************************************************")
if objet['fields']['etat']==3 :
    print(objet['fields']['nom'])
f.close()
```

Travail n°5 :
Tester et analyser le code ci-dessus afin de comprendre l'architecture des données JSON et son traitement par Python. Modifier ensuite le code afin d'afficher le nom de toutes les espèces allergisantes dont le niveau d'émission est de 2 (émission en cours)

3.3 Traitement des données en ligne

Il est aussi possible de récupérer en ligne les données au format JSON à partir d'une l'URL.
Nous allons travailler avec un nouveau jeu de données moins dépendant de la saison : l'indice ATMO prévisionnel de l'agglomération nantaise. Il s'agit de l'indice de qualité de l'air. C'est un indice prévisionnel calculé pour J-0, J+1 à J+2 et pour différentes zones géographiques.
https://data.nantesmetropole.fr/explore/dataset/323266205_indice-atmo-previsionnel-agglomeration-nantaise/download/?format=json&timezone=Europe/Berlin&lang=fr

Voici un extrait du modèle de données (10/15) :

Nom	type	Exemple	Signification
date_ech	date	2011-10-21	Date de l'indice
code_qual	int	2	Valeur de l'indice
lib_qual	texte	moyen	Qualificatif correspondant à l'indice
coul_qual	texte	#50ccaa	Couleur du qualificatif (hex)
date_dif	date	2021-10-19	Date de diffusion de l'indice
source	texte	Air Pays de la Loire	Source de la donnée
type_zone	texte	commune	Type d'entité administrative
code_zone	texte	44194	Code postal
lib_zone	texte	Sautron	Nom de la collectivité
…	…	…	…

Le code ci-dessous permet de récupérer les données depuis une url :

```
import requests
r = requests.get(' https://data.nantesmetropole.fr/explore/dataset/323266205_indice-atmo-previsionnel-agglomeration-nantaise/download/?format=json&timezone=Europe/Berlin&lang=fr')
print(reponse.text)
```

Les données récupérées sont encapsulées dans un tableau (array) d'objets (object). Chaque objet correspond à un enregistrement de données.
Remarque : Le module json de Python applique par défaut les conversions suivantes en décodant :

JSON	Python
objet	*dict*
array	*list*

Rappel : pour accéder à une valeur d'un élément d'un dictionnaire, nous utilisons l'instruction `dic['key']`, alors que pour accéder à la valeur d'un élément d'une liste nous utilisons l'instruction `list[0]`

Exemple de code :

```
import json
import requests

reponse = requests.get(' https://data.nantesmetropole.fr/explore/dataset/323266205_indice-atmo-previsionnel-agglomeration-nantaise/download/?format=json&timezone=Europe/Berlin&lang=fr')
print(reponse.text)

print("************************************")
tabJSON = json.loads(reponse.text)
i=0
for objetJSON in tabJSON:
        print("objet n°",i)
        print(objetJSON)
        i+=1

print("************************************")
print("Exemple de données issues du dernier objet du fichier json:")
print(objetJSON['fields']['date_ech'])
print(objetJSON['fields']['lib_qual'])
print(objetJSON['fields']['lib_zone'])
```

Le jeu de données de l'indice ATMO fournit aussi le niveau de différents polluants :

Polluants	Clé json	valeurs
dioxyde d'azote (NO2)	code_no2	Valeur du sous-indice, entier de 1 à 6, ou 0 si absent
dioxyde de soufre (SO2)	code_so2	Valeur du sous-indice, entier de 1 à 6, ou 0 si absent
ozone (O3)	code_o3	Valeur du sous-indice, entier de 1 à 6, ou 0 si absent
particules fines (PM10)	code_pm10	Valeur du sous-indice, entier de 1 à 6, ou 0 si absent
particules fines (PM25)	code_pm25	Valeur du sous-indice, entier de 1 à 6, ou 0 si absent

Travail n°6 :
Tester et analyser le code ci-dessus afin de comprendre l'architecture des données JSON récupérées par l'URL et son traitement par Python.
Modifier ensuite le code afin d'afficher le nom de toutes les communes dont le niveau de particules fines (PM10) est supérieur à 1.

3.4 Mise en forme des données

Notre objectif est de représenter par une courbe l'évolution du prénom masculin Sacha parmi les enfants nés à Saint-Nazaire.
Pour cela, nous allons utiliser le jeu de données fourni par l'opendata de Saint-Nazaire :
https://data.agglo-carene.fr/explore/dataset/214401846_prenoms_liste/download/?format=json&timezone=Europe/Berlin&lang=fr

Le modèle de données est le suivant :


```
{
    "datasetid": "214401846_prenoms_liste",
    "recordid": "d5c3edbac120fbcdc7532ecf902499ac7a7881df",
    "fields":
    {
        "enfant_prenom": "Henri",
        "enfant_sexe": "M",
        "nombre_occurences": 4.0,
        "coll_insee": "44184",
        "commune_nom": "Saint-Nazaire",
        "annee": "2008"
    },
    "record_timestamp": "2021-02-16T17:37:06.320+01:00"
}
```


Pour tracer des graphiques, nous allons utiliser le module matplotlib (voir mémo matplotlib)

Etape 1 : Extraction des données
Réaliser une fonction find_name(name,s,d) qui passe en paramètre le prénom recherché (name), le sexe (s) et le jeu de données json (d). Cette fonction retourne 2 listes : liste des années et liste des occurrences du prénom.

Etape 2 : Tri des données
Vous remarquez que les années ne sont pas fournies dans l'ordre, il faut donc remettre dans l'ordre les années et en parallèle remettre dans l'ordre les occurrences correspondantes.
Years=[2011, 2015, 2014, 2013, 2010, 2012, 2004, 2018, 2019, 2007, 2002, 2008, 2003, 2006, 2000, 2011, 2001, 2017, 2016, 2009, 2004, 2005, 2006, 2001, 2007, 2002, 2010, 2016, 2020]
Nb=[18, 16, 15, 15, 10, 19, 15, 6, 11, 8, 4, 7, 3, 1, 2, 1, 1, 14, 16, 8, 1, 10, 8, 5, 1, 3, 1, 1, 11]

Réaliser une fonction tri(liste1,liste2) qui passe en paramètre le liste des années et la liste des occurrences du prénom. Cette fonction retourne deux listes triées par odre chronologique.
Years=[[2000, 2001, 2001, 2002, 2002, 2003, 2004, 2004, 2005, 2006, 2006, 2007, 2007, 2008, 2009, 2010, 2010, 2011, 2011, 2012, 2013, 2014, 2015, 2016, 2016, 2017, 2018, 2019, 2020]
Nb= [2, 1, 5, 4, 3, 3, 15, 1, 10, 1, 8, 8, 1, 7, 8, 10, 1, 18, 1, 19, 15, 15, 16, 16, 1, 14, 6, 11, 11]

Etape 3 : Représentation des données
Réaliser une fonction graphe(x,y,color) qui passe en paramètre la liste des années (x), la liste des occurrences du prénom (y) et la couleur de la courbe sous forme de caractère (color). Voir mémo Matplotlib.

Travail n°7 :
En respectant les consignes ci-dessus (travailler étape par étape), réaliser un programme python pour représenter graphiquement l'évolution du prénom masculin Sacha parmi les enfants nés à Saint-Nazaire.

3.5 Cartographier des données

Notre objectif est de placer sur une carte l'ensemble des points de collecte des déchets multi matériaux près du lycée Aristide Briand de Saint-Nazaire.

Pour cela nous allons utiliser le module Folium de python. Voici un exemple d'utilisation :

```
import webbrowser
import folium
# Création d'une carte
fmap = folium.Map(location=[47.273471690927614, -2.232579926981806],
tiles='OpenStreetMap', zoom_start=8)
# Ajout d'un marqueur
folium.Marker([47.273471690927614, -2.232579926981806],
              popup='Lycée Aristide Briand',
              icon=folium.Icon(color='green')).add_to(fmap)
# Génération du fichier HTML contenant la carte
fmap.save('monLycee.html')
webbrowser.open('monLycee.html')
```

Recopiez le code et testez-le puis visualisez le résultat obtenu à l'aide du fichier html généré.

Pour notre travaille, nous allons utiliser le jeu de données suivant fourni par l'opendata de Saint-Nazaire :
https://data.agglo-carene.fr/explore/dataset/244400644_pav_dechets0/download/?format=json&timezone=Europe/Berlin&lang=fr

Le modèle de données est le suivant :


```
{
            "datasetid": "244400644_pav_dechets0",
            "recordid": "b977cec559ee976ef27bde9defb6919e921538e2",
            "fields": {
                  "reference": "29365",
                  "coll_siret": "244400644",
                  "coll_nom": "CARENE",
                  "geo_point_2d": [47.273589463, -2.23405651845],
                  "domaine": "Privé",
                  "flux": "Multi matériaux",
                  "type_col": "aérien",
                  "code_insee": "44184",
                  "geo_shape": {"type": "Point", "coordinates": [-2.234056518453134, 47.27358946299399]},
                  "collecte_par": "CARENE"},
                  "geometry": {"type": "Point","coordinates": [-2.23405651845, 47.273589463]},
                  "record_timestamp": "2021-01-07T11:25:06.385+01:00"
}
```

Etape 1 : Extraction des données

Réaliser une fonction find(flux,code,d) qui passe en paramètre le type de flux rechercher « multi matériaux » (flux), la localisation souhaitée en utilisant le code_insee 44184 (code) et les données json (d). Cette fonction retourne une liste de listes contenant les coordonnées géographiques des points de collecte.

Etape 2 : Cartographie

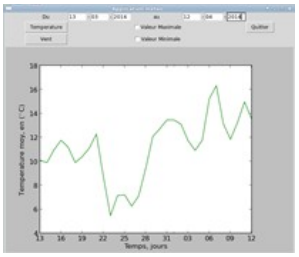
Générer une carte représentant les points de collecte autour du lycée Aristide Briand.

Travail n°8 :

En respectant les consignes ci-dessus (travailler étape par étape), réaliser un programme python pour placer sur une carte l'ensemble des points de collecte des déchets multi matériaux près du lycée Aristide Briand de Saint-Nazaire

4) Projet

4.1 Cahier des charges



L'objectif est de réaliser une interface graphique avec **Tkinter** qui permet de visualiser des données au format **JSON** (ou csv, ou xml) issues d'un **OpenData** (ou autre serveur ou API, mais dans tous les cas, l'application doit venir récupérer les données en ligne).
Le sujet est libre mais le projet doit rester raisonnable afin d'être réalisé dans le temps imparti (3 séances de 2h soit 6h).
Une présentation de 5 min sera réalisée (à l'aide d'un diaporama).
Vous travaillerez en binôme, il faut donc bien se répartir les tâches.

Contraintes supplémentaires :

- Vous devrez mettre en œuvre au minimum une **classe** (POO).
- Vous devez réaliser la docstring !!!

4.2 Critères d'évaluation

Barème sur le projet (partie code) sur 20pts :

- les données sont récupérées en ligne (2pts)
- une ihm est réalisées (3pts)
- la visualisation des données est réalisée (3pts)
- une classe POO est réalisée (2pts)
- la complexité du traitement des données (rechercher, extraire, formater, convertir, trier,...) (2pts)
- l'IHM est soignée (2pts)
- l'IHM propose différentes fonctionnalités (2pts)
- Qualité de la visualisation des données (2pts)
- Qualité de la programmation POO (2pts)

Barème sur la présentation orale sur 20pts :

- Présentation structurée (4pts)
- Qualité orale (4pts)
- Qualité du diaporama (4pts)
- Contenu (8pts)

Si les tâches ne sont pas assez bien réparties, un coefficient est appliqué sur le note commune afin de différentier l'implication de chaque membre de l'équipe (coef allant de 0.5 à 1)

Ressources : Vous avez à votre disposition

- une fiche de conseils pour réussir votre présentation orale
- un exemple de diaporama (exemple de plan à suivre)