MasterCamp DataScience TD7

Ce TD se déroule:

1-Un exercice de récolte de donnée via API ou requête json
2- un exemple de web scraping avec Beautifulsoup
3- une heure de quiz sur nowledgeable

Exercice 1 : Donnée infoclimat

Info utile: *La "sérialisation" avec les modules pickle (pour object serialization) et json (pour text serialization).*
Il s'agit essentiellement des 4 méthodes: dump, dumps, load et loads

Il s'agit de tracer les températures à Paris sur une semaine via le webservice du site https://www.infoclimat.fr/
Infoclimat est une association à but non lucratif, essayez de ne pas "bombarder" le site par des requêtes répétées. Astuce spyder, favorisez le recours à **run current cell** et **run selection** pour ne pas exécuter votre script complet à chaque essai.

Infoclimat vous permet de récupérer les données de prévisions météo de Paris à 7 jours en JSON, XML, CSV. **Nous allons utiliser JSON** via l'API Infoclimat, qui nous retournera les prévisions détaillées pour Paris.
Le lien de l'API JSON et la description des données sont disponible ici:
https://www.infoclimat.fr/api-previsions-meteo.html?id=2988507&cntry=FR

a-Importer les modules suivant:

```
import json
import requests
```

b-Récupérez le lien de l'API JSON est affecté le à la variable url

```
url="http://www.infoclimat.fr/public-api/gfs/json?_ll=48.8534...
```

c-Ecrivez et testez le code suivant, et observez bien l'utilisation de json

```
try:
    api_request=requests.get(url)
    data=json.loads(api_request.content)
except Exception as e:
    data="Error..."
```

d-Vérifiez le type(data), il s'agit bien d'un dictionnaire, Explorez data. (for k in data, for k, v in..)
e-Vérifiez data des clés suivantes : request_state, request_key, message, model_run, source
f-Supprimez-les de votre dictionnaire.

g-Explorez la valeur pour une clé (texte représentant une date) et **vérifiez** bien qu'il s'agit d'un dictionnaire également.
Comme par exemple:

```
>>>print(data['2023-06-09 08:00:00'])
{'temperature': {'2m': 291.8, 'sol': 291.7, '500hPa': -0.1, '850hPa':
-0.1}, 'pression': {'niveau_de_la_mer': 101010}, 'pluie': 0,
'pluie_convective': 0, 'humidite': {'2m': 68.9}, 'vent_moyen': {'10m':
11.5}, 'vent_rafales': {'10m': 26.4}, 'vent_direction': {'10m': 436},
'iso_zero': 3488, 'risque_neige': 'non', 'cape': 0, 'nebulosite': {'haute':
0, 'moyenne': 0, 'basse': 0, 'totale': 40}}
```

h-Comment accéder à la température à 2m à cette date? La température au sol?
i-Comment accéder à l'humidité?

j-Importer datetime du module datetime

```
from datetime import datetime
```

Pour convertir une chaine de caractère de cette forme '2021-04-06 14:00:00' en un objet datetime nous pouvons utiliser la fonction datetime.strptime comme par exemple:

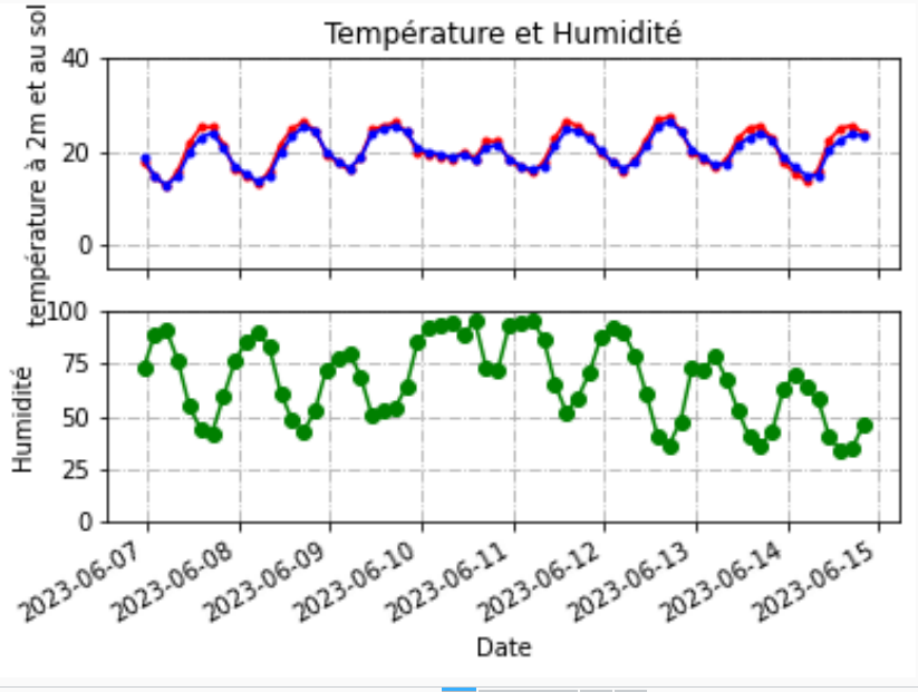
```
dt=datetime.strptime("2023-06-09 08:00:00", "%Y-%m-%d %H:%M:%S")
```

k-En s'appuyant sur la Compréhension des listes calculez les listes suivantes: lesDates, lesTempA2m, lesTempAuSol, lesHumidites
lesDates: liste contenant toutes les dates ayant servi de clé dans data (attention il s'agit des objets datetime.datetime et non pas des str.
lesTempA2m: liste contenant toutes les températures à 2m (en °C et non pas en kelvin)
lesTempAuSol: liste contenant toutes les températures au sol (en °C et non pas en kelvin)
lesHumiditesA2m: liste des humidités à 2m

l-Importez matplotlib.pyplot

```
import matplotlib.pyplot as plt
```

m-Tracez lesTempA2m, lesTempAuSol et lesHumiditesA2m en fonction de lesDates



> ***Astuce:*** *Pour afficher le plot dans une fenêtre séparée sous spyder :*
> *Tapez* ***%matplotlib auto*** *dans l'interpréteur python, pour rechanger taper* ***%matplotlib inline***

Infoclimat avec Pandas:

Refaites le même travail avec pandas, vous aurez besoin de pd.read_json pour lire api_request.content de pd.drop pour nettoyer et de pd.json_normalize pour splitter les informations en colonnes.

```
#éléments de réponses
dfjson=pd.read_json(          , orient=)
dfjson = dfjson.drop(                         , axis=0)
dfjson.index=pd.to_datetime(                  )
dfsplited = pd.json_normalize(              )
#ajouter time as index
#plot avec pandas
```

Part2 : web scraping with Beautiful soup

Avant de commencer cette partie, vous pouvez jeter un coup d'oeil sur cette introduction de l'utilisation du web scraping avec Beautiful soup
https://www.twilio.com/blog/web-scraping-analyse-html-python-beautiful-soup

Soit l'url suivant https://www.worldometers.info/world-population/population-by-country
Contenant des statistiques sur la population mondiale. Notre but est de récupérer cette table et la transformer en un DataFrame par web scrapping avec Beautiful soup sans avoir à copier manuellement ces données.

Commencer par importer les modules nécessaires:

```
import requests
from bs4 import BeautifulSoup
import pandas as pd
```

Définissez url:

```
url = "https://www.worldometers.info/world-population/population-by-country/"
```

Récupérez la page en appliquant requests.get

```
req = requests.get(url)
```

Créons un objet bs4.BeautifulSoup avec le fonction BeautifulSoup() appliqué à req.txt

```
soup = BeautifulSoup(req.text)
```

Regardez le help de la méthode soup.find_all et appliquez là pour récupérer le tableau de donnée (le premier tableau)

```
data = soup.find_all(
```

Regardez le help de la methode pd.read_html, puis appliquer là sur les données obtenues

```
df_population = pd.read_html(str(data))[0]
print(df_population.head())
```

Maintenant que vous avez votre dataframe, explorez le; puis utilisez la méthode to_csv pour l'enregistrer sur votre machine.

```
export_csv = df_population.to_csv(r"     ", index =      , header =     )
```