1- Import

In [ ]:
```
%reload_ext autoreload
%autoreload 2
import numpy as np
import pandas as pd
from process import *
```

2 - Fouille de données

In [ ]:
```
FILE = '../data/raw/cities.csv'
df = pd.read_csv(FILE)
```

In [ ]:
```
df.shape
```

Out[ ]:
```
(8040, 6)
```

In [ ]:
```
df.head(10)
```

Out[ ]:

	id	local_name	unique_name	latitude	longitude	population
0	5159	Padua, Veneto, Italia	padua	45.406435	11.876761	209678.0
1	76	Barcelona, Cataluña, España	barcelona	41.385064	2.173404	1611822.0
2	81	Basel, Basel-Stadt, Schweiz	basel	47.593437	7.619812	NaN
3	259	Erlangen, Bayern, Deutschland	erlangen	49.589674	11.011961	105412.0
4	11979	Balș, Olt, România	balș	44.353354	24.095672	NaN
5	10314	Град Пожаревац, Централна Србија, Србија	град-пожаревац	44.619095	21.176522	NaN
6	11155	Bussy-Lettrée, Grand-Est, France	bussy-lettree	48.804600	4.259500	NaN
7	11788	Chamonix, Valle d'Aosta, Italia	chamonix-italia	45.817156	6.952375	NaN
8	11984	Borger, Texas, United States of America	borger	35.667820	-101.397388	NaN
9	11825	okres Zvolen, Banskobystrický kraj, Slovensko	okres-zvolen	48.576181	19.137116	NaN

In [ ]:
```
df['latitude'].isnull().sum(), df['longitude'].isnull().sum()
```

Out[ ]:
```
(0, 0)
```

In [ ]:
```
df.dtypes
```

Out[ ]:
```
id              int64
local_name     object
unique_name    object
latitude      float64
longitude     float64
population    float64
dtype: object
```

Analyses

- id : identifiant
- local_name : triplé -> lieu, région, pays
- unique_name : nom du lieu
- latitude : coordonnée
- longitude : coordonnée
- population : nombres d'habitants

Plusieurs observations :

- Certains lieux ne représentent pas la ville mais une localisation plus précise (exemple : ligne 23 'Aix En Provence Gare Routiere'). Si on tente d'automatiser la recherche du nombre d'habitants à partir de la colonne local_name ou unique_name, il se peut que l'on n'obtienne aucun résultat.
- De ce fait, il est plutot logique d'avoir autant de valeur NaN dans la colonne population.
- On remarque la présence de plusieurs langues / alphabets dans la colonne local_name. Il se peut qu'il y ait des doublons. (exemple : Barcelona, Barcelone)

Data Preparation

In [ ]:
```
df_copy = df.copy()
```

In [ ]:
```
PRECISION = 4
index_duplicates = df_copy[df_copy.loc[:, ['latitude', 'longitude']].round(PRECISION).duplicated(keep=False)].index
```

In [ ]:
```
column_city = np.vectorize(get_city)(df_copy['latitude'], df_copy['longitude'])
```

In [ ]:
```
df_copy['city'] = column_city
```

In [ ]:
```
df_copy.loc[index_duplicates].sort_values('city')
```

Out[ ]:

	id	local_name	unique_name	latitude	longitude	population	city
796	3744	's-Hertogenbosch, Noord-Brabant, Nederland	s-hertogenbosch	51.697816	5.303675	NaN	's-Hertogenbosch
4423	8836	s Hertogenbosch (Bois-Le-Duc), Noord-Brabant, ...	s-hertogenbosch-	51.690600	5.295000	NaN	's-Hertogenbosch
2432	218	Den Bosch, Noord-Brabant, Nederland	den-bosch	51.697816	5.303675	NaN	's-Hertogenbosch
3742	9449	Bois-Le-Duc, Noord-Brabant, Nederland	bois-le-duc	51.690600	5.295000	NaN	's-Hertogenbosch
7869	7035	La Coruna San Cristobal, Galicia, España	la-coruna-san-cristobal	43.352800	-8.409700	NaN	A Coruña
...	...	...	...	...	...	...	...
7370	9370	Чернівці, Чернівецька область, Україна	чернівці	48.286470	25.937653	NaN	Чернівецька міська громада
7846	11883	Choumen, Choumla, Bulgaria	choumen	43.271300	26.936000	NaN	Шумен
1440	2412	Shumen, Shumen, Bulgaria	shumen	43.271272	26.936025	NaN	Шумен
122	1762	Horbanivka, Poltava region, Ukraine	horbanivka	49.556920	34.489860	NaN	Щербанівська сільська громада
7318	11682	Горбанівка, Полтавська область, Україна	горбанівка	49.556900	34.489900	NaN	Щербанівська сільська громада

1930 rows × 7 columns

Il y a bel et bien présence de doublons. Notre hypothèse sur les langues / alphabets est juste.
Par exemple, Bois-Le-Duc apparait 4 fois :

- En français
- En allemand
- En Néerlandais
- En Néerlandais avec une structure différente (la traduction a été ajoutée)

On pourrait choisir de supprimer les doublons. En contrepartie, il faudra fusionner les données dans les autres fichiers puisque qu'on suprrime des id.

In [ ]:
```
df_copy = df_copy.drop_duplicates(subset=['city'])
```

In [ ]:
```
percent_missing_before_merge = df_copy['population'].isnull().sum() * 100 / len(df_copy)
```

In [ ]:
```
column_population_unique_name = df_copy.apply(lambda x: get_population(x.unique_name) if pd.isnull(x.population) else x.population, axis=1)
column_population_city = df_copy.apply(lambda x: get_population(x.city) if pd.isnull(x.population) else x.population, axis=1)
```

In [ ]:
```
df_population = pd.DataFrame()
df_population['city_pop'] = column_population_city
df_population['unique_name_pop'] = column_population_unique_name
```

In [ ]:
```
df_population['city_pop'] = df_population['unique_name_pop'].where(pd.isnull(df_population['city_pop']),df_population['city_pop'])
```

In [ ]:
```
df_copy['population'] = df_population['city_pop'].where(pd.isnull(df_copy['population']),df_copy['population'])
```

In [ ]:
```
percent_missing_after_merge = df_copy['population'].isnull().sum() * 100 / len(df_copy)
print (f"pourcentage de valeurs manquantes dans la colonne population AVANT / APRES fusion: {percent_missing_before_merge} / {percent_missing_after_merge} %")
```

```
pourcentage de valeurs manquantes dans la colonne population AVANT / APRES fusion: 91.49550706033376 / 85.49422336328627 %
```

In [ ]:
```
df_copy.to_csv('../data/cleaned/cities_cleaned.csv')
```