Connaissance de pandas
Les DataFrame

Nous allons continuer notre exploration de pandas en nous consacrant cette fois ayx DataFrames qui peuvent être vues comme un assemblage de plusieurs Series.

Je vous propose de regarder un fichier en open data qui est le référentiel des gares voyageurs de la SNCF. Ce fichier se trouve sur le site Internet de données ouvertes de la SNCF (https://ressources.data.sncf.com), j'ai téléchargé la version du 4 décembre 2020 mais rien ne vous empêche de charger la dernière version.

Les données dans le fichier ne sont pas séparées par un virgule mais par un point-virgule d'où le l'attribut sep qui indique la qualité du séparateur.

```
Entrée [1]: import pandas as pd

Entrée [2]: données  = pd.read_csv('c:/Temp/referentiel-gares-voyageurs.csv', sep=';')
            données
```

Out[2]:

	Code plate-forme	Code UIC	Date fin validité plateforme	Intitulé plateforme	Code postal	Code Commune	Commune	Code département	Département	Longitude	...	Intitulé fronton de gare	Gare DRG	Gare étrangère
0	00006-1	87784884	NaN	Ur - Les Escaldes	66760	218	Ur	66	Pyrénées-Orientales	1.940482	...	Ur les Escaldes	True	False
1	00011-1	87784835	NaN	Saillagouse	66800	167	Saillagouse	66	Pyrénées-Orientales	2.032113	...	Saillagouse	True	False
2	00014-1	87784801	NaN	Bolquère - Eyne	66210	27	Cabanasse	66	Pyrénées-Orientales	2.087559	...	Bolquère - Eyne	True	False

La base de données est importante puisque le fichier fait

2866	03215-1	87697359	NaN	Le Bourget T11	93350	13	Bourget	93	Seine-Saint-Denis	2.423160	...	Le

2867 rows × 28 columns

2867lignes x 28 colonnes

Le DataFrame

Après avoir vu l'élément de base qui est la série, le ou la DataFrale a le m*me principe qu'une Séries mais cette fois-ci en deux dimensions

Index et Colonnes

Comme pour une Series, une DataFrame a un index qui permet de donner un nom à chaque colonne. Dans notre cas, et en l'absence d'informations supplémentaires l'index des colonnes est une valeur numérique qui commence à 0 et finit à 2866.

Mais comme on est maintenant dans un environnement à deux dimensions, on a un deuxième index cette fois-ci pour les colonnes.

```
Entrée [3]: données.index

   Out[3]: RangeIndex(start=0, stop=2867, step=1)

Entrée [4]: données.columns

   Out[4]: Index(['Code plate-forme', 'Code UIC', 'Date fin validité plateforme',
                  'Intitulé plateforme', 'Code postal', 'Code Commune', 'Commune',
                  'Code département', 'Département', 'Longitude', 'Latitude',
                  'Segment DRG', 'Niveau de service', 'RG', 'TVSs', 'SOPs', 'Gare',
                  'Intitulé gare', 'Intitulé fronton de gare', 'Gare DRG',
                  'Gare étrangère', 'DTG', 'Région SNCF', 'Unité gare', 'UT',
                  'Nbre plateformes', 'TVS', 'WGS 84'],
                 dtype='object')
```

Il est très facile de changer l'index d'une DataFrame avec la méthode setindex :

```
Entrée [5]: données.set_index('Intitulé fronton de gare', inplace=True)
            données
```

Out[5]:

Intitulé fronton de gare	Code plate-forme	Code UIC	Date fin validité plateforme	Intitulé plateforme	Code postal	Code Commune	Commune	Code département	Département	L
Ur les Escaldes	00006-1	87784884	NaN	Ur - Les Escaldes	66760	218	Ur	66	Pyrénées-Orientales	
Saillagouse	00011-1	87784835	NaN	Saillagouse	66800	167	Saillagouse	66	Pyrénées-Orientales	

et pour la lire de manière plus agréable, il ne nous reste plus qu'à trier notre DataFrame en utilisant l'index :

```
Entrée [6]: données.sort_index()
```

Out[6]:

Intitulé fronton de gare	Code plate-forme	Code UIC	Date fin validité plateforme	Intitulé plateforme	Code postal	Code Commune	Commune	Code département
Abancourt	02266-1	87313759	NaN	Abancourt	60220	1	Abancourt	60
Abbaretz	01604-1	87481614	NaN	Abbaretz	44170	1	Abbaretz	44

Construction des DataFrames

Tout comme les Series, il est possible de créer des dataFrame directement dans pandas. La construction ressemble au principe de construction des dictionnaires en python ; les clefs seront les colonnes et les valeurs associées les colonnes.

```
Entrée [10]: mes_données = pd.DataFrame({'pays': ['France','Allemagne','Maroc'],
                                          'en Europe': [True, True, False],
                                          'habitants': [65, 85, 37]}
                                         )
```

```
Entrée [11]: mes_données
```

Out[11]:

	pays	en Europe	habitants
0	France	True	65
1	Allemagne	True	85
2	Maroc	False	37

mais comme pandas traite souvent des tableaux de données de taille importante, il dispose de plusieurs fonctions permettant de charger des DataFrame :

- `pd.read_csv`: lecture de fichiers CSV
- `pd.read_excel`: lecture de ficiers excel
- `pd.read_hdf`: fichiers hdf
- `pd.read_sql`: BdD SQL
- …

Se familiariser avec ses données

Reprenons notre fichier sur les gares

```
Entrée [15]: données = pd.read_csv('./referentiel-gares-voyageurs.csv', sep=';')
             données
```

Out[15]:

	Code plate-forme	Code UIC	Date fin validité plateforme	Intitulé plateforme	Code postal	Code Commune	Commune	Code département	Département	Longitude	...	Intitulé fronton de gare	Gare DRG	Gare étrangère
0	00006-1	87784884	NaN	Ur - Les Escaldes	66760	218	Ur	66	Pyrénées-Orientales	1.940482	...	Ur les Escaldes	True	False
1	00011-1	87784835	NaN	Saillagouse	66800	167	Saillagouse	66	Pyrénées-Orientales	2.032113	...	Saillagouse	True	False
2	00014-1	87784801	NaN	Bolquère - Eyne	66210	27	Cabanasse	66	Pyrénées-Orientales	2.087559	...	Bolquère - Eyne	True	False

Comme ce fichier est très volumineux, tout comme dans les Series (2868 lignes), il est possible avec les commandes head() et tail() de visualiser les premières ou dernières lignes de nos données.

Il existe une commande intéressante qui permet de récupérer des lignes de manière aléatoire au sein de la base, la méthode sample() :

```
Entrée [16]: données.sample(5)
```

Out[16]:

	Code plate-forme	Code UIC	Date fin validité plateforme	Intitulé plateforme	Code postal	Code Commune	Commune	Code département	Département
760	01062-1	87611087	NaN	Martres-Tolosane	31220	324	Martres-Tolosane	31	Haute-Garonne
1643	02515-1	87276022	NaN	Enghien-les-Bains	95880	210	Enghien-les-Bains	95	Val-d'Oise
2591	01520-1	87491100	NaN	Pons	17800	283	Pons	17	Charente-Maritime
401	02209-1	87317610	NaN	Beau Marais	62100	193	Calais	62	Pas-de-Calais
240	01320-1	87576306	NaN	Bigny	18190	270	Vallenay	18	Cher

5 rows × 28 columns

Comme pour une Series, la commande len() donne le nombre de lignes, pour avoir le nombre de colonnes, il faut utiliser l'attribut colums de la base :

```
Entrée [19]: len(données.columns)
  Out[19]: 28
```

Il est possible d'utiliser les commandes dtypes afin de connaître les types de données de notre DataFrame et describe() qui donne des informations numériques (sur les colonnes à valeur numérique bien sûr)

```
Entrée [21]: données.describe()
```

Out[21]:

	Code UIC	Date fin validité plateforme	Code postal	Code Commune	Code département	Lo
count	2.867000e+03	0.0	2867.000000	2867.000000	2867.000000	2863
mean	8.749419e+07	NaN	53109.605162	231.371120	52.745378	2
std	2.036322e+05	NaN	26277.865673	181.668411	26.264106	2
min	8.700148e+07	NaN	1000.000000	1.000000	1.000000	-4
25%	8.731381e+07	NaN	31835.000000	81.000000	31.000000	1
50%	8.749121e+07	NaN	58800.000000	188.000000	58.000000	2
75%	8.768441e+07	NaN	74510.000000	336.500000	74.000000	4
max	8.798872e+07	NaN	95880.000000	907.000000	95.000000	8

La moyenne du code postal n'est effectivement pas une information très pertinente.

Si vous voulez mettre les noms de colonnes dans une véritable liste python, hop un petit script :

```
Entrée [23]: mes_colonnes = [x  for x in données.columns]
             mes_colonnes
  Out[23]: ['Code plate-forme',
            'Code UIC',
            'Date fin validité plateforme',
            'Intitulé plateforme',
            'Code postal',
            'Code Commune',
            'Commune',
            'Code département',
            'Département',
            'Longitude',
            'Latitude',
```

Récupérer des éléments d'un DataFrame

Un peu comme avec les Series, les DataFrames utilisent une version généralisée de loc,

iloc et des tests logiques

iloc

iloc correspond à index location. Ici, pour récupérer une données précise, on utilisera donc, le numéro de la ligne et celui de la colonne :

```
Entrée [26]: données.iloc[2,6]
  Out[26]: 'Cabanasse'
```

Il est tout à fait possible de récupérer des tranches de données en utilisant les :.

Out[27]:

	Code plate-forme	Code UIC	Date fin validité plateforme
0	00006-1	87784884	NaN
1	00011-1	87784835	NaN

Si l'on veut toutes les colonnes, il faut comme avec numpy utiliser :

Par exemple, pour obtenir les trois premières lignes avec toutes les colonnes :

```
Entrée [28]: données.iloc[0:2,:]
```

Out[28]:

	Code plate-forme	Code UIC	Date fin validité plateforme	Intitulé plateforme	Code postal	Code Commune	Commune	Code département	Départe
0	00006-1	87784884	NaN	Ur - Les Escaldes	66760	218	Ur	66	Pyrén Orien
1	00011-1	87784835	NaN	Saillagouse	66800	167	Saillagouse	66	Pyrén Orien

2 rows × 28 columns

loc

Tout comme dans les Series, loc se généralise dans le cas des DataFrame.

```
Entrée [4]: données.loc[1, 'Commune']
    Out[4]: 'Saillagouse'
```

Remarquez bien qu'ici le 1 correspond à l'index de la ligne pas le numéro et que si on change l'index des lignes de notre DataFrame, il faudrait également changer l'index :

```
Entrée [6]: données.set_index('Intitulé fronton de gare', inplace=True)
            données.loc['Saillagouse', 'Commune']
    Out[6]: 'Saillagouse'
```

Il est possible d'obtenir des tranches du DataFrame comme pour une Series :

```
Entrée [11]: données.loc[1:4, 'Commune']
    Out[11]: 1    Saillagouse
             2      Cabanasse
             3    Canaveilles
             4         Olette
             Name: Commune, dtype: object
```

Note importante : le fonctionnement de pandas est un peu différent de celui de python où lorsqu'on travaille sur une tranche, on n'a pas l'extrémité (par exemple sur les chaînes de caractère)

Note 2 :

Si l'on passe un unique argument à la méthode loc, pandas considérera que l'on travaille sur les lignes

Les fonctions logiques

L'utilisation des fonctions logiques et booléennes est très utilisée dans le traitement des DataFrame, il y a plusieurs éléments à considérer :

1. si un tableau de booléens est fourni à loc, il considérera que ce sont les lignes
2. si le tableau de booléens a un Index (c'est une Series), l'alignement se fera sur les valeurs de la colonne correspondant à l'index
3. pour faire un test sur les colonnes, il faudra utiliser **.loc[:, test]**

Mais quelques exemples valent mieux qu'une longue explication :

Entrée [24]:
```
données_triées =  données.sort_values('Code Commune')
données_triées.head(5)
```

Out[24]:

	Code plate-forme	Code UIC	Date fin validité plateforme	Intitulé plateforme	Code postal	Code Commune	Commune	dép
2021	01460-1	87545269	NaN	Ablon-sur-Seine	94480	1	Ablon-sur-Seine	
985	02207-1	87319012	NaN	Aix-en-Provence TGV	13592	1	Aix-en-Provence	
1795	00291-1	87751404	NaN	Aix-en-Provence	13100	1	Aix-en-Provence	

Entrée [25]:
```
données_triées['Code département']
```

Out[25]:
```
2021    94
985     13
1795    13
1356    23
2534    47
        ..
424      2
2165    62
1576    62
1586    62
389     62
Name: Code département, Length: 2867, dtype: int64
```

ici on obtient une Series, cas n°2 on s’aligne sur les valeurs de la colonne

```
Entrée [26]: données_triées[données_triées['Code département'] > 75]
```

Out[26]:

	Code plate-forme	Code UIC	Date fin validité plateforme	Intitulé plateforme	Code postal	Code Commune	Commune	Code département	Département
2021	01460-1	87545269	NaN	Ablon-sur-Seine	94480	1	Ablon-sur-Seine	94	Val-de-Marne
782	01171-1	87592667	NaN	Aixe-sur-Vienne	87700	1	Aixe-sur-Vienne	87	Haute-Vienne
403	02219-1	87317362	NaN	Abbeville	80100	1	Abbeville	80	Somme

Pour un tableau de booléens sans index, on crée une liste de booléens dont la taille est égale au nombre de lignes du DataFrame

```
Entrée [41]: def alea():
                 if random.randint(0, 1) == 0:
                     return True
                 else:
                     return False
```

```
Entrée [42]: tab = [alea() for x in range(len(données_triées))]
```

Et on applique la règle n°1 :

```
Entrée [45]: données_triées.loc[tab]
```

Out[45]:

	Code plate-forme	Code UIC	Date fin validité plateforme	Intitulé plateforme	Code postal	Code Commune
1795	00291-1	87751404	NaN	Aix-en-Provence	13100	1
2604	01604-1	87481614	NaN	Abbaretz	44170	1
1500	01791-1	87444711	NaN	Alençon	61000	1

<u>Note sur les crochets :</u>

Il est possible d'utiliser les crochets sans loc et iloc. Le résultat de l'exécution sera dans ce cas le suivant :

- si on met une colonne ou une liste de colonnes, ces colonnes seront retournées
- si on entre une tranche (ie **[:]**), cela revient à sélectionner les index comme avec iloc

Les DataFrames : une collection de Series

Extraire une colonne d'un DataFrame la transforme en Series, extraire une liste de colonnes donne un DataFrame (ce qui peut semble logique). Attention, cependant à la notation

```
Entrée [53]: les_départements = données['Code département']
             type(les_départements)
Out[53]: pandas.core.series.Series

Entrée [54]: plusieurs_colonnes = données[['Département','Longitude']]
             type(plusieurs_colonnes)
Out[54]: pandas.core.frame.DataFrame
```

Les exercices

Prenez maintenant votre notebook ou éditeur préféré et reprenez le fichier fourni par la SNCF sur les gares

1. combien y a-t-il de lignes dans le fichier ?
2. Combien y a-t-il de colonnes dans le fichier ?
3. Lorsque vous faîtes données.columns, pandas n'affiche pas toutes les colonnes sous forme d'une liste mais sous la forme d'un Index pandas. Écrire un script python qui donne les labels des colonnes sous formes de listes
4. Comme vous pouvez le constater , le nombre de colonnes est très important, nous allons réduire notre base aux colonnes les plus intéressantes.. Je vous propose de reprendre une base avec les colonnes suivantes 'Commune', 'Département', 'Longitude', 'Latitude', 'Intitulé fronton de gare', 'Nbre plateformes'

5. Pour avoir une idée des données que nous sommes en train d'examiner, utiliser une commande qui vous donne un échantillon de 20 lignes
   Comme vous pouvez le voir, la colonne Nbe_plateformes ne semble pas contenir d'informations pertinentes. Utilisez la méthode value_counts(), qu'en déduisez-vous ?

6. Supprimer cette colonne 'Nbre plateformes »

```
Entrée [18]: mes_données['Latitude'].describe()

Out[18]: count    2863.000000
         mean       47.188418
         std         2.159540
         min        42.419967
         25%        45.546862
         50%        47.633284
         75%        48.897603
         max        51.030470
         Name: Latitude, dtype: float64
```

7. Quelle est la latitude la plus élevées de toutes les gares ?
8.

```
Entrée [32]: mes_données[(mes_données['Département'] == 'Nord') | (mes_données['Département'] == 'Pas-de-Calais')]
```

Out[32]:

	Commune	Département	Longitude	Latitude
385	Cattenières	Nord	3.333788	50.128074
386	Bertry	Nord	3.448996	50.091127
387	Maurois	Nord	3.456255	50.067451
388	Cantin	Nord	3.121943	50.312455
389	Libercourt	Pas-de-Calais	3.008670	50.480250

Quelle est la latitude la plus importante des départements du nord et du pas de Calais ?