Séance du 19 janvier 2023

Exploration de la bibliothèque `pandas`

Prenez l'habitude de faire
`import pandas as pd`

La bibliothèque pandas offre deux classes `Series` et `DataFrame`.

`Series`

Une série **Series** est un conteneur ordonné à une dimension. Les valeurs peuvent être accédées soit par la position (comme une liste python), soit par un index spécifique (comme un dictionnaire python)

Procédez pas à pas, et pour chaque commande expliquez ce qui se passe

```
>>> import pandas as pd
>>> a = [-1, 2.5, -3, 5, -2, 8, 17] # une liste de valeurs
>>> s1 = pd.Series(a) # transformation en une série
>>> s1
>>> s1.values
>>> s1.index
>>> s1.index.values
>>> s1[1]
>>> s1[1:4]
>>> s1[[1,3,5]]
>>> s1[s1 < 0]
>>> tags = list("ABECIDF")
>>> tags
>>> s2 = pd.Series(a, index=tags)
>>> s2
>>> s2.values
>>> s2.index
>>> s2.index.values
>>> s2[1]
>>> s2[1:4]
>>> s2[[1,3,5]]
>>> s2[s2 < 0]
>>> s2['A':'D']
>>> s2[['A', 'B', 'C']]
```

On peut modifier une valeur, ou un groupe de valeurs

```
>>> s1[1] = 42
>>> s1[2:] = 13
>>> s1[13] = -24
>>> s1.index
>>> s1.index.values
>>> s2['A'] = 13
>>> s2[2] = 37
>>> s2.F = -53
>>> s2['G'] = 1.25
>>> s2[14] = 5
>>> s2.index.values
>>> s2[s2%2==1] = 0
>>> s2
```

Il est tout à fait possible d'avoir d'autres types d'index, typiquement des dates pour pouvoir jouer avec les séries temporelles. La bibliothèque offre deux types `DatetimeIndex` et `TimedeltaIndex` qui sont définis en donnant une *date* ou *heure* de début, une *période* et une *fréquence*

symbole	signification
`N`	nanosecondes
`U`	microsecondes
`L`	millisecondes
`S`	secondes
`T`	minutes
`H`	heures
`D`	jours
`W`	semaines
`M`	mois
`Q`	trimestres
`Y`	années

Note

Les fréquences mois, trimestre, année fixent par défaut leur valeur au dernier jour de la période, si on préfère le premier jour il faut écrire `MS` `QS` et `YS`. De même la fréquence `W` se cale sur le dernier jour de la semaine (le dimanche), si on veut se baser sur un autre jour de la semaine, il faut utiliser `W-day` avec $\text{day} \in \{\text{MON}, \text{TUE}, \text{WED}, \text{THU}, \text{FRI}, \text{SAT}, \text{SUN}\}$

par exemple essayez, comprenez, expérimentez

```
>>> import numpy as np
>>> import random
>>> tmps = pd.date_range("2020-06-07", periods=13, freq='Q')
>>> valeurs = np.random.rand(tmps.size)
>>> s3 = pd.Series(valeurs, index=tmps)
>>> s3
>>> s3.index
>>> s3.index.values
>>> tmps = pd.timedelta_range("15H", periods=7, freq="1H15T")
>>> valeurs = np.random.rand(tmps.size)
>>> s4 = pd.Series(valeurs, index=tmps)
>>> s4
>>> tmps = pd.date_range("2021", end='2021-03', freq='W')
>>> s5 = pd.Series(random.choices([0,1], k=tmps.size), index=tmps)
>>> s5
```

Exercice

créez une série indexée sur les journées de janvier et février 2020 et dont les valeurs sont positives pour la première quinzaine du mois, négatives pour la seconde quinzaine du mois
créez une seconde série avec les mêmes index, mais les valeurs seront aléatoirement prises dans 1,2,3
affichez la première série si la valeur dans la seconde série est impaire

DataFrame

Une table est en `pandas` une **DataFrame**, c'est l'agglomération de `Series` partageant le même index. Chaque série est alors une colonne de la table.

Une première façon de créer une table est de construire une liste de listes de valeurs

```
>>> m = [ [1,2,3], [4,5,6], [2,-2,2], [0,0,1]]
>>> df = pd.DataFrame(m)
>>> df
>>> df.index
>>> df.index.values
>>> df.columns
>>> df.columns.values
>>> df.values
>>> df.ndim
>>> df.size
>>> df.shape
>>>
```

L'indexation des tables est un peu déroutante au départ, les *indices* font référence aux colonnes, tandis que les *slices* sont associées aux lignes

```
>>> df[2]
>>> df[1:]
```

Si on veut appliquer les tranches aux colonnes ou les indices aux lignes, on peut passer par la transposition (comme en `numpy` via la notation `.T`)
On aussi peut utiliser `.loc` ou `.iloc`

```
>>> df.T
>>> df.T[2]
>>> df.T[1:]
>>> df.T[1:].T
>>> df.loc[2]
>>> df.loc[:,1:]
```

On peut accéder à une partie de la table aussi bien en énumérant des index qu'en utilisant des prédicats, le symbole `NaN` se lit **Not a Number**

```
>>> df[[2,0]]
>>> df[df%3==1]
```

On peut modifier a posteriori les étiquettes, que ce soient celles des lignes ou des colonnes

```
>>> df.columns=list("ABC")
>>> df.index=list("aeio")
>>> df
>>> df.A
>>> df['A']
>>> df.loc['e':]
>>> df.iloc[1:3]
```

> **Note**
>
> `loc` est utilisé lorsque l'on cherche à accéder aux données par leurs étiquettes
> `iloc` est utilisé lorsque l'on cherche à accéder aux données par leur position

L'usage le plus courant des tables est d'avoir pour les colonnes des étiquettes, et pour les lignes soient des index entier, soient des index temporel. L'autre intérêt des tables c'est la possibilité de pouvoir mixer des informations de nature différentes.

L'initialisation d'une table se fait le plus souvent de deux manières

1. Utilisation de dictionnaires regroupant plusieurs séries
2. La lecture de fichier

```
>>> dates = pd.date_range("2020-01", end="2022-06", freq="W-MON")
>>> categories = "trottinette vélo voiture marche tram bus train".split()
>>> true_false = (dates.month > 3) * (dates.month < 7)
>>> delay = (pd.TimeStamp('now') - dates).round('D')
>>> big_df = pd.DataFrame({'mobility':
pd.Categorical(random.choices(categories, k=dates.size)), 'start': 0,
'stop': 0.1 * np.array(random.choices(list(range(100)), k=dates.size)),
'delay':delay, 'date':dates, 'target':true_false})
```

Sur moodle récupérez le fichier `sales.csv` ce fichier contient pour chaque investissement dans les média (TV, Radio, Journaux), quel est l'impact sur les ventes. Après avoir jeter un œil dans le fichier, on voit qu'il y a une première ligne indiquant les colonnes et que chaque valeur est séparée de la suivante par une virgule

```
>>> ventes = pd.read_csv("sales.csv") # help(pd.read_csv) pour les options
```

Quelques commandes

Je suppose que vous avez les deux tables accessibles `big_df` et `ventes`. En général les données manipulées sont assez volumineuses, il est donc bien pratique de pouvoir n'examiner qu'une partie de l'information, avoir des informations sur la nature du contenu ou encore sur les statistiques

```
>>> big_df.head()
>>> big_df.head(2)
>>> big_df.tail()
>>> big_df.tail(2)
>>> big_df.info()
>>> big_df.describe()
>>> big_df.describe(include='all')
>>> big_df.mobility.unique()
```

Exercice

Dans `big_df` combien y-at-il de données telles que 'mobility' soit trottinette. Quelle est la valeur moyenne de 'delay' pour ces valeurs ? Quelle est la valeur du premier quartile de 'delay' ? Quel est le maximum de la variable 'stop' ?

On peut appliquer des fonctions telles que `.min()` `.max()` `.mean()` `.sum()` `.quantile(v)` sur des séries numériques

```
>>> big_df.stop.min()
>>> big_df.stop.mean()
>>> ventes['Sales'].sum()
>>> for k in (.25, .5, .75): print(k, '->', ventes.Sales.quantile(k))
```

On peut ajouter/supprimer des nouvelles informations

```
>>> big_df.shape
>>> big_df['OH'] = range(df.shape[0], -1, -1)
>>> big_df['carre'] = df.stop**2
>>> big_df['couleur'] = pd.Series({0:'green', 25:'yellow', 100:'red'})
>>> big_df.drop('carre', axis=1)
>>> big_df.columns
>>> big_df.drop(2)
>>> big_df.head()
>>> big_df.drop('carre', inplace=True)
>>> big_df.columns
>>> big_df.drop(2, inplace=True)
>>> big_df.head()
>>> big_df.iloc[2] = {'mobility': 'vélo', 'stop'=42}
>>> big_df.head()
>>> big_df.iloc[2,3]
>>> big_df.iloc[2,3] = pd.Timedelta('200 days 01:01:01')
>>> big_df.head()
>>> ventes['cout'] = ventes['TV']+ventes['Radio']+ventes.Newspaper
```

On peut remplacer des valeurs manquantes

```
>>> big_df['couleur'].fillna('black')
>>> big_df['couleur'].fillna(method='ffill')
>>> big_df['couleur'].fillna(method='bfill')
>>> big_df['stop'].fillna(big_df.stop.mean())
```

Remarque la commande `fillna()` renvoie une copie de la série ou la table modifiée, si vous souhaitez que la commande soit effective sur la série (ou la table originale) il faut utiliser le paramètre `inplace=True`

La commande `.count()` permet de connaître le nombre de valeurs non nulle, `.value_counts()` permet de savoir la répartition de chaque valeur, une option intéressante possible `.value_counts(normalize=True)`

Exercice

Pour la série 'couleur' de `big_df` donnez les valeurs obtenues par les méthodes `.count` et `.value_counts` ; puis appliquez les 3 manières de remplacement et donnez les scores que vous obtenez
Dans la table `ventes` ajouter une colonne "facteur" qui soit le ratio entre "Sales" et

"TV". Quelles sont les statistiques pour la série "facteur" ? Combien y-a-t-il de valeurs dans le dernier quartile ?
Quelle est la commande pour supprimer *définitivement* la colonne "facteur" ?
Comment supprimer les données de la table `ventes` qui ont une valeur "TV" > 150 ?
On souhaite rajouter une colonne 'budget_pub' dans la table `ventes` qui contiennent 'gros' si la valeur "TV" est > 200, 'moyen' si la valeur est > 100, 'bas' sinon.