Informatique & Algorithmique
2023 – 2024
.ᔓ pandas ᔕ.

`pandas` est une librairie python qui permet de manipuler facilement des données à analyser, notamment :

- manipuler des tableaux de données avec des étiquettes de variables (colonnes) et d'individus (lignes). Un tel tableau est appelé `DataFrame`.
- importer et la lire ces données sous la forme d'un fichier `csv` (comma separated values : valeurs séparées par une virgule) provenant de sources diverses comme le site data.gouv.fr ou le site de l'Insee.
- tracer des graphes à partir de ces données grâce à matplotlib.

Pour utiliser `pandas`, on importe la librairie avec la commande `import pandas` ou sous la forme d'un alias :

```
import pandas as pd
```

1 La commande `DataFrame`.

On considère une liste d'étudiants et leurs notes au cours des trois trimestres d'une année scolaire. Ces résultats sont compilés dans un tableau EXCEL dans un fichier noté **EXEMPLE1**, on importe les données sous la forme d'un fichier `csv` à l'aide des commandes suivantes :

```
1 import pandas
2 donnees=pandas.read_csv('EXEMPLE1.csv',sep=';',decimal=',',encoding='windows-1252')
3 print(donnees)
```

donne :

```
*** Console de processus distant Réinitialisée ***
         NOM  TRI 1  TRI 2  TRI 3
0      Laura   16.2   15.7   13.3
1    Thibaut   14.2   16.8    7.7
2      Shane   14.5   14.6   13.5
3     Maxime   15.5   15.2   14.0
4    Antoine   15.5   15.8   14.7
5     Emilie   13.8   15.1   15.7
6    Gustave   13.6   17.5   10.4
7       Paul   12.5   10.6   10.5
8      Robin   15.4   14.3   12.3
9    Antoine    9.6   11.6   12.0
10  Caroline   11.0   17.5   17.1
11      Iris   11.5   11.0   15.5
>>>
```

On peut transformer ce tableau avec la librairie `pandas` pour le mettre sous la forme d'un objet sur lequel on peut travailler : un `DataFrame`. Le plus simple est de procéder comme suit :

```
4  donnees_df=pandas.DataFrame(donnees)
```

Il est possible d'isoler des lignes et/ou des colonnes dans un `DataFrame` :
par exemple avec

```
5  print(donnees_df.NOM)
```

ou bien

```
5  print(donnees_df['NOM'])
```

Si on tape `print(donnees_df.NOM.values)` on obtient la liste des différents prénoms sous la forme d'une liste.
Accédons maintenant aux lignes, par exemple l'étudiant **Shane** par sa position (indice 2) avec `iloc`.

```
print(donnees_df.iloc[2])
```

affiche

```
*** Console de processus distant Réinitialisée ***
NOM      Shane
TRI 1     14.5
TRI 2     14.6
TRI 3     13.5
Name: 2, dtype: object
>>>
```

2 Les autres commandes au programme.

2.1 La commande head

La commande

```
print(donnees_df.head())
```

affiche par défaut les cinq première lignes du DataFrame `donnees_df`. On peut personnaliser le nombre de lignes en ajoutant un entier comme argument de `head()`

```
*** Console de processus distant Réinitialisée ***
       NOM  TRI 1  TRI 2  TRI 3
0    Laura   16.2   15.7   13.3
1  Thibaut   14.2   16.8    7.7
2    Shane   14.5   14.6   13.5
3   Maxime   15.5   15.2   14.0
4  Antoine   15.5   15.8   14.7
>>>
```

2.2 La commande shape

La commande

```
print(donnees_df.shape)
```

affiche la taille du DataFrame sous la forme d'un tuple `(nombre de lignes , nombre de colonnes)` soit ici

```
(12,4)
```

2.3 La commande describe

La commande

```
print(donnees_df.describe())
```

produit l'affichage :

```
*** Console de processus distant Réinitialisée ***
           TRI 1      TRI 2      TRI 3
count  12.000000  12.000000  12.000000
mean   13.608333  14.641667  13.058333
std     2.062857   2.392206   2.647969
min     9.600000  10.600000   7.700000
25%    12.250000  13.625000  11.625000
50%    14.000000  15.150000  13.400000
75%    15.425000  16.050000  14.900000
max    16.200000  17.500000  17.100000
>>>
```

on peut spécifier un peu

- `donnees_df.mean()` donne la moyenne par colonne
- `donnees_df.std()` donne l'écart-type de la série par colonne
- `donnees_df.median()` donne la médiane de la série par colonne
- `donnees_df.count()` donne le nombre d'éléments de la série par colonne

le reste est clair

2.4 La commande sort_values

permet le tri de données dans le DataFrame par exemple :
La commande

```
print(donnees_df.sort_values(by=['NOM']))
```

affiche

```
*** Console de processus distant Réinitialisée ***
         NOM  TRI 1  TRI 2  TRI 3
4    Antoine   15.5   15.8   14.7
9    Antoine    9.6   11.6   12.0
10  Caroline   11.0   17.5   17.1
5     Emilie   13.8   15.1   15.7
6    Gustave   13.6   17.5   10.4
11      Iris   11.5   11.0   15.5
0      Laura   16.2   15.7   13.3
3     Maxime   15.5   15.2   14.0
7       Paul   12.5   10.6   10.5
8      Robin   15.4   14.3   12.3
2      Shane   14.5   14.6   13.5
1    Thibaut   14.2   16.8    7.7
>>>
```

Le tri est par défaut croissant, si on souhaite un tri par ordre décroissant on doit écrire :

```
print(print(donnees_df.sort_values(by='TRI 3',ascending=False)))
```

affiche

```
*** Console de processus distant Réinitialisée ***
         NOM  TRI 1  TRI 2  TRI 3
10  Caroline   11.0   17.5   17.1
5     Emilie   13.8   15.1   15.7
11      Iris   11.5   11.0   15.5
4    Antoine   15.5   15.8   14.7
3     Maxime   15.5   15.2   14.0
2      Shane   14.5   14.6   13.5
0      Laura   16.2   15.7   13.3
8      Robin   15.4   14.3   12.3
9    Antoine    9.6   11.6   12.0
7       Paul   12.5   10.6   10.5
6    Gustave   13.6   17.5   10.4
1    Thibaut   14.2   16.8    7.7
>>>
```

* * *