Chapitre I Introduction à Pandas

Cours Data Science M1 IA-GL

Ilyas Bambrik

2024

Table des matières

Introduction 3

I - **Module pandas** 4

II - **DataFrame** 5

III - **Series** 6

IV - **Lecture de DataFrame à partir de fichier** 7

V - **Indexation et accès aux valeurs** 9

VI - **Selection par label** 12

VII - **Changement d'index** 14

VIII - **Sélection conditionnelle** 16

IX - **Affectation** 19

Références 20

Introduction

Dans ce chapitre, vous apprendrez l'essentiel sur pandas, la bibliothèque Python la plus populaire pour l'analyse de données. Pour ceci, vous réaliserez plusieurs exercices pratiques avec des données réelles.

Module pandas

Pour utiliser pandas[1*], vous commencerez généralement par la ligne de code suivante. Cette instruction importe *pandas* sous le pseudo *pd*.

```
1 import pandas as pd
```

Dans le domaine Data Science et Analyse de donnée, il existe deux classes d'objets implémentées dans pandas essentiels pour la représentation des données: le *DataFrame* et la *Series*.

DataFrame

Un DataFrame est un tableau. Il contient une liste d'entrées individuelles, chacune ayant une certaine valeur. Chaque entrée correspond à une ligne (ou enregistrement comme dans les BDD) et une colonne. Par exemple, considérons le DataFrame simple suivant :

```
pd.DataFrame({'Yes': [50, 21], 'No': [131, 2]})
```

Dans cet exemple, l'entrée « 0, No» a la valeur 131. L'entrée « 0, Yes» a la valeur 50, et ainsi de suite. Les entrées DataFrame ne se limitent pas aux entiers.

	Yes	No
0	50	131
1	21	2

```
pd.DataFrame({'Bob': ['I liked it.', 'It was awful.'], 'Sue': ['Pretty good.',
 'Bland.']})
```

	Bob	Sue
0	I liked it.	Pretty good.
1	It was awful.	Bland.

Nous utilisons le constructeur *pd.DataFrame()* pour générer un objet *DataFrame*. Le constructeur *DataFrame* prend en paramètre un dictionnaire dont les clés sont les noms de colonnes (Bob et Sue dans cet exemple) et dont les valeurs sont une liste d'entrées.

Les étiquettes sont attribués aux colonnes, et un nombre croissant à partir de 0 (0, 1, 2, 3, ...) pour les étiquettes de lignes. Ceci est généralement acceptable, mais nous souhaitons souvent attribuer ces étiquettes nous-mêmes. La liste des étiquettes de ligne utilisées dans un *DataFrame* est appelée index et joue le rôle d'un identifiant pour les lignes. Lors de la création du DataFrame, nous pouvons attribuer des valeurs à l'index en utilisant le paramètre *index* dans notre constructeur :

```
pd.DataFrame({'Bob': ['I liked it.', 'It was awful.'],
              'Sue': ['Pretty good.', 'Bland.']},
             index=['Product A', 'Product B'])
```

Series

Un objet *Series* est une séquence de valeurs de données. Si un DataFrame est équivalent à une table dans une BDD relationnelle, un objet Series est une colonne de la table. Et en fait, vous pouvez en créer une avec l'instruction suivante :

```
pd.Series([1, 2, 3, 4, 5])
```

```
0 1

1 2

2 3

3 4

4 5

dtype: int64
```

Un objet Series est, par essence, une seule colonne d'un *DataFrame*. Vous pouvez donc attribuer des étiquettes aux lignes à la série de la même manière qu'auparavant, en utilisant un paramètre d'index. Cependant, une série n'a pas de nom de colonne :

```
pd.Series([30, 35, 40], index=['2015 Sales', '2016 Sales', '2017 Sales'], name=
 'Product A')
```

```
2015 Sales 30

2016 Sales 35

2017 Sales 40

Name: Product A, dtype: int64
```

Les Series et DataFrame sont intimement liés. Il est utile de considérer un DataFrame comme n'étant en réalité qu'un ensemble de séries collées ensemble où chaque colonne est une série.

Lecture de DataFrame à partir de fichier

IV

La plupart du temps, nous ne créerons pas nos propres DataFrame. Au lieu, nous travaillerons avec des données qui existent déjà sous forme de fichier. Les données peuvent être stockées sous différents formats. Le plus basique d'entre eux est de loin le modeste fichier CSV. Lorsque vous ouvrez un fichier CSV, vous obtenez quelque chose qui ressemble à ceci :

```
Product A,Product B,Product C,
30,21,9,
35,34,1,
41,11,11
```

Un fichier CSV est donc un tableau de valeurs séparées par des virgules. D'où le nom : « Comma-Separated Values » (Valeurs séparées par des virgules), ou CSV. Nous utiliserons la fonction pd.read_csv() pour lire les données du Dataset World University Rankings 2023[2*] dans un DataFrame :

```
rankings=pd.read_csv("World University Rankings 2023.csv")
```

Nous pouvons utiliser l'attribut *shape* pour vérifier la taille et la structure du DataFrame lue à partir du fichier World University Rankings 2023.csv:

```
rankings.shape
```

```
(2341, 13)
```

Ainsi, notre nouveau DataFrame contient 2341 enregistrements répartis sur 13 colonnes différentes. Nous pouvons examiner le contenu du DataFrame résultant à l'aide de la commande head(), qui récupère les cinq premières lignes :

```
rankings.head()
```

	University Rank	Name of University	Location	No of student	No of student per staff	International Student	Female:Male Ratio	OverAll Score	Teaching Score	Research Score	Citations Score	Industry Income Score	International Outlook Score
0	1	University of Oxford	United Kingdom	20,965	10.6	42%	48 : 52	96.4	92.3	99.7	99.0	74.9	96.2
1	2	Harvard University	United States	21,887	9.6	25%	50 : 50	95.2	94.8	99.0	99.3	49.5	80.5
2	3	University of Cambridge	United Kingdom	20,185	11.3	39%	47 : 53	94.8	90.9	99.5	97.0	54.2	95.8
3	3	Stanford University	United States	16,164	7.1	24%	46 : 54	94.8	94.2	96.7	99.8	65.0	79.8
4	5	Massachusetts Institute of Technology	United States	11,415	8.2	33%	40 : 60	94.2	90.7	93.6	99.8	90.9	89.3

La fonction *pd.read_csv()* est définie avec plus de 30 paramètres facultatifs que vous pouvez spécifier. Par exemple, vous pouvez voir dans cet ensemble de données que le fichier CSV possède un index intégré, que pandas ne récupère pas automatiquement. Pour que pandas utilise cette colonne comme l'index (au lieu d'en créer un index séquentiel), nous pouvons spécifier la valeur du paramètres *index_col*. Dans cet exemple nous avons choisi la deuxième colonne *Name of University* comme index :

```
1 rankings=pd.read_csv("World University Rankings 2023.csv",index_col=1)
2 rankings
```

Name of University	University Rank	Location	No of student	No of student per staff	International Student	Female:Male Ratio	OverAll Score	Teaching Score	Research Score	Citations Score	Industry Income Score	International Outlook Score
University of Oxford	1	United Kingdom	20,965	10.6	42%	48 : 52	96.4	92.3	99.7	99.0	74.9	96.2
Harvard University	2	United States	21,887	9.6	25%	50 : 50	95.2	94.8	99.0	99.3	49.5	80.5
University of Cambridge	3	United Kingdom	20,185	11.3	39%	47 : 53	94.8	90.9	99.5	97.0	54.2	95.8
Stanford University	3	United States	16,164	7.1	24%	46 : 54	94.8	94.2	96.7	99.8	65.0	79.8
Massachusetts Institute of Technology	5	United States	11,415	8.2	33%	40 : 60	94.2	90.7	93.6	99.8	90.9	89.3
...	...	...	...	...	...	...	...	...	...	...	...	...
University of the West of Scotland	-	NaN	NaN	NaN	NaN	NaN	34.0–39.2	24.1	15.5	61.5	37.9	76.8
University of Windsor	-	NaN	NaN	NaN	NaN	NaN	34.0–39.2	35.1	29.4	34.5	44.2	88.7

Indexation et accès aux valeurs

V

La sélection de valeurs spécifiques d'un DataFrame ou d'une série pandas sur lesquelles travailler est une étape importante dans toutes les opérations de données que vous exécuterez. L'une des premières choses que vous devez apprendre en travaillant avec des données en Python est de savoir comment sélectionner les données et les entrées qui vous intéressent.

Considérez ce DataFrame nommé *rankings*:

	University Rank	Name of University	Location	No of student	No of student per staff	International Student	Female:Male Ratio	OverAll Score	Teaching Score	Research Score	Citations Score	Industry Income Score	International Outlook Score
0	1	University of Oxford	United Kingdom	20,965	10.6	42%	48 : 52	96.4	92.3	99.7	99.0	74.9	96.2
1	2	Harvard University	United States	21,887	9.6	25%	50 : 50	95.2	94.8	99.0	99.3	49.5	80.5
2	3	University of Cambridge	United Kingdom	20,185	11.3	39%	47 : 53	94.8	90.9	99.5	97.0	54.2	95.8
3	3	Stanford University	United States	16,164	7.1	24%	46 : 54	94.8	94.2	96.7	99.8	65.0	79.8
4	5	Massachusetts Institute of Technology	United States	11,415	8.2	33%	40 : 60	94.2	90.7	93.6	99.8	90.9	89.3
...	...	...	...	...	...	...	...	...	...	...	...	...	...
2336	-	University of the West of Scotland	NaN	NaN	NaN	NaN	NaN	34.0–39.2	24.1	15.5	61.5	37.9	76.8
2337	-	University of Windsor	NaN	NaN	NaN	NaN	NaN	34.0–39.2	35.1	29.4	34.5	44.2	88.7
2338	-	University of Wolverhampton	NaN	NaN	NaN	NaN	NaN	34.0–39.2	18.2	14.3	68.8	37.3	72.0
2339	-	University of Wuppertal	NaN	NaN	NaN	NaN	NaN	34.0–39.2	26.4	26.7	52.8	52.1	47.6

En Python comme dans d'autres langages orientés objet, nous pouvons accéder à la propriété d'un objet. Par exemple, un objet *book* peut avoir une propriété *title*, à laquelle nous pouvons accéder en appelant *book.title*. Les colonnes d'un DataFrame pandas sont accessibles de la même manière qu'un attribut.

Ainsi, pour accéder à la propriété *Location* du DataFrame *rankings*, nous pouvons utiliser :

```
rankings.Location
```

```
0       United Kingdom
1        United States
2       United Kingdom
3        United States
4        United States
            ...
2336               NaN
2337               NaN
2338               NaN
2339               NaN
2340               NaN
```

```
Name: Location, Length: 2341, dtype: object
```

Si nous disposons d'un dictionnaire Python, nous pouvons accéder à ses valeurs à l'aide de l'opérateur d'indexation ([]). Nous pouvons faire la même chose avec les colonnes d'un DataFrame :

```
rankings['Location']
```

```
0       United Kingdom
1        United States
2       United Kingdom
3        United States
4        United States
             ...
2336               NaN
2337               NaN
2338               NaN
2339               NaN
2340               NaN
Name: Location, Length: 2341, dtype: object
```

L'opérateur d'indexation [] a l'avantage de pouvoir gérer les noms des colonnes contenant des caractères réservés (par exemple, si nous avions une colonne *Citations Score*, *rankings.Citations Score* ne fonctionnerait pas).

Un objet DataFrame peut être considéré comme un dictionnaire sophistiqué. Il n'est donc pas surprenant que, pour accéder à une seule valeur spécifique, nous n'ayons besoin que d'utiliser à nouveau l'opérateur d'indexation [] :

```
rankings['Location'][0]
```

```
'United Kingdom'
```

L'opérateur d'indexation et la sélection d'attributs sont intéressants car ils fonctionnent exactement comme dans le reste de l'écosystème Python. Cela les rend faciles à prendre en main et à utiliser. Cependant, pandas a ses propres opérateurs d'accès, loc et iloc. Pour les opérations plus avancées, ce sont celles-là que vous êtes censé utiliser.

L'indexation Pandas fonctionne selon l'un des deux paradigmes suivants. La première est la sélection basée sur un index : sélectionner des données en fonction de leurs positions numérique. iloc suit ce paradigme.

Pour sélectionner la première ligne de données dans un DataFrame, nous pouvons utiliser ce qui suit :

```
rankings.iloc[0]
```

```
University Rank                             1
Name of University            University of Oxford
Location                             United Kingdom
No of student                            20,965
No of student per staff                    10.6
International Student                       42%
Female:Male Ratio                        48 : 52
OverAll Score                              96.4
Teaching Score                             92.3
Research Score                             99.7
Citations Score                            99.0
Industry Income Score                      74.9
International Outlook Score                96.2
Name: 0, dtype: object
```

À lui seul, l'opérateur « : » , qui vient également de Python natif, signifie « tout ». Toutefois, lorsqu'il est combiné avec d'autres sélecteurs, il peut être utilisé pour indiquer une plage de valeurs. Par exemple, pour sélectionner la colonne *Name of University* (colonne 1) uniquement dans la première, la deuxième et la troisième ligne, nous ferions :

```
rankings.iloc[0:3, 1]
```

```
0       University of Oxford
1         Harvard University
2    University of Cambridge
Name: Name of University, dtype: object
```

Ou, pour sélectionner uniquement la deuxième et troisième entrée de la même colonne:

```
rankings.iloc[1:3, 1]
```

```
1         Harvard University
2    University of Cambridge
Name: Name of University, dtype: object
```

Il est également possible de passer une liste des numéros des lignes désirées:

```
rankings.iloc[[0, 1, 2], 1]
```

```
0       University of Oxford
1         Harvard University
2    University of Cambridge
Name: Name of University, dtype: object
```

Enfin, il faut savoir que les nombres négatifs peuvent être utilisés en sélection. Un index -K dans un DataFrame de N lignes désigne l'élément à la position N-K. Ainsi, par exemple, voici les cinq derniers éléments de l'ensemble de données.

```
rankings.iloc[-5:]
```

	University Rank	Name of University	Location	No of student	No of student per staff	International Student	Female:Male Ratio	OverAll Score	Teaching Score	Research Score	Citations Score	Industry Income Score	International Outlook Score
2336	-	University of the West of Scotland	NaN	NaN	NaN	NaN	NaN	34.0–39.2	24.1	15.5	61.5	37.9	76.8
2337	-	University of Windsor	NaN	NaN	NaN	NaN	NaN	34.0–39.2	35.1	29.4	34.5	44.2	88.7
2338	-	University of Wolverhampton	NaN	NaN	NaN	NaN	NaN	34.0–39.2	18.2	14.3	68.8	37.3	72.0
2339	-	University of Wuppertal	NaN	NaN	NaN	NaN	NaN	34.0–39.2	26.4	26.7	52.8	52.1	47.6
2340	-	Xi'an Jiaotong-Liverpool University	NaN	NaN	NaN	NaN	NaN	34.0–39.2	17.8	14.8	68.2	38.2	72.4

Selection par label

VI

Le deuxième option permettant la sélection est celle proposée par l'opérateur *loc* : la sélection basée sur les étiquettes. Avec cette méthode, c'est la valeur de l'index des données qui compte et non sa position.

Par exemple, pour obtenir la première entrée dans le DataFrame *rankings*, nous procéderions désormais comme suit :

```
rankings.loc[0, 'University Rank']
```

```
'1'
```

Lorsque nous utilisons *iloc*, nous traitons l'ensemble de données comme une grande matrice (une liste de listes), dans laquelle nous devons indexer par position. loc, en revanche, utilise les informations contenues dans les indices pour faire son travail. Étant donné que votre ensemble de données contient généralement des indices significatifs, il est généralement plus facile de faire les choses en utilisant *loc*. Par exemple, voici une opération beaucoup plus simple avec *loc* :

```
rankings.loc[:, ['University Rank', 'OverAll Score', 'Teaching Score', 'Research Score']]
```

	University Rank	OverAll Score	Teaching Score	Research Score
0	1	96.4	92.3	99.7
1	2	95.2	94.8	99.0
2	3	94.8	90.9	99.5
3	3	94.8	94.2	96.7
4	5	94.2	90.7	93.6
...	...	...	...	...
2336	-	34.0–39.2	24.1	15.5
2337	-	34.0–39.2	35.1	29.4
2338	-	34.0–39.2	18.2	14.3
2339	-	34.0–39.2	26.4	26.7
2340	-	34.0–39.2	17.8	14.8

Fondamental : Diffrence entre iloc et loc

Lors du choix ou de la transition entre *loc* et *iloc*, il convient de garder à l'esprit les points suivants: les deux méthodes utilisent des schémas d'indexation légèrement différents.

iloc utilise le schéma d'indexation Python stdlib, où le premier élément de la plage est inclus et le dernier exclu. Donc 0:10 sélectionnera les entrées 0,...,9. Par contre, *loc* indexe *inclusivement*. Donc 0:10 sélectionnera les entrées 0,...,10.

Changement d'index

VII

La sélection basée sur les étiquettes tire sa puissance des étiquettes de l'index que nous pouvons manipuler l'index comme bon nous semble. La méthode *set_index*() peut être utilisée pour effectuer le travail. Voici ce qui se passe lorsque nous définissons le champ *"Name of University" comme index*.

Ceci est utile si vous pouvez proposer un index pour l'ensemble de données qui est meilleur que l'index actuel.

```
1 rankings.set_index("Name of University")
```

Name of University	University Rank	Location	No of student	No of student per staff	International Student	Female:Male Ratio	OverAll Score	Teaching Score	Research Score	Citations Score	Industry Income Score	International Outlook Score
University of Oxford	1	United Kingdom	20,965	10.6	42%	48 : 52	96.4	92.3	99.7	99.0	74.9	96.2
Harvard University	2	United States	21,887	9.6	25%	50 : 50	95.2	94.8	99.0	99.3	49.5	80.5
University of Cambridge	3	United Kingdom	20,185	11.3	39%	47 : 53	94.8	90.9	99.5	97.0	54.2	95.8
Stanford University	3	United States	16,164	7.1	24%	46 : 54	94.8	94.2	96.7	99.8	65.0	79.8
Massachusetts Institute of Technology	5	United States	11,415	8.2	33%	40 : 60	94.2	90.7	93.6	99.8	90.9	89.3
...	...	...	...	...	...	...	...	...	...	...	...	...
University of the West of Scotland	-	NaN	NaN	NaN	NaN	NaN	34.0–39.2	24.1	15.5	61.5	37.9	76.8
University of Windsor	-	NaN	NaN	NaN	NaN	NaN	34.0–39.2	35.1	29.4	34.5	44.2	88.7
University of Wolverhampton	-	NaN	NaN	NaN	NaN	NaN	34.0–39.2	18.2	14.3	68.8	37.3	72.0
University of Wuppertal	-	NaN	NaN	NaN	NaN	NaN	34.0–39.2	26.4	26.7	52.8	52.1	47.6
Xi'an Jiaotong-Liverpool University	-	NaN	NaN	NaN	NaN	NaN	34.0–39.2	17.8	14.8	68.2	38.2	72.4

Il est aussi possible d'utiliser plusieurs colonnes comme indexes (index-multiple). Par exemple, le code suivant crée un index composé de la colonne *University Rank* et *Location* .

```
1 rankings.set_index(["University Rank" , "Location" ])
```

Name of University	Location	University Rank	No of student	No of student per staff	International Student	Female:Male Ratio	OverAll Score	Te
University of Oxford	United Kingdom	1	20,965	10.6	42%	48 : 52	96.4	
Harvard University	United States	2	21,887	9.6	25%	50 : 50	95.2	
University of Cambridge	United Kingdom	3	20,185	11.3	39%	47 : 53	94.8	
Stanford University	United States	3	16,164	7.1	24%	46 : 54	94.8	

A noter que set_index ne modifie pas l'objet DataFrame lui même mais renvoie une copie modifiée de celui-ci. Afin de modifier l'index de l'objet DataFrame, il faudra réaffecter (*rankings=rankings.set_index(["University Rank" , "Location"])*) ou utiliser le paramètre optionnel *inplace* (rankings.set_index(["University Rank" , "Location"] , *inplace=True*)).

Pour inverser l'effet de set_index (supprimer un ou plusieurs colonnes de l'index), la méthode reset_index() est utilisée :

```
rankings=rankings.set_index(["University Rank" , "Location" ])
rankings.reset_index("Location") # supprime la colonne Location de l'index
```

Name of University	Location	University Rank	No of student	No of student per staff	International Student	Female:Male Ratio	OverAll Score	T
University of Oxford	United Kingdom	1	20,965	10.6	42%	48 : 52	96.4	
Harvard University	United States	2	21,887	9.6	25%	50 : 50	95.2	
University of Cambridge	United Kingdom	3	20,185	11.3	39%	47 : 53	94.8	
Stanford University	United States	3	16,164	7.1	24%	46 : 54	94.8	

De meme, reset_index() ne modifie pas l'objet DataFrame :

```
rankings=rankings.reset_index(["University Rank" , "Location" ])
rankings.head()
```

	University Rank	Name of University	Location	No of student	No of student per staff	International Student	Female:Male Ratio	OverAll Score	Teaching Score	Research Score	Citations Score	Industry Income Score	International Outlook Score
0	1	University of Oxford	United Kingdom	20,965	10.6	42%	48 : 52	96.4	92.3	99.7	99.0	74.9	96.2
1	2	Harvard University	United States	21,887	9.6	25%	50 : 50	95.2	94.8	99.0	99.3	49.5	80.5
2	3	University of Cambridge	United Kingdom	20,185	11.3	39%	47 : 53	94.8	90.9	99.5	97.0	54.2	95.8
3	3	Stanford University	United States	16,164	7.1	24%	46 : 54	94.8	94.2	96.7	99.8	65.0	79.8
4	5	Massachusetts Institute of Technology	United States	11,415	8.2	33%	40 : 60	94.2	90.7	93.6	99.8	90.9	89.3

Sélection conditionnelle

VIII

Cependant, pour faire des choses intéressantes avec les données, nous devons souvent poser des questions basées sur les conditions. Par exemple, supposons que nous nous intéressions spécifiquement aux universités Anglaises.

```
rankings["Location"]=="United Kingdom"
```

```
0         True
1        False
2         True
3        False
4        False
         ...
2336     False
2337     False
2338     False
2339     False
2340     False
Name: Location, Length: 2341, dtype: bool
```

Cette opération a produit une série de booléens Vrai/Faux basés sur la valeur du champ Location de chaque enregistrement. Ce résultat peut ensuite être utilisé à l'intérieur de *loc* ou de l'opérateur [] pour sélectionner les données pertinentes :

```
rankings.loc[rankings["Location"]=="United Kingdom"]
```

	University Rank	Name of University	Location	No of student	No of student per staff	International Student	Female:Male Ratio	OverAll Score	Teaching Score	Research Score	Citations Score	Industry Income Score	International Outlook Score
0	1	University of Oxford	United Kingdom	20,965	10.6	42%	48 : 52	96.4	92.3	99.7	99.0	74.9	96.2
2	3	University of Cambridge	United Kingdom	20,185	11.3	39%	47 : 53	94.8	90.9	99.5	97.0	54.2	95.8
9	10	Imperial College London	United Kingdom	18,545	11.2	61%	40 : 60	90.4	82.8	90.8	98.3	59.8	97.5
21	22	UCL	United Kingdom	36,790	10.3	60%	59 : 41	85.7	74.5	85.4	97.9	44.5	96.7
28	29	University of Edinburgh	United Kingdom	32,845	11.8	47%	62 : 38	79.8	66.9	74.5	97.1	40.9	95.6
...	...	...	...	...	...	...	...	...	...	...	...	...	...
2192	Reporter	University of West London	United Kingdom	9,510	15.9	42%	60 : 40	NaN	NaN	NaN	NaN	NaN	NaN
2196	Reporter	University of Worcester	United Kingdom	7,400	16.8	13%	69 : 31	NaN	NaN	NaN	NaN	NaN	NaN
2199	Reporter	Wrexham Glyndwr University	United Kingdom	3,520	22.0	14%	61 : 39	NaN	NaN	NaN	NaN	NaN	NaN
2200	Reporter	Writtle University College	United Kingdom	655	7.7	10%	79 : 21	NaN	NaN	NaN	NaN	NaN	NaN
2205	Reporter	York St John University	United Kingdom	6,315	18.6	12%	65 : 35	NaN	NaN	NaN	NaN	NaN	NaN

Ce résultat contient 149 lignes. Le DataFrame original en comptait environ 2341. Cela signifie qu'environ 6 % des universités dans le classement sont situées en Angleterre.

Nous voulions également savoir lesquels des universités anglaises ont un score de recherche supérieur ou égale à 70 (Research Score). Ceci est faisable avec l'opérateur *&*.

```
rankings.loc[(rankings.Location=="United Kingdom") & (rankings["Research Score"]>=
70)]
```

	University Rank	Name of University	Location	No of student	No of student per staff	International Student	Female:Male Ratio	OverAll Score	Teaching Score	Research Score	Citations Score	Industry Income Score	International Outlook Score
0	1	University of Oxford	United Kingdom	20,965	10.6	42%	48 : 52	96.4	92.3	99.7	99.0	74.9	96.2
2	3	University of Cambridge	United Kingdom	20,185	11.3	39%	47 : 53	94.8	90.9	99.5	97.0	54.2	95.8
9	10	Imperial College London	United Kingdom	18,545	11.2	61%	40 : 60	90.4	82.8	90.8	98.3	59.8	97.5
21	22	UCL	United Kingdom	36,790	10.3	60%	59 : 41	85.7	74.5	85.4	97.9	44.5	96.7
28	29	University of Edinburgh	United Kingdom	32,845	11.8	47%	62 : 38	79.8	66.9	74.5	97.1	40.9	95.6
34	35	King's College London	United Kingdom	28,965	11.8	52%	63 : 37	77.1	58.0	72.9	98.2	45.6	96.1
36	37	London School of Economics and Political Science	United Kingdom	11,120	11.9	73%	55 : 45	76.5	59.2	74.3	95.1	37.8	92.8

Supposons que nous cherchons les universités situés en Algérie ou en Tunisie. Pour cela nous utilisons un pipe (|) pour combiné les conditions:

```
rankings.loc[(rankings["Location"]=="Algeria") | (rankings["Location"]=="Tunisia"
) ]
```

	University Rank	Name of University	Location	No of student	No of student per staff	International Student	Female:Male Ratio	OverAll Score	Teaching Score	Research Score	Citations Score	Industry Income Score	International Outlook Score
422	401–500	Ferhat Abbas Sétif University 1	Algeria	34,637	22.7	1%	63 : 37	42.1–44.9	18.2	19.8	94.7	36.9	40.0
1063	1001–1200	University of Manouba	Tunisia	16,315	12.1	1%	59 : 41	24.4–29.7	19.2	8.3	52.7	36.9	44.6
1159	1201–1500	Université 8 Mai 1945 Guelma	Algeria	17,530	20.1	1%	67 : 33	18.4–24.3	16.2	8.6	41.3	37.0	36.5
1196	1201–1500	University of Carthage	Tunisia	28,962	9.9	1%	67 : 33	18.4–24.3	19.0	9.8	19.1	38.3	44.2
1238	1201–1500	Université de Gabès	Tunisia	13,152	12.8	1%	75 : 25	18.4–24.3	18.1	9.3	25.7	36.9	42.1
1301	1201–1500	University of Monastir	Tunisia	19,203	9.6	2%	70 : 30	18.4–24.3	19.6	10.0	26.5	36.9	45.1
1326	1201–1500	Oran 1 University	Algeria	25,775	19.2	1%	66 : 34	18.4–24.3	30.4	8.3	22.2	37.6	39.5
1358	1201–1500	University of Sfax	Tunisia	29,118	12.9	2%	68 : 32	18.4–24.3	25.2	14.0	22.6	37.3	43.6
1373	1201–1500	University of Sousse	Tunisia	25,602	13.4	2%	68 : 32	18.4–24.3	23.4	9.4	24.1	36.9	40.1
1409	1201–1500	University of Tunis El Manar	Tunisia	26,825	11.1	2%	70 : 30	18.4–24.3	28.8	13.7	19.5	38.2	43.7
		University of											

Pandas est livré avec quelques sélecteurs conditionnels intégrés, dont deux que nous soulignerons ici.

Le premier est *isin*. Celui-ci vous permet de sélectionner des données dont la valeur 'est dans' une liste de valeurs. Par exemple, voici comment nous pouvons l'utiliser pour sélectionner les universités situés en Algérie ou en Tunisie :

```
rankings.loc[rankings.Location.isin(["Algeria","Tunisia"]) ]
```

Le second *isnull* (et son contre-part *notnull*). Ces méthodes permettent de sélectionner les valeurs nulles ou non nulles (NaN) dans une colonne. Par exemple, pour filtrer les universités qui ne possèdent pas de valeur pour la colonne *Location*:

```
1 rankings[rankings.Location.isnull()]
```

	University Rank	Name of University	Location	No of student	No of student per staff	International Student	Female:Male Ratio	OverAll Score	Teaching Score	Research Score	Citations Score	Industry Income Score	International Outlook Score
15	16	Tsinghua University	NaN	38,324	11.6	10%	NaN	88.2	90.1	97.4	88.0	100.0	40.3
16	17	Peking University	NaN	31,994	10.3	19%	NaN	88.1	92.5	96.7	80.4	91.8	65.0
18	19	National University of Singapore	NaN	32,337	19.8	25%	51 : 49	87.1	76.4	93.0	90.2	87.0	94.0
29	30	Technical University of Munich	NaN	33,960	40.6	36%	37 : 63	79.3	69.8	82.2	84.5	100.0	77.7
30	31	University of Hong Kong	NaN	18,087	18.2	43%	54 : 46	78.5	65.6	74.1	92.4	60.6	98.7
...	...	...	...	...	...	...	...	...	...	...	...	...	...
2336	-	University of the West of Scotland	NaN	NaN	NaN	NaN	NaN	34.0–39.2	24.1	15.5	61.5	37.9	76.8
2337	-	University of Windsor	NaN	NaN	NaN	NaN	NaN	34.0–39.2	35.1	29.4	34.5	44.2	88.7
2338	-	University of Wolverhampton	NaN	NaN	NaN	NaN	NaN	34.0–39.2	18.2	14.3	68.8	37.3	72.0
2339	-	University of Wuppertal	NaN	NaN	NaN	NaN	NaN	34.0–39.2	26.4	26.7	52.8	52.1	47.6
2340	-	Xi'an Jiaotong-Liverpool University	NaN	NaN	NaN	NaN	NaN	34.0–39.2	17.8	14.8	68.2	38.2	72.4

Affectation

IX

Il est possible de créer une nouvelle colonne en affectant la colonne à une valeur ou à une liste de valeurs :

```
rankings['col1']='valeur 1'
rankings.head()
```

	University Rank	Name of University	Location	No of student	No of student per staff	International Student	Female:Male Ratio	OverAll Score	Teaching Score	Research Score	Citations Score	Industry Income Score	International Outlook Score	col1
0	1	University of Oxford	United Kingdom	20,965	10.6	42%	48 : 52	96.4	92.3	99.7	99.0	74.9	96.2	valeur 1
1	2	Harvard University	United States	21,887	9.6	25%	50 : 50	95.2	94.8	99.0	99.3	49.5	80.5	valeur 1
2	3	University of Cambridge	United Kingdom	20,185	11.3	39%	47 : 53	94.8	90.9	99.5	97.0	54.2	95.8	valeur 1
3	3	Stanford University	United States	16,164	7.1	24%	46 : 54	94.8	94.2	96.7	99.8	65.0	79.8	valeur 1
4	5	Massachusetts Institute of Technology	United States	11,415	8.2	33%	40 : 60	94.2	90.7	93.6	99.8	90.9	89.3	valeur 1

```
rankings['University Rank']=range(len(rankings))
rankings
```

	University Rank	Name of University	Location	No of student	No of student per staff	International Student	Female:Male Ratio	OverAll Score	Teaching Score	Research Score	Citations Score	Industry Income Score	International Outlook Score
0	0	University of Oxford	United Kingdom	20,965	10.6	42%	48 : 52	96.4	92.3	99.7	99.0	74.9	96.2
1	1	Harvard University	United States	21,887	9.6	25%	50 : 50	95.2	94.8	99.0	99.3	49.5	80.5
2	2	University of Cambridge	United Kingdom	20,185	11.3	39%	47 : 53	94.8	90.9	99.5	97.0	54.2	95.8
3	3	Stanford University	United States	16,164	7.1	24%	46 : 54	94.8	94.2	96.7	99.8	65.0	79.8
4	4	Massachusetts Institute of Technology	United States	11,415	8.2	33%	40 : 60	94.2	90.7	93.6	99.8	90.9	89.3
...	...	...	...	...	...	...	...	...	...	...	...	...	...
2336	2336	University of the West of Scotland	NaN	NaN	NaN	NaN	NaN	34.0–39.2	24.1	15.5	61.5	37.9	76.8
2337	2337	University of Windsor	NaN	NaN	NaN	NaN	NaN	34.0–39.2	35.1	29.4	34.5	44.2	88.7
2338	2338	University of Wolverhampton	NaN	NaN	NaN	NaN	NaN	34.0–39.2	18.2	14.3	68.8	37.3	72.0
2339	2339	University of Wuppertal	NaN	NaN	NaN	NaN	NaN	34.0–39.2	26.4	26.7	52.8	52.1	47.6
2340	2340	Xi'an Jiaotong-Liverpool University	NaN	NaN	NaN	NaN	NaN	34.0–39.2	17.8	14.8	68.2	38.2	72.4

Références

[01] pandas 2024, https://pandas.pydata.org/

[02] World University Rankings 2023 2023, https://www.kaggle.com/datasets/alitaqi000/world-university-rankings-2023