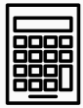


# **Python pour la Data Science**



Section 1 : Remise à niveau rapide sur Python



Section 2: Data Science avec Python



Section 3: Structures de données Pandas



Section 4: Nettoyage de données

# **STRUCTURES DE DONNÉES PANDAS**



Remise à niveau rapide  
sur Python

Data Science avec  
Python

Structures de données  
Pandas

Structures de données Pandas

1

2

C'est quoi un DataFrame de  
Pandas ?

3

C'est quoi Jupyter ?

4

Installation d'Anaconda pour le  
système d'exploitation Windows

5

Installation d'Anaconda  
pour Mac OS

6

Installation d'Anaconda pour le  
système d'exploitation Ubuntu

7

Comment implémenter Python  
dans Jupyter

8

Gestion des répertoires  
dans Jupyter Notebook

9

Entrées/sorties

10

Travailler avec différents  
types de données

11

Variables

12

Opérateurs arithmétiques



# Structures de données dans Pandas

- Pandas dispose de deux structures de données principales;
  - DataFrame, qui est bidimensionnelle
  - Series, qui est unidimensionnelle

	Nom	calories	Protéine	Vitamines	Cote
0	100% Son	70	4	25	68.402973
1	100 % Son Naturel	120	3	0	33.983679
2	Tout-Son	70	4	25	59.425505
3	Tout-Son avec fibres supplémentaires	50	4	25	93.704912
4	Délice d'amandes	110	2	25	34.384843
5	Cheerios pomme-cannelle	110	2	25	29.509541
6	Pomme Jacks	110	2	25	33.174094
7	Basique 4	130	3	25	37.038562
8	Boulettes de Son	90	2	25	49.120253
9	Flocons de Son	90	3	25	53.313813

*DataFrame*

```
0    70
1   120
2    70
3    50
4   110
5   110
6   110
7   130
8    90
9    90
Name: calories, dtype: int64
```

*Series*



Remise à niveau rapide  
sur Python

Data Science avec  
Python

Structures de données  
Pandas

Structures de données Pandas 1

2 C'est quoi un DataFrame de  
Pandas ?

C'est quoi la série Pandas ? 3

4 Installation d'Anaconda pour le  
système d'exploitation Windows

5 Installation d'Anaconda  
pour Mac OS

6 Installation d'Anaconda pour le  
système d'exploitation Ubuntu

7 Comment implémenter Python  
dans Jupyter

8 Gestion des répertoires  
dans Jupyter Notebook

9 Entrées/sorties

10 Travailler avec différents  
types de données

11 Variables

12 Opérateurs arithmétiques



# C'est quoi un DataFrame de Pandas ?

- Pandas fournit une structure de données bidimensionnelle appelée DataFrame.
- Une ligne est représentée par des étiquettes de ligne, également appelées index, qui peuvent être numériques ou des chaînes de caractères.
- Une colonne est représentée par des étiquettes de colonne qui peuvent être numériques ou des chaînes de caractères.
- Le DataFrame suivant contient 10 lignes (0-9) et 5 colonnes (nom, calories, protéines, vitamines, cote).

	Nom	calories	Protéine	Vitamines	Cote
0	100% Son	70	4	25	68.402973
1	100 % Son Naturel	120	3	0	33.983679
2	Tout-Son	70	4	25	59.425505
3	Tout-Son avec fibres supplémentaires	50	4	25	93.704912
4	Délice d'amandes	110	2	25	34.384843
5	Cheerios pomme-cannelle	110	2	25	29.509541
6	Pomme Jacks	110	2	25	33.174094
7	Basique 4	130	3	25	37.038562
8	Boulettes de Son	90	2	25	49.120253
9	Flocons de Son	90	3	25	53.313813

index

étiquettes de colonne



Remise à niveau rapide  
sur Python

Data Science avec  
Python

Structures de données  
Pandas

Structures de données Pandas

1

2 C'est quoi un DataFrame de  
Pandas ?

C'est quoi la série Pandas ?

3

4 Création d'un DataFrame à  
l'aide de listes

Installation d'Anaconda  
pour Mac OS

5

6 Installation d'Anaconda pour le  
système d'exploitation Ubuntu

Comment implémenter Python  
dans Jupyter

7

8 Gestion des répertoires  
dans Jupyter Notebook

Entrées/sorties

9

10 Travailler avec différents  
types de données

Variables

11

12 Opérateurs arithmétiques





## C'est quoi la série Pandas ?

- Une série dans Pandas est une structure de données unidimensionnelle.
- Elle se compose d'une seule ligne ou colonne.
- La série suivante contient 10 lignes (0-9) et une colonne appelée calories..

0	70
1	120
2	70
3	50
4	110
5	110
6	110
7	130
8	90
9	90
Nom : calories, dtype: int64	

index

valeurs de la colonne

nom de la colonne

type de données de la colonne



## DataFrame et Série

- Un DataFrame Pandas est simplement une collection d'une ou plusieurs séries.
- Les séries de l'exemple précédent ont été extraites du DataFrame.

	Nom	calories	Protéine	Vitamines	Cote
0	100% Son	70	4	25	68.402973
1	100 % Son Nature	120	3	0	33.983679
2	Tout-Son	70	4	25	59.425505
3	Tout-Son avec fibres supplémentaires	50	4	25	93.704912
4	Délice d'amandes	110	2	25	34.384843
5	Cheerios pomme-cannelle	110	2	25	29.509541
6	Pomme Jacks	110	2	25	33.174094
7	Basique 4	130	3	25	37.038562
8	Boulettes de Son	90	2	25	49.120253
9	Flocons de Son	90	3	25	53.313813

DataFrame

0	70
1	120
2	70
3	50
4	110
5	110
6	110
7	130
8	90
9	90
Nom : calories, dtype: int64	

Serie

Identique



Remise à niveau rapide  
sur Python

Data Science avec  
Python

Structures de données  
Pandas

Structures de données Pandas

1

C'est quoi la série Pandas ?

2

C'est quoi un DataFrame de  
Pandas ?

3

Création d'un DataFrame à  
l'aide d'un dictionnaire

4

Création d'un DataFrame à  
l'aide de listes

5

Comment implémenter Python  
dans Jupyter

6

Installation d'Anaconda pour le  
système d'exploitation Ubuntu

7

8

Gestion des répertoires  
dans Jupyter Notebook

Entrées/sorties

9

10

Travailler avec différents  
types de données

Variables

11

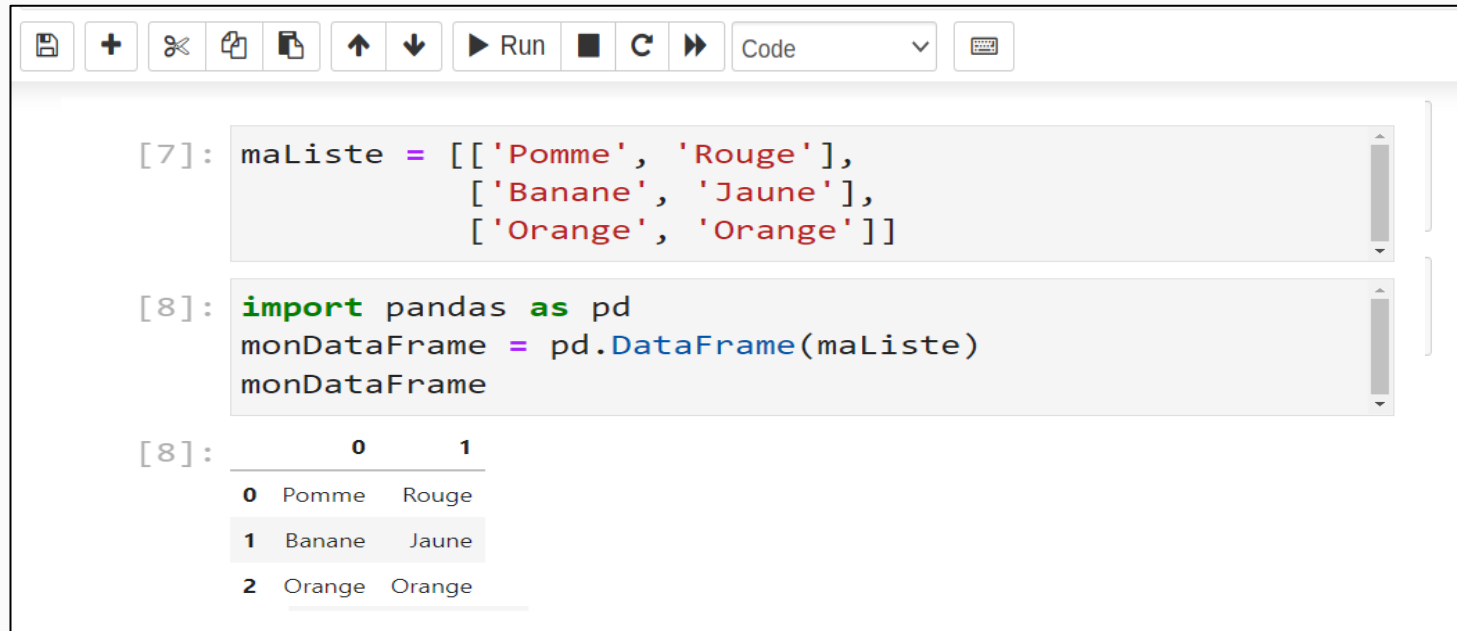
12

Opérateurs arithmétiques



## Création d'un DataFrame à l'aide de listes(1/3)

- Nous pouvons créer un DataFrame en utilisant des listes.
- Nous passons la liste comme argument à la fonction `pandas.DataFrame()` qui nous renvoie un DataFrame.
- Pandas attribue automatiquement des étiquettes de ligne numériques à chaque ligne du DataFrame.
- Comme nous n'avons pas fourni d'étiquettes de colonne, Pandas attribue automatiquement des étiquettes de colonne numériques à chaque colonne.



```
[7]: maListe = [['Pomme', 'Rouge'],  
               ['Banane', 'Jaune'],  
               ['Orange', 'Orange']]
```

```
[8]: import pandas as pd  
monDataFrame = pd.DataFrame(maListe)  
monDataFrame
```

```
[8]:
```

	0	1
0	Pomme	Rouge
1	Banane	Jaune
2	Orange	Orange



## Création d'un DataFrame à l'aide de listes(2/3)

- Créons un autre DataFrame en utilisant la même liste, mais cette fois avec des étiquettes de colonne personnalisées.
- `Pandas.DataFrame()` prend un autre argument optionnel appelé 'columns' qui prend une liste de noms de colonnes personnalisés à définir comme étiquettes de colonnes.

```
[9]: maListe = [['Pomme', 'Rouge'],  
               ['Banane', 'Jaune'],  
               ['Orange', 'Orange']]
```

```
[10]: import pandas as pd  
monDataFrame = pd.DataFrame(maListe, columns=['Fruit', 'Couleur'])  
monDataFrame
```

```
[10]:
```

	Fruit	Couleur
0	Pomme	Rouge
1	Banane	Jaune
2	Orange	Orange



## Création d'un DataFrame à l'aide de listes(3/3)

- Comme nous savons qu'un NumPy Array est similaire à une liste Python avec des fonctionnalités supplémentaires, nous pouvons également convertir un NumPy Array en DataFrame en utilisant la même méthode.

```
[11]: import numpy as np
      maListe = np.array([[0, 1],
                        [2, 3],
                        [4, 5]])
```

```
[12]: import pandas as pd
      monDataFrame = pd.DataFrame(maListe, columns=['pair', 'impair'])
      monDataFrame
```

```
[12]:
```

	pair	impair
0	0	1
1	2	3
2	4	5



Remise à niveau rapide  
sur Python

Data Science avec  
Python

Structures de données  
Pandas

Structures de données Pandas

1

C'est quoi la série Pandas ?

2

C'est quoi un DataFrame de  
Pandas ?

3

Création d'un DataFrame à  
l'aide d'un dictionnaire

4

Création d'un DataFrame à  
l'aide de listes

5

Chargement d'un fichier csv en  
tant que DataFrame

6

Comment implémenter Python  
dans Jupyter

7

Gestion des répertoires  
dans Jupyter Notebook

8

Entrées/sorties

9

Travailler avec différents  
types de données

10

Variables

11

Opérateurs arithmétiques

12



## Création d'un DataFrame à l'aide d'un dictionnaire

- Nous pouvons également passer un dictionnaire à la fonction `pandas.DataFrame()` pour créer un DataFrame.
- Chaque clé du dictionnaire doit être associée à une liste d'une ou plusieurs valeurs.
- Les clés du dictionnaire deviennent des étiquettes de colonne.
- Pandas attribue automatiquement des étiquettes de ligne numériques à chaque ligne du DataFrame.

The screenshot shows a Jupyter Notebook interface with the title "DataFrames". The code cells show the following steps:

```
[16]: import pandas as pd
import numpy as np

[17]: monDictionnaire = {'Fruit': ['Pomme', 'Banane', 'Orange'], 'Couleur': ['Rouge', 'Jaune', 'Orange']}

[18]: monDataFrame = pd.DataFrame(monDictionnaire)
monDataFrame
```

The output of the last cell is a DataFrame with two columns, "Fruit" and "Couleur", and three rows of data:

	Fruit	Couleur
0	Pomme	Rouge
1	Banane	Jaune
2	Orange	Orange





Remise à niveau rapide  
sur Python

Data Science avec  
Python

Structures de données  
Pandas

Structures de données Pandas

1

C'est quoi la série Pandas ?

2

C'est quoi un DataFrame de  
Pandas ?

3

Création d'un DataFrame à  
l'aide d'un dictionnaire

4

Création d'un DataFrame à  
l'aide de listes

5

Chargement d'un fichier csv en  
tant que DataFrame

6

Comment implémenter Python  
dans Jupyter

7

Gestion des répertoires  
dans Jupyter Notebook

8

Entrées/sorties

9

Travailler avec différents  
types de données

10

Variables

11

Opérateurs arithmétiques

12



# Chargement d'un fichier csv en tant que DataFrame

- Nous pouvons également charger un fichier csv (comma separated values) en tant que DataFrame dans Pandas à l'aide de la fonction `pandas.read_csv()`.
- Chaque valeur de la première ligne du fichier csv devient une étiquette de colonne.
- Pandas attribue automatiquement des étiquettes de ligne numériques à chaque ligne du DataFrame.

```
[19]: import pandas as pd
      df = pd.read_csv("céréale.csv")
      df
```

```
[19]:
```

	Nom	Calorie	Proteine	Vitamines	Cote
0	100% Son Naturel	70	4	25	68.402973
1	Tout-Son	120	3	0	33.983679
2	Tout-Son avec fibres supplémentaires	70	4	25	59.425505
3	Delice d'amandes	50	4	25	93.704912
4	Cheerios pomme-cannelle	110	2	25	34.384843
5	Pomme Jacks	110	2	25	29.509541
6	Basique 4	110	2	25	33.174094
7	Boulettes de Son	130	3	25	37.038562
8	Flocons de Son	90	2	25	49.120253



Remise à niveau rapide  
sur Python

Data Science avec  
Python

Structures de données  
Pandas

Structures de données Pandas

1

C'est quoi la série Pandas ?

2

C'est quoi un DataFrame de  
Pandas ?

3

Création d'un DataFrame à  
l'aide d'un dictionnaire

4

Création d'un DataFrame à  
l'aide de listes

5

Chargement d'un fichier csv en  
tant que DataFrame

6

Modification de la colonne d'index

7

Gestion des répertoires  
dans Jupyter Notebook

8

Entrées/sorties

9

Travailler avec différents  
types de données

10

Variables

11

Opérateurs arithmétiques

12



## Modification de la colonne d'index

- Nous pouvons définir l'une des colonnes existantes comme la nouvelle colonne d'index du DataFrame en utilisant la fonction `.set_index()`

```
In [13]: df
```

```
Out[13]:
```

	Nom	Calorie	Proteine	Vitamines	Cote
0	100% Son Naturel	70	4	25	68.402973
1	Tout-Son	120	3	0	33.983679
2	Tout-Son avec fibres supplémentaires	70	4	25	59.425505
3	Delice d'amandes	50	4	25	93.704912
4	Cheerios pomme-cannelle	110	2	25	34.384843

```
df.set_index('Nom', inplace = True)
```

```
df
```

	Calorie	Proteine	Vitamines	Cote
Nom				
100% Son Naturel	70	4	25	68.402973
Tout-Son	120	3	0	33.983679
Tout-Son avec fibres supplémentaires	70	4	25	59.425505
Delice d'amandes	50	4	25	93.704912
Cheerios pomme-cannelle	110	2	25	34.384843



Remise à niveau rapide  
sur Python

Data Science avec  
Python

Structures de données  
Pandas

Structures de données Pandas

1

C'est quoi la série Pandas ?

2

C'est quoi un DataFrame de  
Pandas ?

3

Création d'un DataFrame à  
l'aide d'un dictionnaire

4

Création d'un DataFrame à  
l'aide de listes

5

Chargement d'un fichier csv en  
tant que DataFrame

6

Modification de la colonne d'index

7

8

inplace

Entrées/sorties

9

10

Travailler avec différents  
types de données

Variables

11

12

Opérateurs arithmétiques



## Inplace (1/2)

- N'oubliez pas que la plupart des fonctions de Pandas ne modifient pas le DataFrame original.
- Dans la section précédente, nous avons modifié la colonne d'index de notre DataFrame. Si nous imprimons à nouveau notre DataFrame, nous constatons que le DataFrame d'origine est inchangé.

```
[24]: df
```

```
[24]:
```

	Nom	Calorie	Proteine	Vitamines	Cote
0	100% Son Naturel	70	4	25	68.402973
1	Tout-Son	120	3	0	33.983679
2	Tout-Son avec fibres supplementaires	70	4	25	59.425505
3	Delice d'amandes	50	4	25	93.704912
4	Cheerios pomme-cannelle	110	2	25	34.384843



## Inplace (2/2)

- Nous pouvons utiliser l'argument inplace pour apporter des modifications au DataFrame d'origine.
- Dans l'exemple suivant, nous utilisons la fonction `.set_index()` pour modifier l'index de notre DataFrame et définir `inplace = True`.
- Comme le montre la figure, notre DataFrame original a été modifié.

```
[20]: df.set_index('Nom', inplace = True)
```

```
[21]: df
```

```
[21]:
```

	Calorie	Proteine	Vitamines	Cote
<b>Nom</b>				
<b>100% Son Naturel</b>	70	4	25	68.402973
<b>Tout-Son</b>	120	3	0	33.983679
<b>Tout-Son avec fibres supplementaires</b>	70	4	25	59.425505
<b>Delice d'amandes</b>	50	4	25	93.704912
<b>Cheerios pomme-cannelle</b>	110	2	25	34.384843
<b>Pomme Jacks</b>	110	2	25	29.509541
<b>Basique 4</b>	110	2	25	33.174094
<b>Boulettes de Son</b>	130	3	25	37.038562
<b>Flocons de Son</b>	90	2	25	49.120253



Remise à niveau rapide  
sur Python

Data Science avec  
Python

Structures de données  
Pandas

Structures de données Pandas	1	
	2	C'est quoi un DataFrame de Pandas ?
C'est quoi la série Pandas ?	3	
	4	Création d'un DataFrame à l'aide de listes
Création d'un DataFrame à l'aide d'un dictionnaire	5	
	6	Chargement d'un fichier csv en tant que DataFrame
Modification de la colonne d'index	7	
	8	inplace
Examen des données	9	
	10	Travailler avec différents types de données
Variables	11	
	12	Opérateurs arithmétiques





## Examen des données (1/2)

### head()

- La fonction head() nous donne par défaut les 5 **premières** lignes du DataFrame/Series.
- Pour obtenir plus de lignes, nous pouvons passer le nombre souhaité comme argument à la fonction head().

```
[25]: df.head(7)
```

```
[25]:
```

	Nom	Calorie	Proteine	Vitamines	Cote
0	100% Son Naturel	70	4	25	68.402973
1	Tout-Son	120	3	0	33.983679
2	Tout-Son avec fibres supplementaires	70	4	25	59.425505
3	Delice d'amandes	50	4	25	93.704912
4	Cheerios pomme-cannelle	110	2	25	34.384843
5	Pomme Jacks	110	2	25	29.509541
6	Basique 4	110	2	25	33.174094



## Examen des données (2/2)

### `tail()`

- La fonction `tail()` nous donne par défaut les 5 **dernières** lignes du DataFrame/Series.
- Pour obtenir plus de lignes, nous pouvons passer le nombre souhaité comme argument à la fonction `tail()`.

```
[27]: df.tail(7)
```

	Nom	Calorie	Proteine	Vitamines	Cote
2	Tout-Son avec fibres supplementaires	70	4	25	59.425505
3	Delice d'amandes	50	4	25	93.704912
4	Cheerios pomme-cannelle	110	2	25	34.384843
5	Pomme Jacks	110	2	25	29.509541
6	Basique 4	110	2	25	33.174094
7	Boulettes de Son	130	3	25	37.038562
8	Flocons de Son	90	2	25	49.120253



Remise à niveau rapide  
sur Python

Data Science avec  
Python

Structures de données  
Pandas

Structures de données Pandas

1

C'est quoi la série Pandas ?

2

C'est quoi un DataFrame de  
Pandas ?

3

Création d'un DataFrame à  
l'aide d'un dictionnaire

4

Création d'un DataFrame à  
l'aide de listes

5

Chargement d'un fichier csv en  
tant que DataFrame

6

Modification de la colonne d'index

7

inplace

8

Examen des données

9

Obtenir un résumé  
statistique du DataFrame

10

Variables

11

Opérateurs arithmétiques

12



## Résumé statistique

- Nous pouvons utiliser la fonction `describe()` pour obtenir un résumé statistique rapide de chaque colonne du DataFrame.

```
[28]: df.describe()
```

```
[28]:
```

	Calorie	Proteine	Vitamines	Cote
<b>count</b>	9.000000	9.000000	9.000000	9.000000
<b>mean</b>	95.555556	2.888889	22.222222	48.749374
<b>std</b>	26.977357	0.927961	8.333333	21.493170
<b>min</b>	50.000000	2.000000	0.000000	29.509541
<b>25%</b>	70.000000	2.000000	25.000000	33.983679
<b>50%</b>	110.000000	3.000000	25.000000	37.038562
<b>75%</b>	110.000000	4.000000	25.000000	59.425505
<b>max</b>	130.000000	4.000000	25.000000	93.704912



Remise à niveau rapide  
sur Python

Data Science avec  
Python

Structures de données  
Pandas

Structures de données Pandas

1

C'est quoi la série Pandas ?

2

C'est quoi un DataFrame de  
Pandas ?

3

Création d'un DataFrame à  
l'aide d'un dictionnaire

4

Création d'un DataFrame à  
l'aide de listes

5

Chargement d'un fichier csv en  
tant que DataFrame

6

Modification de la colonne d'index

7

8

inplace

Examen des données

9

10

Obtenir un résumé  
statistique du DataFrame

Découpage de rangées à l'aide  
de l'opérateur de crochets

11

12

Opérateurs arithmétiques



## Opérateur pour le découpage en rangs [ ](1/2)

- Nous pouvons utiliser l'opérateur de parenthèses ([ ]) pour découper les lignes du DataFrame.
- Nous passons un index de début (inclusif) et un index de fin (exclusif) à l'opérateur de crochets ([ ]) pour découper les lignes du DataFrame

```
[29]: df[1:4]
```

```
[29]:
```

	Nom	Calorie	Proteine	Vitamines	Cote
1	Tout-Son	120	3	0	33.983679
2	Tout-Son avec fibres supplementaires	70	4	25	59.425505
3	Delice d'amandes	50	4	25	93.704912



## Opérateur pour le découpage en rangs [ ](2/2)

- Rappelez-vous que l'opérateur [ ] fonctionne sur la position des lignes et non sur les étiquettes des lignes.
- Par exemple, dans le cas suivant, les étiquettes de ligne sont des chaînes de caractères. Mais nous passons les positions des lignes que nous voulons découper.

```
[30]: df.set_index('Nom', inplace = True)  
df
```

```
[30]:
```

	Calorie	Proteine	Vitamines	Cote
Nom				
100% Son Naturel	70	4	25	68.402973
Tout-Son	120	3	0	33.983679
Tout-Son avec fibres supplementaires	70	4	25	59.425505
Delice d'amandes	50	4	25	93.704912

```
[31]: df[1:4]
```

```
[31]:
```

	Calorie	Proteine	Vitamines	Cote
Nom				
Tout-Son	120	3	0	33.983679
Tout-Son avec fibres supplementaires	70	4	25	59.425505
Delice d'amandes	50	4	25	93.704912



## Quiz Time

1. Considérons le DataFrame donné appelé df. Laquelle des propositions suivantes nous donnera les lignes 5-10 du DataFrame ?
  - a) `df[5:10]`
  - b) `df[5:11]`
  - c) `df[4:10]`
  - d) `df[4:11]`





## Quiz Time

1. Considérons le DataFrame donné appelé df. Laquelle des propositions suivantes nous donnera les lignes 5-10 du DataFrame?
  - a) `df[5:10]`
  - b) `df[5:11]`
  - c) `df[4:10]`
  - d) `df[4:11]`



Remise à niveau rapide  
sur Python

Data Science avec  
Python

Structures de données  
Pandas

Structures de données Pandas

1

C'est quoi la série Pandas ?

2

C'est quoi un DataFrame de  
Pandas ?

3

Création d'un DataFrame à  
l'aide d'un dictionnaire

4

Création d'un DataFrame à  
l'aide de listes

5

Chargement d'un fichier csv en  
tant que DataFrame

6

Modification de la colonne d'index

7

inplace

8

Examen des données

9

Obtenir un résumé  
statistique du DataFrame

10

Découpage de rangées à l'aide  
de l'opérateur de crochets

11

Indexation des colonnes à  
l'aide d'opérateurs de  
parenthèse

12



## Opérateur pour l'indexation des colonnes [ ]

- Nous pouvons également utiliser l'opérateur de crochets ([ ]) pour indexer une colonne du DataFrame.
- L'indexation d'une seule colonne renvoie une série.
- L'indexation d'une liste de colonnes renvoie un DataFrame.
- Rappelez-vous que pour indexer les colonnes, nous passons leurs étiquettes à l'opérateur [ ] et non leurs positions.

```
[33]: df[['Nom', 'Cote']]
```

```
[33]:
```

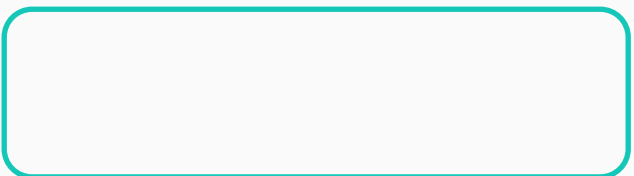
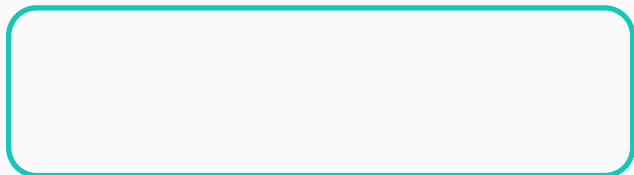
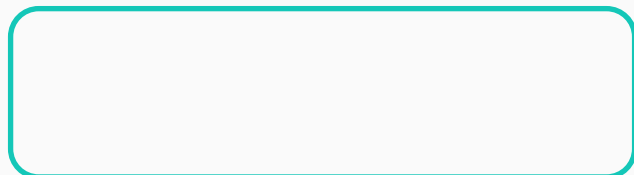
	Nom	Cote
0	100% Son Naturel	68.402973
1	Tout-Son	33.983679
2	Tout-Son avec fibres supplémentaires	59.425505
3	Delice d'amandes	93.704912
4	Cheerios pomme-cannelle	34.384843



Remise à niveau rapide  
sur Python

Data Science avec  
Python

Structures de données  
Pandas



Liste booléenne

13

14

Logical Operators

Conditional statements

15

16

Loops

Sequences: Lists

17

18

Sequences: Dictionaries

Sequences: Tuples

19

20

Functions: Built-in Functions

Functions: User-defined Functions

21



## Liste booléenne

- Nous pouvons également passer une liste de booléens à l'opérateur [ ].
- Nous obtenons toutes les lignes du DataFrame pour lesquelles l'élément correspondant de la liste est True.
- Les lignes du DataFrame pour lesquelles l'élément correspondant de la liste est False sont ignorées.
- Remarque : le DataFrame original reste inchangé.

In [71]: df

Out[71]:

	Nom	Calorie	Proteine	Vitamines	Cote
0	100% Son Naturel	70	4	25	68.402973
1	Tout-Son	120	3	0	33.983679
2	Tout-Son avec fibres supplémentaires	70	4	25	59.425505
3	Delice d'amandes	50	4	25	93.704912
4	Cheerios pomme-cannelle	110	2	25	34.384843

```
[45]: ColnTrois = [False, False, True, False, False, False, False, False, False]  
df[ColnTrois]
```

```
[45]:
```

	Nom	Calorie	Proteine	Vitamines	Cote
2	Tout-Son avec fibres supplémentaires	70	4	25	59.425505



Remise à niveau rapide  
sur Python

Data Science avec  
Python

Structures de données  
Pandas

Liste booléenne	13	
	14	Filtrage des lignes
Conditional statements	15	
	16	Loops
Sequences: Lists	17	
	18	Sequences: Dictionaries
Sequences: Tuples	19	
	20	Functions: Built-in Functions
Functions: User-defined Functions	21	



## Filtrage des lignes (1/3)

- Nous pouvons également utiliser l'opérateur [ ] pour appliquer des conditions sur une ou plusieurs colonnes du DataFrame.
- Les lignes du DataFrame qui satisfont à ces conditions sont filtrées.

```
[72]: condition=df['Calorie'] > 70  
df[condition]
```

```
[72]:
```

	Nom	Calorie	Proteine	Vitamines	Cote
1	Tout-Son	120	3	0	33.983679
4	Cheerios pomme-cannelle	110	2	25	34.384843

```
[73]: df[df['Calorie'] > 70]
```

```
[73]:
```

	Nom	Calorie	Proteine	Vitamines	Cote
1	Tout-Son	120	3	0	33.983679
4	Cheerios pomme-cannelle	110	2	25	34.384843



## Filtrage des lignes (2/3)

### et (&)

- Nous pouvons également regrouper des conditions en utilisant l'opérateur « et ».
- Le symbole pour and dans pandas est « & ». Il fonctionne de la même manière que « et » en Python.
- Remarque : chaque condition doit être entre parenthèses.

```
[74]: df[(df['Calorie'] > 70) & (df['Proteine'] < 4)]
```

```
[74]:
```

	Nom	Calorie	Proteine	Vitamines	Cote
1	Tout-Son	120	3	0	33.983679
4	Cheerios pomme-cannelle	110	2	25	34.384843
5	Pomme Jacks	110	2	25	29.509541
6	Basique 4	110	2	25	33.174094
7	Boulettes de Son	130	3	25	37.038562
8	Flocons de Son	90	2	25	49.120253





## Filtrage des lignes (3/3)

**ou (|)**

- Nous pouvons également regrouper des conditions en utilisant l'opérateur « ou ».
- Le symbole pour and dans pandas est « | » Il fonctionne de la même manière que « ou » en Python.
- Remarque : chaque condition doit être entre parenthèses.

```
[75]: df[(df['Calorie'] > 70) | (df['Proteine'] < 4)]
```

```
[75]:
```

	Nom	Calorie	Proteine	Vitamines	Cote
1	Tout-Son	120	3	0	33.983679
4	Cheerios pomme-cannelle	110	2	25	34.384843
5	Pomme Jacks	110	2	25	29.509541
6	Basique 4	110	2	25	33.174094
7	Boulettes de Son	130	3	25	37.038562
8	Flocons de Son	90	2	25	49.120253



Remise à niveau rapide  
sur Python

Data Science avec  
Python

Structures de données  
Pandas

Liste booléenne

13

Filtrage des lignes

14

loc et iloc

15

Loops

16

Sequences: Lists

17

Sequences: Dictionaries

18

Sequences: Tuples

19

Functions: Built-in Functions

20

Functions: User-defined Functions

21



## loc (1/4)

### Indexage

- loc est utilisé pour indexer/trancher un groupe de lignes et de colonnes sur la base de leurs étiquettes.
- Le premier argument est l'étiquette de la ligne et le second argument est l'étiquette de la colonne.
- Dans l'exemple suivant, nous indexons la première ligne et la première colonne.

```
[76]: df
```

```
[76]:
```

	Nom	Calorie	Proteine	Vitamines	Cote
0	100% Son Naturel	70	4	25	68.402973
1	Tout-Son	120	3	0	33.983679
2	Tout-Son avec fibres supplémentaires	70	4	25	59.425505
3	Delice d'amandes	50	4	25	93.704912
4	Cheerios pomme-cannelle	110	2	25	34.384843
5	Pomme Jacks	110	2	25	29.509541
6	Basique 4	110	2	25	33.174094
7	Boulettes de Son	130	3	25	37.038562
8	Flocons de Son	90	2	25	49.120253

```
[77]: df.loc[0, 'Nom']
```

```
[77]: '100% Son Naturel'
```



## loc (2/4)

### Indexage

- Si nous passons une liste d'étiquettes de lignes et de colonnes, nous obtenons un DataFrame.
- Dans l'exemple suivant, nous indexons la première ligne et la première colonne, mais nous transmettons les étiquettes sous forme de listes. Nous obtenons un DataFrame.

```
[78]: df
```

```
[78]:
```

	Nom	Calorie	Proteine	Vitamines	Cote
0	100% Son Naturel	70	4	25	68.402973
1	Tout-Son	120	3	0	33.983679
2	Tout-Son avec fibres supplementaires	70	4	25	59.425505
3	Delice d'amandes	50	4	25	93.704912
4	Cheerios pomme-cannelle	110	2	25	34.384843
5	Pomme Jacks	110	2	25	29.509541
6	Basique 4	110	2	25	33.174094
7	Boulettes de Son	130	3	25	37.038562
8	Flocons de Son	90	2	25	49.120253

```
[79]: df.loc[[0],['Nom']]
```

```
[79]:
```

	Nom
0	100% Son Naturel



## loc (3/4)

### Découpage

- Nous pouvons également découper des lignes et/ou des colonnes à l'aide de la méthode loc.
- Les indices de début et de fin d'une tranche avec loc sont inclusifs.
- Dans l'exemple suivant, nous découpons les 5 premières lignes et les 3 premières colonnes du DataFrame. Le résultat est un DataFrame.

```
[80]: df
```

```
[80]:
```

	Nom	Calorie	Proteine	Vitamines	Cote
0	100% Son Naturel	70	4	25	68.402973
1	Tout-Son	120	3	0	33.983679
2	Tout-Son avec fibres supplémentaires	70	4	25	59.425505
3	Delice d'amandes	50	4	25	93.704912
4	Cheerios pomme-cannelle	110	2	25	34.384843
5	Pomme Jacks	110	2	25	29.509541
6	Basique 4	110	2	25	33.174094
7	Boulettes de Son	130	3	25	37.038562
8	Flocons de Son	90	2	25	49.120253

```
[81]: df.loc[0:4, 'Nom': 'Proteine']
```

```
[81]:
```

	Nom	Calorie	Proteine
0	100% Son Naturel	70	4
1	Tout-Son	120	3
2	Tout-Son avec fibres supplémentaires	70	4
3	Delice d'amandes	50	4
4	Cheerios pomme-cannelle	110	2



## loc (4/4)

### *Indexage et Découpage*

- Nous pouvons également indexer et découper simultanément.
- Dans l'exemple suivant, nous indexons les lignes et découpons les colonnes. L'inverse est également possible.

```
[82]: df.loc[[5,8], 'Nom': 'Proteine']
```

```
[82]:
```

	Nom	Calorie	Proteine
5	Pomme Jacks	110	2
8	Flocons de Son	90	2



## Quiz Time

1. Considérons le DataFrame donné appelé df. Quel sera le résultat de la commande loc suivante ?  
`df.loc[[0, 1], ['Nom']]`
  - a) DataFrame
  - b) Série
  - c) Cellule



## Quiz Time

1. Considérons le DataFrame donné appelé df. Quel sera le résultat de la commande loc suivante ?

`df.loc[[0, 1], ['Nom']]`

- a) DataFrame
- b) Série
- c) Cellule





## iloc (1/4)

### Indexage

- iloc est utilisé pour indexer/découper un groupe de lignes et de colonnes.
- Iloc prend les positions des lignes et des colonnes comme arguments et non leurs étiquettes.
- Le premier argument est la position de la ligne et le second argument est la position de la colonne.
- Dans l'exemple suivant, nous indexons la 10ème ligne et la 3ème colonne. Le résultat est une série.

```
[83]: df
```

```
[83]:
```

	Nom	Calorie	Proteine	Vitamines	Cote
0	100% Son Naturel	70	4	25	68.402973
1	Tout-Son	120	3	0	33.983679
2	Tout-Son avec fibres supplémentaires	70	4	25	59.425505
3	Delice d'amandes	50	4	25	93.704912
4	Cheerios pomme-cannelle	110	2	25	34.384843
5	Pomme Jacks	110	2	25	29.509541
6	Basique 4	110	2	25	33.174094
7	Boulettes de Son	130	3	25	37.038562
8	Flocons de Son	90	2	25	49.120253

```
[85]: df.iloc[8, 2]
```

```
[85]: 2
```



## iloc (2/4)

### Indexage

- Si nous passons une liste de positions de lignes et de colonnes, nous obtenons un DataFrame.
- Dans l'exemple suivant, nous indexons la 10e ligne et la troisième colonne, mais nous transmettons les positions sous forme de listes. Nous obtenons un DataFrame.

```
[86]: df.iloc[[8], [2]]
```

```
[86]: Proteine  
      8      2
```



## iloc (3/4)

### Découpage

- Nous pouvons également découper des lignes et/ou des colonnes en utilisant la méthode iloc.
- Nous fournissons les positions des lignes et des colonnes pour le découpage à l'aide de iloc.
- L'index de début d'une tranche avec iloc est inclusif. Cependant, l'index de fin est exclusif.
- Dans l'exemple suivant, nous découpons les 5 premières lignes et les 3 premières colonnes du DataFrame. Le résultat est un DataFrame.

```
[87]: df.iloc[0:5, 0:3]
```

```
[87]:
```

	Nom	Calorie	Proteine
0	100% Son Naturel	70	4
1	Tout-Son	120	3
2	Tout-Son avec fibres supplémentaires	70	4
3	Delice d'amandes	50	4
4	Cheerios pomme-cannelle	110	2



## iloc (4/4)

### Indexage et Découpage

- Nous pouvons également indexer et découper simultanément.
- Dans l'exemple suivant, nous indexons les lignes et découpons les colonnes. L'inverse est également possible.

```
[88]: df.iloc[[0,2,4], 0:3]
```

```
[88]:
```

	Nom	Calorie	Proteine
0	100% Son Naturel	70	4
2	Tout-Son avec fibres supplémentaires	70	4
4	Cheerios pomme-cannelle	110	2



## Quiz Time

1. Considérons le DataFrame donné appelé df. Quel sera le résultat de la commande iloc suivante ?  
`df.loc[[0, 2], [2]]`
  - a) DataFrame
  - b) Série
  - c) Cellule
1. L'indice d'arrêt dans la tranche iloc est inclusif. Cette affirmation est-elle True ou False ?
  - a) True
  - b) False



## Quiz Time

1. Considérons le DataFrame donné appelé df. Quel sera le résultat de la commande iloc suivante?  
df.loc[[0, 2], [2]]
  - a) DataFrame
  - b) Series
  - c) Cellule
1. L'indice d'arrêt dans la tranche iloc est inclusif. Cette affirmation est-elle True ou False?
  - a) True
  - b) False



Remise à niveau rapide  
sur Python

Data Science avec  
Python

Structures de données  
Pandas

Liste booléenne

13

Filtrage des lignes

14

loc et iloc

15

Ajout et suppression de  
lignes et de colonnes

16

Sequences: Lists

17

Sequences: Dictionaries

18

Sequences: Tuples

19

Functions: Built-in Functions

20

Functions: User-defined Functions

21



# Ajout et Suppression de Lignes et Colonnes (1/4)

## Ajout de lignes

- Nous pouvons ajouter des rangées supplémentaires à notre DataFrame en utilisant la méthode loc.
- Si l'étiquette de la ligne n'existe pas, une nouvelle ligne avec l'étiquette spécifiée sera ajoutée à la fin de la ligne.

```
[89]: df
```

```
[89]:
```

	Nom	Calorie	Proteine	Vitamines	Cote
0	100% Son Naturel	70	4	25	68.402973
1	Tout-Son	120	3	0	33.983679
2	Tout-Son avec fibres supplémentaires	70	4	25	59.425505
3	Delice d'amandes	50	4	25	93.704912
4	Cheerios pomme-cannelle	110	2	25	34.384843
5	Pomme Jacks	110	2	25	29.509541
6	Basique 4	110	2	25	33.174094
7	Boulettes de Son	130	3	25	37.038562
8	Flocons de Son	90	2	25	49.120253

```
[92]: df.loc[6]=['Trix', 110, 1, 25, 27.753301]
```

```
[92]:
```

	Nom	Calorie	Proteine	Vitamines	Cote
0	100% Son Naturel	70	4	25	68.402973
1	Tout-Son	120	3	0	33.983679
2	Tout-Son avec fibres supplémentaires	70	4	25	59.425505
3	Delice d'amandes	50	4	25	93.704912
4	Cheerios pomme-cannelle	110	2	25	34.384843
5	Pomme Jacks	110	2	25	29.509541
6	Trix	110	1	25	27.753301
7	Boulettes de Son	130	3	25	37.038562
8	Flocons de Son	90	2	25	49.120253





## Ajout et Suppression de Lignes et Colonnes (2/4)

### Suppression de lignes

- Nous pouvons supprimer des lignes du DataFrame en utilisant la fonction `drop()` en spécifiant `axis=0` pour les lignes.
- Fournissez les étiquettes des lignes à supprimer comme argument à la fonction `drop()`.
- N'oubliez pas d'utiliser `inplace=True`, sinon le DataFrame original restera inchangé.

```
[93]: df.drop(2, axis=0, inplace=True)
```

```
[94]: df
```

```
[94]:
```

	Nom	Calorie	Proteine	Vitamines	Cote
0	100% Son Naturel	70	4	25	68.402973
1	Tout-Son	120	3	0	33.983679
3	Delice d'amandes	50	4	25	93.704912
4	Cheerios pomme-cannelle	110	2	25	34.384843
5	Pomme Jacks	110	2	25	29.509541
6	Trix	110	1	25	27.753301
7	Boulettes de Son	130	3	25	37.038562
8	Flocons de Son	90	2	25	49.120253



## Ajout et Suppression de Lignes et Colonnes (3/4)

### Ajouter des Colonnes

- Pour ajouter une colonne au DataFrame, nous utilisons la même notation que pour ajouter une paire clé-valeur à un dictionnaire.
- Au lieu de la clé, nous fournissons le nom de la colonne entre crochets, puis une liste de valeurs pour cette colonne.
- Si aucune colonne avec le nom donné n'existe, une nouvelle colonne avec le nom et les valeurs spécifiés sera ajoutée au DataFrame.

```
[96]: df['Ma Colonne']=['A','B','C','D','E','F','G','H']  
df
```

```
[96]:
```

	Nom	Calorie	Proteine	Vitamines	Cote	Ma Colonne
0	100% Son Naturel	70	4	25	68.402973	A
1	Tout-Son	120	3	0	33.983679	B
3	Delice d'amandes	50	4	25	93.704912	C
4	Cheerios pomme-cannelle	110	2	25	34.384843	D
5	Pomme Jacks	110	2	25	29.509541	E
6	Trix	110	1	25	27.753301	F
7	Boulettes de Son	130	3	25	37.038562	G
8	Flocons de Son	90	2	25	49.120253	H



## Ajout et Suppression de Lignes et Colonnes (4/4)

### Suppression de Colonnes

- Nous pouvons également supprimer des colonnes du DataFrame en utilisant la fonction `drop()` en spécifiant `axis=1` pour les colonnes.
- Fournissez les noms des colonnes à supprimer comme argument à la fonction `drop()`.
- N'oubliez pas d'utiliser `inplace=True`, sinon le DataFrame original restera inchangé.

```
[101... df.drop('Ma Colonne', axis=1, inplace=True)
```

```
[102... df
```

```
[102...
```

	Nom	Calorie	Proteine	Vitamines	Cote
0	100% Son Naturel	70	4	25	68.402973
1	Tout-Son	120	3	0	33.983679
3	Delice d'amandes	50	4	25	93.704912
4	Cheerios pomme-cannelle	110	2	25	34.384843
5	Pomme Jacks	110	2	25	29.509541
6	Trix	110	1	25	27.753301
7	Boulettes de Son	130	3	25	37.038562
8	Flocons de Son	90	2	25	49.120253



Remise à niveau rapide  
sur Python

Data Science avec  
Python

Structures de données  
Pandas

Liste booléenne	13	
	14	Filtrage des lignes
loc et iloc	15	
	16	Ajout et suppression de lignes et de colonnes
Tri des valeurs	17	
	18	Sequences: Dictionaries
Sequences: Tuples	19	
	20	Functions: Built-in Functions
Functions: User-defined Functions	21	



## Tri des Valeurs (1/2)

### Croissante

- Nous pouvons trier les valeurs d'un DataFrame par rapport à une colonne en utilisant la fonction `sort_values()`, qui trie par défaut les valeurs par ordre croissant.
- Si les valeurs de la colonne sont des alphabets, elles sont triées par ordre alphabétique.
- Si les valeurs de la colonne sont des nombres, elles sont triées numériquement.

```
[103... df.sort_values(by='Calorie')
```

```
[103...
```

	Nom	Calorie	Proteine	Vitamines	Cote
3	Delice d'amandes	50	4	25	93.704912
0	100% Son Naturel	70	4	25	68.402973
8	Flocons de Son	90	2	25	49.120253
4	Cheerios pomme-cannelle	110	2	25	34.384843
5	Pomme Jacks	110	2	25	29.509541
6	Trix	110	1	25	27.753301
1	Tout-Son	120	3	0	33.983679
7	Boulettes de Son	130	3	25	37.038562



## Tri des Valeurs (2/2)

### Décroissante

- Pour trier les valeurs dans l'ordre décroissant, nous définissons ascending = False dans la fonction sort\_values().

```
[104...] df.sort_values(by='Calorie', ascending = False )
```

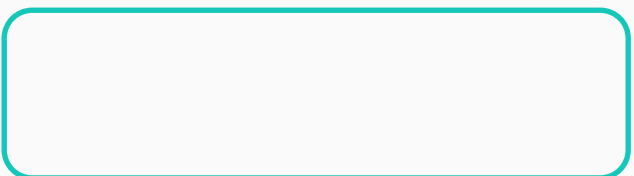
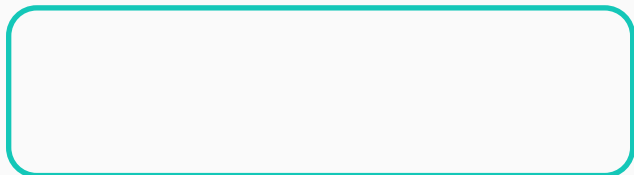
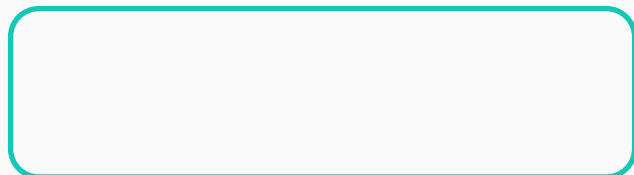
[104...]	Nom	Calorie	Proteine	Vitamines	Cote
7	Boulettes de Son	130	3	25	37.038562
1	Tout-Son	120	3	0	33.983679
4	Cheerios pomme-cannelle	110	2	25	34.384843
5	Pomme Jacks	110	2	25	29.509541
6	Trix	110	1	25	27.753301
8	Flocons de Son	90	2	25	49.120253
0	100% Son Naturel	70	4	25	68.402973
3	Delice d'amandes	50	4	25	93.704912



Remise à niveau rapide  
sur Python

Data Science avec  
Python

Structures de données  
Pandas



Liste booléenne

13

Filtrage des lignes

14

loc et iloc

15

Ajout et suppression de  
lignes et de colonnes

16

Tri des valeurs

17

Expotation et Sauvegarde  
de DataFrame Pandas

18

Sequences: Tuples

19

Functions: Built-in Functions

20

Functions: User-defined Functions

21



## Exportation et Sauvegarde de DataFrame Pandas

- Pour exporter un DataFrame en tant que fichier csv, utilisez la fonction `to_csv()`.
- Si un fichier avec le nom de fichier spécifié existe, il sera modifié. Sinon, un nouveau fichier avec le nom de fichier spécifié sera créé.
- Si vous ne voulez pas stocker la colonne d'index dans le fichier csv, vous pouvez définir `index_label=False` dans la fonction `to_csv()`.

```
[106... df.to_csv('MonFichier.csv', index_label=False)
```

```
[108... NouveauDf=pd.read_csv('MonFichier.csv')  
NouveauDf
```

```
[108... 
```

	Nom	Calorie	Proteine	Vitamines	Cote
0	100% Son Naturel	70	4	25	68.402973
1	Tout-Son	120	3	0	33.983679
3	Delice d'amandes	50	4	25	93.704912
4	Cheerios pomme-cannelle	110	2	25	34.384843
5	Pomme Jacks	110	2	25	29.509541
6	Trix	110	1	25	27.753301
7	Boulettes de Son	130	3	25	37.038562
8	Flocons de Son	90	2	25	49.120253

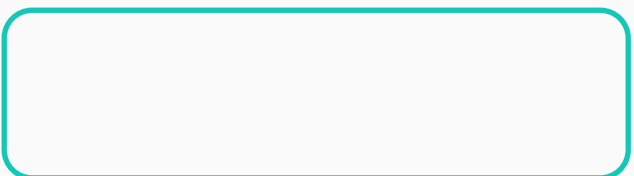
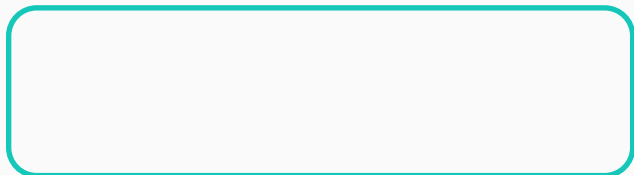
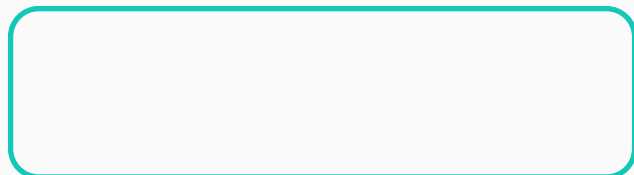




Remise à niveau rapide  
sur Python

Data Science avec  
Python

Structures de données  
Pandas



Liste booléenne

13

Filtrage des lignes

14

loc et iloc

15

Ajout et suppression de  
lignes et de colonnes

16

Tri des valeurs

17

Expotation et Sauvegarde  
de DataFrame Pandas

18

Concaténation de DataFrames

19

Functions: Built-in Functions

20

Functions: User-defined Functions

21



## Concaténation de DataFrames (1/3)

- Nous pouvons concaténer deux DataFrames ou plus à l'aide de la fonction `pandas.concat()`.

Premier  
DataFrame

	Nom	Calorie	Proteine	Vitamines	Cote
0	100% Son	70	4	25	68.402973
1	100% Son Naturel	120	3	0	33.983679
2	Tout-Son	70	4	25	59.425505
3	Tout-Son avec fibres supplémentaires	50	4	25	93.704912
4	Delice d'amandes	110	2	25	34.384843

Deuxième  
DataFrame

	Nom	Calorie	Proteine	Vitamines	Cote
0	Cheerios pomme-cannelle	110	2	25	29.509541
1	Pomme Jacks	110	2	25	33.174094
2	Basique 4	130	3	25	37.038562
3	Boulettes de Son	90	2	25	49.120253
4	Flocons de Son	90	3	25	58.313813
5	Cap'n'Crunch	120	1	25	18.042851

	Nom	Calorie	Proteine	Vitamines	Cote
0	100% Son	70	4	25	68.402973
1	100% Son Naturel	120	3	0	33.983679
2	Tout-Son	70	4	25	59.425505
3	Tout-Son avec fibres supplémentaires	50	4	25	93.704912
4	Delice d'amandes	110	2	25	34.384843
5	Cheerios pomme-cannelle	110	2	25	29.509541
6	Pomme Jacks	110	2	25	33.174094
7	Basique 4	130	3	25	37.038562
8	Boulettes de Son	90	2	25	49.120253
9	Flocons de Son	90	3	25	58.313813
10	Cap'n'Crunch	120	1	25	18.042851

DataFrame résultant



## Concaténation de DataFrames (2/3)

- Nous pouvons également concaténer deux ou plusieurs DataFrames côte à côte.

Premier  
DataFrame

	Nom	Calorie	Proteine	Vitamines	Cote
0	100% Son	70	4	25	68.402973
1	100% Son Naturel	120	3	0	33.983679
2	Tout-Son	70	4	25	59.425505

Deuxième  
DataFrame

	Nom	Calorie	Proteine	Vitamines	Cote
0	Cheerios pomme-cannelle	110	2	25	29.509541
1	Pomme Jacks	110	2	25	33.174094
2	Basique 4	130	3	25	37.038562

	Nom	Calorie	Proteine	Vitamines	Cote	Nom	Calorie	Proteine	Vitamines	Cote
0	100% Son	70	4	25	68.402973	Cheerios pomme-cannelle	110	2	25	29.509541
1	100% Son Naturel	120	3	0	33.983679	Pomme Jacks	110	2	25	33.174094
2	Tout-Son	70	4	25	59.425505	Basique 4	130	3	25	37.038562

DataFrame résultant



## Concaténation de DataFrames (3/3)

- Pour joindre deux DataFrames ou plus côte à côte, utilisez `axis = 1` dans la fonction `pandas.concat()`.

[112... df1

	Nom	Calorie	Proteine	Vitamines	Cote
0	100% Son	70	4	25	68.402973
1	100% Son Naturel	120	3	0	33.983679
2	Tout-Son	70	4	25	59.425505

[113... df2

	Nom	Calorie	Proteine	Vitamines	Cote
0	Cheerios pomme-cannelle	110	2	25	29.509541
1	Pomme Jacks	110	2	25	33.174094
2	Basique 4	130	3	25	37.038562

[114... `pd.concat([df1,df2],axis=1)`

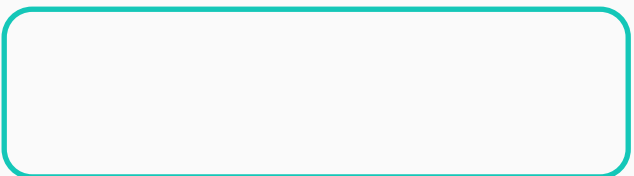
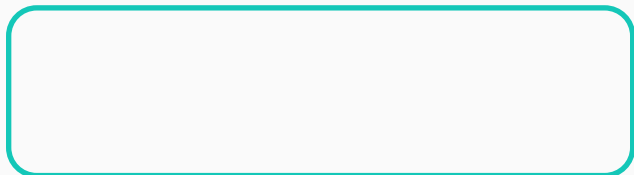
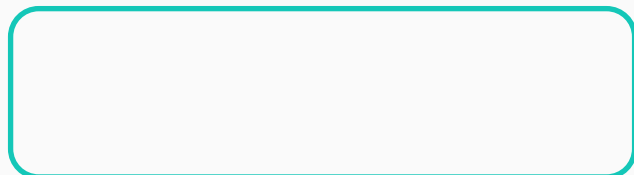
	Nom	Calorie	Proteine	Vitamines	Cote	Nom	Calorie	Proteine	Vitamines	Cote
0	100% Son	70	4	25	68.402973	Cheerios pomme-cannelle	110	2	25	29.509541
1	100% Son Naturel	120	3	0	33.983679	Pomme Jacks	110	2	25	33.174094
2	Tout-Son	70	4	25	59.425505	Basique 4	130	3	25	37.038562



Remise à niveau rapide  
sur Python

Data Science avec  
Python

Structures de données  
Pandas



Liste booléenne

13

Filtrage des lignes

14

loc et iloc

15

Ajout et suppression de  
lignes et de colonnes

16

Tri des valeurs

17

Exportation et Sauvegarde  
de DataFrame Pandas

18

Concaténation de DataFrames

19

groupby()

20



## groupby() (1/7)

- La fonction groupby() est utilisée pour regrouper les DataFrame en fonction des séries.
- Le DataFrame est divisé en groupes.
- Une fonction d'agrégation est appliquée à chaque colonne du DataFrame fractionné.
- Les résultats sont combinés ensemble.
- Considérons le DataFrame suivant.

	Sexe	Age
0	Feminin	85
1	Masculin	88
2	Feminin	95
3	Masculin	80



## groupby() (2/7)

- La colonne 'Sexe' contient deux valeurs, 'Masculin' et 'Féminin'.
- Divisons notre DataFrame en deux parties sur la base de la colonne 'Sexe' ;
  - La première partie contiendra les lignes où Sexe = Masculin.
  - La deuxième partie contiendra les lignes où Sexe = Féminin.

	Sexe	Age
0	Féminin	85
2	Féminin	95

	Sexe	Age
1	Masculin	88
3	Masculin	80



## groupby() (3/7)

- Si nous trouvons l'âge moyen des deux sexes, voici ce que nous obtenons.

Age	
Sexe	
Feminin	90

Age	
Sexe	
Masculin	84





## groupby() (4/7)

- Combinons les deux résultats ensemble. Voici ce que nous obtenons.

Age	
Sexe	
Feminin	90
Masculin	84



## groupby() (5/7)

- La fonction groupby() fonctionne exactement de la même manière, sauf qu'elle nous facilite la tâche.
- Dans l'exemple donné, nous regroupons notre DataFrame sur la base de la colonne "Sexe", puis nous lui appliquons la fonction d'agrégation mean().

```
In [55]: df
```

```
Out[55]:
```

	Sexe	Age
0	Feminin	85
1	Masculin	88
2	Feminin	95
3	Masculin	80

```
In [56]: df.groupby(x['Sexe']).mean()
```

```
Out[56]:
```

	Age
Sexe	
Feminin	90
Masculin	84



## groupby() (6/7)

- Notez que les fonctions d'agrégation sont appliquées automatiquement sur toutes les colonnes du DataFrame, à l'exception de celle utilisée pour grouper le DataFrame.

In [72]:

```
df
```

Out[72]:

	Sexe	Maths	Anglais
0	Feminin	85	80
1	Masculin	88	88
2	Feminin	95	92
3	Masculin	80	95

In [73]:

```
df.groupby(x['Sexe']).mean()
```

Out[73]:

	Maths	Anglais
Sexe		
Feminin	90.0	86.0
Masculin	84.0	91.5



## groupby() (7/7)

- Les fonctions d'agrégation courantes sont;
  - mean()
  - sum()
  - max()
  - min()
  - median()
  - count()
  - std() (standard deviation)



## *Documentation*

- <https://www.geeksforgeeks.org/python-pandas-dataframe/?ref=lbp>
- [https://pandas.pydata.org/pandas-docs/stable/user\\_guide/dsintro.html](https://pandas.pydata.org/pandas-docs/stable/user_guide/dsintro.html)
- [https://www.w3schools.com/python/pandas/pandas\\_dataframes.asp](https://www.w3schools.com/python/pandas/pandas_dataframes.asp)