Pandas

Entrée [ ]:

```
```

Entrée [1]:

```
import pandas as pd
```

Entrée [2]:

```
data = [
    {'beer_name': 'a', 'type': 'blonde', 'price': 2},
    {'beer_name': 'b', 'type': 'blonde', 'price': 2},
    {'beer_name': 'c', 'type': 'blonde', 'price': 3},
    {'beer_name': 'd', 'type': 'brune', 'price': 4},
    {'beer_name': 'e', 'type': 'ale', 'price': 2}
]
```

Entrée [3]:

```
df = pd.DataFrame(data)  # créer un dataframe à partir d'une liste de dict
```

Entrée [4]:

```
df
```

Out[4]:

	beer_name	type	price
0	a	blonde	2
1	b	blonde	2
2	c	blonde	3
3	d	brune	4
4	e	ale	2

Entrée [5]:

```
df.columns
```

Out[5]:

```
Index(['beer_name', 'type', 'price'], dtype='object')
```

Entrée [6]:

```
type(df)
```

Out[6]:

```
pandas.core.frame.DataFrame
```

Entrée [7]:

```
# Créer un dataframe à partir d'un dict de list:
df2 = pd.DataFrame({
    'x': [1,2,3,5],
    'y': [4,22,33,5]
})
```

Entrée [8]:

```
df2
```

Out[8]:

	x	y
0	1	4
1	2	22
2	3	33
3	5	5

Entrée [9]:

```
df2.shape
```

Out[9]:

```
(4, 2)
```

Entrée [10]:

```
df2.columns
```

Out[10]:

```
Index(['x', 'y'], dtype='object')
```

Entrée [11]:

```
df2.x
```

Out[11]:

```
0    1
1    2
2    3
3    5
Name: x, dtype: int64
```

Entrée [12]:

```
type(df2.x)
```

Out[12]:

```
pandas.core.series.Series
```

Entrée [13]:

```
l = [1, 2, 3, None]
s = pd.Series(l)
```

Entrée [14]:

```
l
```

Out[14]:

```
[1, 2, 3, None]
```

Entrée [15]:

```
s
```

Out[15]:

```
0    1.0
1    2.0
2    3.0
3    NaN
dtype: float64
```

Entrée [16]:

```
s.mean()
```

Out[16]:

```
2.0
```

Entrée [17]:

```
df2['x']
```

Out[17]:

```
0    1
1    2
2    3
3    5
Name: x, dtype: int64
```

Entrée [18]:

```
df2['x'] + 10
```

Out[18]:

```
0    11
1    12
2    13
3    15
Name: x, dtype: int64
```

Entrée [19]:

```
df2['x'] + df2['y']
```

Out[19]:

```
0     5
1    24
2    36
3    10
dtype: int64
```

Entrée [20]:

```
df2['x'] > 2  # renvoie une série de booléens
```

Out[20]:

```
0    False
1    False
2     True
3     True
Name: x, dtype: bool
```

Entrée [21]:

```
(df2['x'] > 2) & (df2['x'] < 5)  # on peut combiner les séries de booléens avec & e
```

Out[21]:

```
0    False
1    False
2     True
3    False
Name: x, dtype: bool
```

Entrée [22]:

```
df
```

Out[22]:

	beer_name	type	price
0	a	blonde	2
1	b	blonde	2
2	c	blonde	3
3	d	brune	4
4	e	ale	2

Entrée [23]:

```
df['price']
```

Out[23]:

```
0    2
1    2
2    3
3    4
4    2
Name: price, dtype: int64
```

Entrée [24]:

```
bool_serie = df['price'] > 2
bool_serie
```

Out[24]:

```
0    False
1    False
2     True
3     True
4    False
Name: price, dtype: bool
```

Entrée [25]:

```
# on peut se servir de la syntaxe [] avec une série de booléens pour filtrer des li
df[bool_serie]
```

Out[25]:

	beer_name	type	price
2	c	blonde	3
3	d	brune	4

Entrée [26]:

```
# Ici on extrait les lignes qui on un price <= 2 + celles qui ont un price > 3
df[(df['price'] > 3) | (df['price'] <= 2)]
```

Out[26]:

	beer_name	type	price
0	a	blonde	2
1	b	blonde	2
3	d	brune	4
4	e	ale	2

Entrée [27]:

```
s = df.price
s
```

Out[27]:

```
0    2
1    2
2    3
3    4
4    2
Name: price, dtype: int64
```

Entrée [28]:

```
s.sort_values()
```

Out[28]:

```
0    2
1    2
4    2
2    3
3    4
Name: price, dtype: int64
```

Entrée [29]:

```
df.sort_values('price')
```

Out[29]:

	beer_name	type	price
0	a	blonde	2
1	b	blonde	2
4	e	ale	2
2	c	blonde	3
3	d	brune	4

Entrée [30]:

```
df.sort_values(['type', 'price'])  # tri sur plusieurs colonnes
```

Out[30]:

	beer_name	type	price
4	e	ale	2
0	a	blonde	2
1	b	blonde	2
2	c	blonde	3
3	d	brune	4

Entrée [31]:

```
dfsorted = df.sort_values(['type', 'price'])
dfsorted
```

Out[31]:

	beer_name	type	price
4	e	ale	2
0	a	blonde	2
1	b	blonde	2
2	c	blonde	3
3	d	brune	4

Entrée [32]:

```
dfsorted.sort_index()  # trier par l'index
```

Out[32]:

	beer_name	type	price
0	a	blonde	2
1	b	blonde	2
2	c	blonde	3
3	d	brune	4
4	e	ale	2

Entrée [33]:

```
df['type']  # une colonne qui contient des str
```

Out[33]:

```
0    blonde
1    blonde
2    blonde
3     brune
4       ale
Name: type, dtype: object
```

Entrée [34]:

```
# L'attribut .str permet d'accéder aux méthodes des str habituelles:
df['type'].str.startswith('b')
```

Out[34]:

```
0     True
1     True
2     True
3     True
4    False
Name: type, dtype: bool
```

Entrée [35]:

```
df
```

Out[35]:

	beer_name	type	price
0	a	blonde	2
1	b	blonde	2
2	c	blonde	3
3	d	brune	4
4	e	ale	2

Entrée [36]:

```
df[['price', 'type']]  # prendre un sous-ensemble des colonnes
```

Out[36]:

	price	type
0	2	blonde
1	2	blonde
2	3	blonde
3	4	brune
4	2	ale

Entrée [37]:

```
df.loc[[0, 2]]  # extrait les lignes à l'index 0 et 2
```

Out[37]:

	beer_name	type	price
0	a	blonde	2
2	c	blonde	3

Entrée [38]:

```
df.loc[0:2]  # extrait les lignes aux index 0, 1, et 2 *inclus*
```

Out[38]:

	beer_name	type	price
0	a	blonde	2
1	b	blonde	2
2	c	blonde	3

Entrée [39]:

```
# extrait les lignes aux index 1, 2, et 3 *inclus*, avec les colonnes price et type
df.loc[1:3, ['price', 'type']]
```

Out[39]:

	price	type
1	2	blonde
2	3	blonde
3	4	brune

Entrée [40]:

```
df['plop'] = [10, 11, 12, 13, 14]  # ajout d'une colonne
```

Entrée [41]:

```
df
```

Out[41]:

	beer_name	type	price	plop
0	a	blonde	2	10
1	b	blonde	2	11
2	c	blonde	3	12
3	d	brune	4	13
4	e	ale	2	14

Entrée [42]:

```
df2 = df.set_index('plop')  # on place la colonne 'plop' en index
df2
```

Out[42]:

	beer_name	type	price
plop			
10	a	blonde	2
11	b	blonde	2
12	c	blonde	3
13	d	brune	4
14	e	ale	2

Entrée [43]:

```
df2.loc[10:12, 'beer_name':'price']
```

Out[43]:

plop	beer_name	type	price
10	a	blonde	2
11	b	blonde	2
12	c	blonde	3

Entrée [44]:

```
df_with_beername_index = df.set_index('beer_name')
```

Entrée [45]:

```
df_with_beername_index
```

Out[45]:

beer_name	type	price	plop
a	blonde	2	10
b	blonde	2	11
c	blonde	3	12
d	brune	4	13
e	ale	2	14

Entrée [46]:

```
df_with_beername_index.loc[['a', 'c'], ['price', 'plop']]
```

Out[46]:

beer_name	price	plop
a	2	10
c	3	12

Entrée [47]:

```
df_with_type_index = df.set_index('type')
df_with_type_index
```

Out[47]:

type	beer_name	price	plop
blonde	a	2	10
blonde	b	2	11
blonde	c	3	12
brune	d	4	13
ale	e	2	14

Entrée [48]:

```
df_with_type_index.loc['blonde']
```

Out[48]:

type	beer_name	price	plop
blonde	a	2	10
blonde	b	2	11
blonde	c	3	12

Entrée [49]:

```
# iloc est similaire à loc sauf qu'il prend en paramètre les positions des lignes/c
# contrairement à loc qui prend leurs valeurs.
df_with_type_index.iloc[:2, 1:]  # extrait les lignes 0 à 2 (exclue) et les colonne
```

Out[49]:

type	price	plop
blonde	2	10
blonde	2	11

Entrée [50]:

```
df
```

Out[50]:

	beer_name	type	price	plop
0	a	blonde	2	10
1	b	blonde	2	11
2	c	blonde	3	12
3	d	brune	4	13
4	e	ale	2	14

Entrée [51]:

```
# Extrait la valeur de la 1ère cellule:
df.iloc[0, 0]
```

Out[51]:

```
'a'
```

Entrée [52]:

```
dfbis = df.copy()
```

Entrée [53]:

```
dfbis
```

Out[53]:

	beer_name	type	price	plop
0	a	blonde	2	10
1	b	blonde	2	11
2	c	blonde	3	12
3	d	brune	4	13
4	e	ale	2	14

Entrée [54]:

```
# Concaténer 2 dataframes:
pd.concat([
    df,
    dfbis
])
```

Out[54]:

	beer_name	type	price	plop
0	a	blonde	2	10
1	b	blonde	2	11
2	c	blonde	3	12
3	d	brune	4	13
4	e	ale	2	14
0	a	blonde	2	10
1	b	blonde	2	11
2	c	blonde	3	12
3	d	brune	4	13
4	e	ale	2	14

Entrée [55]:

```
df_with_type_index
```

Out[55]:

type	beer_name	price	plop
blonde	a	2	10
blonde	b	2	11
blonde	c	3	12
brune	d	4	13
ale	e	2	14

Entrée [56]:

```
df_with_type_index.reset_index()
```

Out[56]:

	type	beer_name	price	plop
0	blonde	a	2	10
1	blonde	b	2	11
2	blonde	c	3	12
3	brune	d	4	13
4	ale	e	2	14

Entrée [57]:

```
for elem in df.index:
    print(elem)
```

```
0
1
2
3
4
```

Entrée [58]:

```
df.set_index(['type', 'price']).sort_index()
```

Out[58]:

		beer_name	plop
type	price		
ale	2	e	14
	2	a	10
blonde	2	b	11
	3	c	12
brune	4	d	13

Entrée [59]:

```
df.to_csv('output.csv', index=False)  # exporter le dataframe dans un fichier csv
```

Entrée [60]:

```
pd.read_csv('output.csv')  # créer un dataframe à partir d'un csv
```

Out[60]:

	beer_name	type	price	plop
0	a	blonde	2	10
1	b	blonde	2	11
2	c	blonde	3	12
3	d	brune	4	13
4	e	ale	2	14

Entrée [61]:

```
pd.read_csv('output2.csv')
```

Out[61]:

	beer_name	price	type	good
0	a	2	blonde	non renseigne
1	b	2	blonde	oui
2	c	3	blonde	oui
3	d	4	brune	non renseigne
4	e	2	ale	non renseigne
5	x	1	blonde	oui

Entrée [63]:

```
# read_csv dispose de plein d'options
pd.read_csv('output2.csv', na_values=['non renseigne'], true_values=['oui'])
```

Out[63]:

	beer_name	price	type	good
0	a	2	blonde	NaN
1	b	2	blonde	True
2	c	3	blonde	True
3	d	4	brune	NaN
4	e	2	ale	NaN
5	x	1	blonde	True