Pandas

① import pandas as pd

② [illegible] متغير = pd. series()

series has two argument

```
gro = pd. series ( data[30, 6, 'yes', 'No'] , index
      = ['eggs', 'apple', 'milk', 'bread'])
```

out eggs 30
apples 6
milk yes
bread No

③ اسم متغير . shape

out (4,) احجام كل بعد من ابعاد البيانات

④ اسم متغير . ndim

out 1 عدد ابعاد البيانات

⑤ اسم متغير . size

out 4 (الحجم الاجمالي للقيم في الموجودة)

⑥ للتاكد من وجود ال index في السلسلة

'banana' in gro

out False ✓

'bread' in gro

out True

⑦ gro ['eggs']

out 30

⑧ gro [['milk', 'bread']]

out milk yes

bread No

dtype: object

⑨ gro [0]

out 3o العنصر الاول

⑩ gro [[0, 1]]

out eggs 30

apples 6

⑪ gro.loc [['eggs', 'apples']]

out eggs 30

apples 6

⑫ gro.iloc [[2, 3]]

out milk yes

bread No

⑬ gro ['eggs'] = 2

gro ['eggs']

out 2

(16) يقوم بمسح عنصر drop ('apples') او axis

ولكن لا يغير السلسله الاصليه

out eggs 2

milk yes

bread No

(15) يستقبل السلسله الاصليه inplace drop [apples, inplace =true]

(14) يمكن اجراء عمليات حسابيه على السلسله من حيث العناصر

import pandas as pd

fruits = pd.series([10, 6, 3], ['apples', 'oranges'])

(17) Fruits + 2

out

apples 12

oranges 8

(13) عندما تكون البيانات مختلطة تأكد من العمليات الحسابيه التي تقوم بها ,

عند الضرب في 2 يكرر النص مرتين

" القسمه على 2 يعطي error

(19) DataFrame

انشاء سيريس او تحميل البيانات من ملف

سيريس من قاموس ①

```
import pandas as pd
items = {'Bob': pd.Series([245, 25, 55], index=['bike', 'pants', 'watch']), 'Alice': pd.Series([40, 110, 500, 45], index=['book', 'glasses', 'bike', 'pants'])}
type(items)
out dict
```

(20) `shopping = pd.DataFrame(items)`

```
shopping
```

out

	Alice	Bob
bike	500,0	245,0
book	40,0	NAN
glasses	140,0	NAN
pants	45,0	25,0
watch	NaN	55,0

(21) shopping.index

out index(['bike', 'book', 'glasses', 'pants',
'watch'], dtype='object')

(22) shopping.columns

out Index(['Alice', 'Bob'], dtype='object')

(23) shopping.values

array([[500., 245.],
[400., nan],
[110., 80.],
[45., 25.],
[nan, 55.]])

(24) shopping.ndim

out 2

(25) shopping.shape

out (5, 2)

(26) shopping.size

out 10

(27) bobshopping = pd.DataFrame(items, columns=['Bob'])

out

	Bob
Bike	245
Pants	25
watch	55

In [1]:
```
import pandas as pd
```

In [2]:
```
items = [{'bikes': 20, 'pants': 30, 'watches': 35}, {'watches': 10, 'glasses': 50, 'bikes': 15, 'pants': 5}]

store_items = pd.DataFrame(items, index=['store 1', 'store 2'])
store_items
```

Out[2]:

	bikes	glasses	pants	watches
store 1	20	NaN	30	35
store 2	15	50.0	5	10

In [3]:
```
store_items[['bikes']]
```

Out[3]:

	bikes
store 1	20
store 2	15

```
In [4]: store_items[['bikes', 'pants']]
```

Out[4]:

	bikes	pants
store 1	20	30
store 2	15	5

```
In [5]: store_items.loc[['store 1']]
```

Out[5]:

	bikes	glasses	pants	watches
store 1	20	NaN	30	35

```
In [6]: store_items['bikes']['store 2']
```

```
Out[6]: 15
```

In [8]:
```
store_items['bikes']['store 2']
```

Out[8]: 15

In [9]:
```
store_items['shirts'] = [15, 2]
store_items
```

Out[9]:

	bikes	glasses	pants	watches	shirts
store 1	20	NaN	30	35	15
store 2	15	50.0	5	10	2

Out[9]:

	bikes	glasses	pants	watches	shirts
store 1	20	NaN	30	35	15
store 2	15	50.0	5	10	2

In [10]:
```
store_items['suits'] = store_items['shirts'] + store_items['pants']
store_items
```

Out[10]:

	bikes	glasses	pants	watches	shirts	suits
store 1	20	NaN	30	35	15	45
store 2	15	50.0	5	10	2	7

In [10]:
```
store_items['suits'] = store_items['shirts'] + store_items['pants']
store_items
```

Out[10]:

	bikes	glasses	pants	watches	shirts	suits
store 1	20	NaN	30	35	15	45
store 2	15	50.0	5	10	2	7

In [11]:
```
new_items = [{'bikes': 20, 'pants': 30, 'watches': 35, 'glasses': 4}]

new_store = pd.DataFrame(new_items, index=['store 3'])
new_store
```

Out[11]:

	bikes	glasses	pants	watches
store 3	20	4	30	35

Out[12]:

	bikes	glasses	pants	shirts	suits	watches
store 1	20	NaN	30	15.0	45.0	35
store 2	15	50.0	5	2.0	7.0	10
store 3	20	4.0	30	NaN	NaN	35

In [14]:
```
store_items['new_watches'] = store_items['watches'][1:]
store_items
```

Out[14]:

	bikes	glasses	pants	shirts	suits	watches	new_watches
store 1	20	NaN	30	15.0	45.0	35	NaN
store 2	15	50.0	5	2.0	7.0	10	10.0
store 3	20	4.0	30	NaN	NaN	35	35.0

In [15]:
```
store_items.insert(5, 'shoes', [8, 5, 0])
store_items
```

Out[15]:

	bikes	glasses	pants	shirts	suits	shoes	watches	new_watches
store 1	20	NaN	30	15.0	45.0	8	35	NaN
store 2	15	50.0	5	2.0	7.0	5	10	10.0
store 3	20	4.0	30	NaN	NaN	0	35	35.0

In [17]:
```
store_items.pop('new_watches')
store_items
```

Out[17]:

	bikes	glasses	pants	shirts	suits	shoes	watches
store 1	20	NaN	30	15.0	45.0	8	35
store 2	15	50.0	5	2.0	7.0	5	10
store 3	20	4.0	30	NaN	NaN	0	35

In [18]:
```
store_items = store_items.drop(['watches', 'shoes'], axis=1)
store_items
```

Out[18]:

	bikes	glasses	pants	shirts	suits
store 1	20	NaN	30	15.0	45.0
store 2	15	50.0	5	2.0	7.0
store 3	20	4.0	30	NaN	NaN

In [19]:
```
store_items = store_items.drop(['store 1', 'store 2'], axis=0)
store_items
```

Out[19]:

	bikes	glasses	pants	shirts	suits
store 3	20	4.0	30	NaN	NaN

```
In [18]: store_items
```

Out[18]:

	bikes	glasses	pants	shirts	suits
store 1	20	NaN	30	15.0	45.0
store 2	15	50.0	5	2.0	7.0
store 3	20	4.0	30	NaN	NaN

```
In [19]: store_items = store_items.drop(['store 1', 'store 2'], axis=0)
         store_items
```

Out[19]:

	bikes	glasses	pants	shirts	suits
store 3	20	4.0	30	NaN	NaN

```
In [20]: store_items = store_items.rename(columns={'bikes': 'hats'})
         store_items
```

Out[20]:

	hats	glasses	pants	shirts	suits
store 3	20	4.0	30	NaN	NaN

```
In [ ]:
```

الآن ، دعنا نغير تسمية الصف باستخدام طريقة إعادة التسمية.

Pandas

localhost:8888/notebooks/Pandas.ipynb

jupyter Pandas Last Checkpoint: 19 minutes ago (unsaved changes)

Logout

File Edit View Insert Cell Kernel Help

Trusted | Python 3

Code

Out[19]:

	bikes	glasses	pants	shirts	suits
store 3	20	4.0	30	NaN	NaN

In [20]:
```
store_items = store_items.rename(columns={'bikes': 'hats'})
store_items
```

Out[20]:

	hats	glasses	pants	shirts	suits
store 3	20	4.0	30	NaN	NaN

In [21]:
```
store_items = store_items.rename(index={'store 3': 'last store'})
store_items
```

Out[21]:

	hats	glasses	pants	shirts	suits
last store	20	4.0	30	NaN	NaN

In [ ]:

يمكننا أيضًا تعيين الفهرس ليكون أحد الأعمدة الموجودة في DataFrame ، مثل هذا.

In [20]:
```
store_items = store_items.rename(columns={'bikes': 'hats'})
store_items
```

Out[20]:

	hats	glasses	pants	shirts	suits
store 3	20	4.0	30	NaN	NaN

In [21]:
```
store_items = store_items.rename(index={'store 3': 'last store'})
store_items
```

Out[21]:

	hats	glasses	pants	shirts	suits
last store	20	4.0	30	NaN	NaN

In [21]:
```
store_items = store_items.rename(index={'store 3': 'last store'})
store_items
```

Out[21]:

	hats	glasses	pants	shirts	suits
last store	20	4.0	30	NaN	NaN

In [22]:
```
store_items = store_items.set_index('pants')
store_items
```

Out[22]:

pants	hats	glasses	shirts	suits
30	20	4.0	NaN	NaN

Pandas

localhost:8888/notebooks/Pandas.ipynb

jupyter Pandas Last Checkpoint: a few seconds ago (unsaved changes)

Logout

File Edit View Insert Cell Kernel Help

Trusted | Python 3

Code

In [1]:
```
import pandas as pd

items = [{'bikes': 20, 'pants': 30, 'watches': 35, 'shirts': 15, 'shoes':8, 'suits':45},
         {'watches': 10, 'glasses': 50, 'bikes': 15, 'pants':5, 'shirts': 2, 'shoes':5, 'suits':7},
         {'bikes': 20, 'pants': 30, 'watches': 35, 'glasses': 4, 'shoes':10}]

store_items = pd.DataFrame(items, index=['store 1', 'store 2', 'store 3'])
store_items
```

Out[1]:

	bikes	glasses	pants	shirts	shoes	suits	watches
store 1	20	NaN	30	15.0	8	45.0	35
store 2	15	50.0	5	2.0	5	7.0	10
store 3	20	4.0	30	NaN	10	NaN	35

In [ ]:

هذا يعني أننا بحاجة إلى طريقة لاكتشاف الأخطاء وتصحيحها في بياناتنا.

In [1]:
```
import pandas as pd

items = [{'bikes': 20, 'pants': 30, 'watches': 35, 'shirts': 15, 'shoes':8, 'suits':45},
         {'watches': 10, 'glasses': 50, 'bikes': 15, 'pants':5, 'shirts': 2, 'shoes':5, 'suits':7},
         {'bikes': 20, 'pants': 30, 'watches': 35, 'glasses': 4, 'shoes':10}]

store_items = pd.DataFrame(items, index=['store 1', 'store 2', 'store 3'])
store_items
```

Out[1]:

	bikes	glasses	pants	shirts	shoes	suits	watches
store 1	20	NaN	30	15.0	8	45.0	35
store 2	15	50.0	5	2.0	5	7.0	10
store 3	20	4.0	30	NaN	10	NaN	35

In [2]:
```
x = store_items.isnull().sum().sum()
print(x)
```

```
3
```

In [ ]:

عدد قيم NaN في DataFrame الخاص بنا. دعونا نكسر هذا. تقوم

In [1]:
```
import pandas as pd

items = [{'bikes': 20, 'pants': 30, 'watches': 35, 'shirts': 15, 'shoes':8, 'suits':45},
         {'watches': 10, 'glasses': 50, 'bikes': 15, 'pants':5, 'shirts': 2, 'shoes':5, 'suits':7},
         {'bikes': 20, 'pants': 30, 'watches': 35, 'glasses': 4, 'shoes':10}]

store_items = pd.DataFrame(items, index=['store 1', 'store 2', 'store 3'])
store_items
```

Out[1]:

	bikes	glasses	pants	shirts	shoes	suits	watches
store 1	20	NaN	30	15.0	8	45.0	35
store 2	15	50.0	5	2.0	5	7.0	10
store 3	20	4.0	30	NaN	10	NaN	35

In [2]:
```
x = store_items.isnull().sum().sum()
print(x)
```

```
3
```

Out[1]:

	bikes	glasses	pants	shirts	shoes	suits	watches
store 1	20	NaN	30	15.0	8	45.0	35
store 2	15	50.0	5	2.0	5	7.0	10
store 3	20	4.0	30	NaN	10	NaN	35

In [3]:
```
x = store_items.isnull() #.sum().sum()
print(x)
```

```
         bikes  glasses  pants  shirts  shoes  suits  watches
store 1  False     True  False   False  False  False    False
store 2  False    False  False   False  False  False    False
store 3  False    False  False    True  False   True    False
```

In [4]:
```
x = store_items.isnull().sum() #.sum()
print(x)
```

```
bikes      0
glasses    1
pants      0
shirts     1
shoes      0
suits      1
watches    0
dtype: int64
```

In [5]:
```
x = store_items.isnull().sum().sum()
print(x)
```

```
3
```

In [6]:
```
store_items.count()
```

Out[6]:
```
bikes      3
glasses    2
pants      3
shirts     2
shoes      3
suits      2
watches    3
dtype: int64
```

```
In [7]: store_items.dropna(axis=0)
```

Out[7]:

	bikes	glasses	pants	shirts	shoes	suits	watches
store 2	15	50.0	5	2.0	5	7.0	10

```
In [8]: store_items.dropna(axis=1)
```

Out[8]:

	bikes	pants	shoes	watches
store 1	20	30	8	35
store 2	15	5	5	10
store 3	20	30	10	35

```
In [9]: store_items.fillna(0)
```

Out[9]:

	bikes	glasses	pants	shirts	shoes	suits	watches
store 1	20	0.0	30	15.0	8	45.0	35
store 2	15	50.0	5	2.0	5	7.0	10
store 3	20	4.0	30	0.0	10	0.0	35

```
In [10]: store_items.fillna(method='ffill', axis=0)
```

Out[10]:

	bikes	glasses	pants	shirts	shoes	suits	watches
store 1	20	NaN	30	15.0	8	45.0	35
store 2	15	50.0	5	2.0	5	7.0	10
store 3	20	4.0	30	2.0	10	7.0	35

```
In [12]: store_items.fillna(method='ffill', axis=1)
```

Out[12]:

	bikes	glasses	pants	shirts	shoes	suits	watches
store 1	20.0	20.0	30.0	15.0	8.0	45.0	35.0
store 2	15.0	50.0	5.0	2.0	5.0	7.0	10.0
store 3	20.0	4.0	30.0	30.0	10.0	10.0	35.0

```
In [13]: store_items.fillna(method='backfill', axis=0)
```

Out[13]:

	bikes	glasses	pants	shirts	shoes	suits	watches
store 1	20	50.0	30	15.0	8	45.0	35
store 2	15	50.0	5	2.0	5	7.0	10
store 3	20	4.0	30	NaN	10	NaN	35

In [14]:
```
store_items.interpolate(method='linear', axis=0)
```

Out[14]:

	bikes	glasses	pants	shirts	shoes	suits	watches
store 1	20	NaN	30	15.0	8	45.0	35
store 2	15	50.0	5	2.0	5	7.0	10
store 3	20	4.0	30	2.0	10	7.0	35

In [15]:
```
store_items.interpolate(method='linear', axis=1)
```

Out[15]:

	bikes	glasses	pants	shirts	shoes	suits	watches
store 1	20.0	25.0	30.0	15.0	8.0	45.0	35.0
store 2	15.0	50.0	5.0	2.0	5.0	7.0	10.0
store 3	20.0	4.0	30.0	20.0	10.0	22.5	35.0

In [1]:
```
import pandas as pd
```

In [2]:
```
google_stock = pd.read_csv('./GOOG.csv')

print(type(google_stock))
print(google_stock.shape)
```

```
<class 'pandas.core.frame.DataFrame'>
(3313, 7)
```

In [3]:
```
google_stock
```

Out[3]:

	Date	Open	High	Low	Close	Adj Close	Volume
0	2004-08-19	49.676899	51.693783	47.669952	49.845802	49.845802	44994500
1	2004-08-20	50.178635	54.187561	49.925285	53.805050	53.805050	23005800
2	2004-08-23	55.017166	56.373344	54.172661	54.346527	54.346527	18393200
3	2004-08-24	55.260582	55.439419	51.450363	52.096165	52.096165	15361800
4	2004-08-25	52.140873	53.651051	51.604362	52.657513	52.657513	9257400
5	2004-08-26	52.135906	53.626213	51.991844	53.606342	53.606342	7148200
6	2004-08-27	53.700729	53.959049	[illegible]	[illegible]	[illegible]	[illegible]
7	2004-08-30	52.299839	52.404160	50.675404	50.675404	50.675404	5235700

لنلق نظرة على بيانات الأسهم.

3302	2017-09-29	952.000000	959.786011	951.510010	959.109985	959.109985	1581000
3303	2017-10-02	959.979980	962.539978	947.840027	953.270020	953.270020	1283400
3304	2017-10-03	954.000000	958.000000	949.140015	957.789978	957.789978	888300
3305	2017-10-04	957.000000	960.390015	950.690002	951.679993	951.679993	952400
3306	2017-10-05	955.489990	970.909973	955.179993	969.960022	969.960022	1213800
3307	2017-10-06	966.700012	979.460022	963.359985	978.890015	978.890015	1173900
3308	2017-10-09	980.000000	985.424988	976.109985	977.000000	977.000000	891400
3309	2017-10-10	980.000000	981.570007	966.080017	972.599976	972.599976	968400
3310	2017-10-11	973.719971	990.710022	972.250000	989.250000	989.250000	1693300
3311	2017-10-12	987.450012	994.119995	985.000000	987.830017	987.830017	1262400
3312	2017-10-13	992.000000	997.210022	989.000000	989.679993	989.679993	1157700

3313 rows × 7 columns

In [4]:
```
google_stock.head()
```

Out[4]:

	Date	Open	High	Low	Close	Adj Close	Volume
0	2004-08-19	49.676899	51.693783	47.669952	49.845802	49.845802	44994500
1	2004-08-20	50.178635	54.187561	49.925285	53.805050	53.805050	23005800
2	2004-08-23	55.017166	56.373344	54.172661	54.346527	54.346527	18393200
3	2004-08-24	55.260582	55.439419	51.450363	52.096165	52.096165	15361800
4	2004-08-25	52.140873	53.651051	51.604362	52.657513	52.657513	9257400

In [5]:
```
google_stock.tail()
```

Out[5]:

	Date	Open	High	Low	Close	Adj Close	Volume
3308	2017-10-09	980.000000	985.424988	976.109985	977.000000	977.000000	891400
3309	2017-10-10	980.000000	981.570007	966.080017	972.599976	972.599976	968400
3310	2017-10-11	973.719971	990.710022	972.250000	989.250000	989.250000	1693300
3311	2017-10-12	987.450012	994.119995	985.000000	987.830017	987.830017	1262400
3312	2017-10-13	992.000000	997.210022	989.000000	989.679993	989.679993	1157700

1	2004-08-20	50.178535	54.187561	49.925285	53.805050	53.805050	23005800

In [6]:
```
google_stock.tail(8)
```

Out[6]:

	Date	Open	High	Low	Close	Adj Close	Volume
3305	2017-10-04	957.000000	960.390015	950.690002	951.679993	951.679993	952400
3306	2017-10-05	955.489990	970.909973	955.179993	969.960022	969.960022	1213800
3307	2017-10-06	966.700012	979.460022	963.359985	978.890015	978.890015	1173900
3308	2017-10-09	980.000000	985.424988	976.109985	977.000000	977.000000	891400
3309	2017-10-10	980.000000	981.570007	966.080017	972.599976	972.599976	968400
3310	2017-10-11	973.719971	990.710022	972.250000	989.250000	989.250000	1693300
3311	2017-10-12	987.450012	994.119995	985.000000	987.830017	987.830017	1262400
3312	2017-10-13	992.000000	997.210022	989.000000	989.679993	989.679993	1157700

```
In [8]: google_stock.isnull().any()
```

```
Out[8]: Date          False
        Open          False
        High          False
        Low           False
        Close         False
        Adj Close     False
        Volume        False
        dtype: bool
```

```
In [ ]:
```

In [9]:
```
google_stock.describe()
```

Out[9]:

	Open	High	Low	Close	Adj Close	Volume
count	3313.000000	3313.000000	3313.000000	3313.000000	3313.000000	3.313000e+03
mean	380.186092	383.493740	376.519309	380.072458	380.072458	8.038476e+06
std	223.818650	224.974534	222.473232	223.853780	223.853780	8.399521e+06
min	49.274517	50.541279	47.669952	49.681866	49.681866	7.900000e+03
25%	226.556473	228.394516	224.003082	226.407440	226.407440	2.584900e+06
50%	293.312286	295.433502	289.929291	293.029114	293.029114	5.281300e+06
75%	536.650024	540.000000	532.409973	536.690002	536.690002	1.065370e+07
max	992.000000	997.210022	989.000000	989.679993	989.679993	8.276810e+07

```
In [10]: google_stock['Adj Close'].describe()

Out[10]: count    3313.000000
         mean      380.072458
         std       223.853780
         min        49.681866
         25%       226.407440
         50%       293.029114
         75%       536.690002
         max       989.679993
         Name: Adj Close, dtype: float64
```

```
In [11]: google_stock.max()

Out[11]: Date          2017-10-13
         Open                992
         High             997.21
         Low                 989
         Close            989.68
         Adj Close        989.68
         Volume         82768100
         dtype: object

In [12]: google_stock.mean()

Out[12]: Open          3.801861e+02
         High          3.834937e+02
         Low           3.765193e+02
         Close         3.800725e+02
         Adj Close     3.800725e+02
         Volume        8.038476e+06
         dtype: float64
```

```
In [13]: google_stock['Close'].min()
```

```
Out[13]: 49.681865999999999
```

```
In [14]: google_stock.corr()
```

Out[14]:

	Open	High	Low	Close	Adj Close	Volume
Open	1.000000	0.999904	0.999845	0.999745	0.999745	-0.564258
High	0.999904	1.000000	0.999834	0.999868	0.999868	-0.562749
Low	0.999845	0.999834	1.000000	0.999899	0.999899	-0.567007
Close	0.999745	0.999868	0.999899	1.000000	1.000000	-0.564967
Adj Close	0.999745	0.999868	0.999899	1.000000	1.000000	-0.564967
Volume	-0.564258	-0.562749	-0.567007	-0.564967	-0.564967	1.000000

In [15]:
```
data = pd.read_csv('./fake_company.csv')
data
```

Out[15]:

	Year	Name	Department	Age	Salary
0	1990	Alice	HR	25	50000
1	1990	Bob	RD	30	48000
2	1990	Charlie	Admin	45	55000
3	1991	Alice	HR	26	52000
4	1991	Bob	RD	31	50000
5	1991	Charlie	Admin	46	60000
6	1992	Alice	Admin	27	60000
7	1992	Bob	RD	32	52000
8	1992	Charlie	Admin	28	62000

```
In [16]: data.groupby(['Year'])['Salary'].sum()

Out[16]: Year
         1990    153000
         1991    162000
         1992    174000
         Name: Salary, dtype: int64

In [17]: data.groupby(['Year'])['Salary'].mean()

Out[17]: Year
         1990    51000
         1991    54000
         1992    58000
         Name: Salary, dtype: int64
```

```
In [18]: data.groupby(['Name'])['Salary'].sum()

Out[18]: Name
         Alice      162000
         Bob        150000
         Charlie    177000
         Name: Salary, dtype: int64

In [19]: data.groupby(['Year', 'Department'])['Salary'].sum()

Out[19]: Year  Department
         1990  Admin          55000
               HR             50000
               RD             48000
         1991  Admin          60000
               HR             52000
               RD             50000
         1992  Admin         122000
               RD             52000
         Name: Salary, dtype: int64
```