In [1]:

```
## Why Pandas
# Pandas “is a fast, powerful, flexible and easy to use open source data analysis and manip
# built on top of the Python programming language.”
# Fast
# easy to use
# for data manipulation
```

In [3]:

```
import pandas as pd
import numpy as np
```

In [4]:

```
## Data types
# Series  1D , size Immutable but value Mutable
# Dataframe  2D   size and value are mutable
```

In [5]:

```
# create Series
```

In [90]:

```
data = ['a','b','c','d']
```

In [91]:

```
s = pd.Series(data, index=[4,8,12,16],dtype=np.str)
```

In [92]:

```
s
```

Out[92]:

```
4     a
8     b
12    c
16    d
dtype: object
```

In [69]:

```
s = pd.Series(data,index=[100,101,102,103])
# scikit-learn
```

In [11]:

```
s
```

Out[11]:

```
4     a
8     b
12    c
16    d
dtype: object
```

In [29]:

```
data = {'a' : 0., 'b' : 1., 'c' : 2.}
```

In [30]:

```
type(data)
```

Out[30]:

```
dict
```

In [31]:

```
s = pd.Series(data)
```

In [32]:

```
s
```

Out[32]:

```
a    0.0
b    1.0
c    2.0
dtype: float64
```

In [33]:

```
s
```

Out[33]:

```
a    0.0
b    1.0
c    2.0
dtype: float64
```

In [34]:

```
# indexing
print(s['c']) ## by index
print(s[1:3]) #
s[s>0] ## filter
```

```
2.0
b    1.0
c    2.0
dtype: float64
```

Out[34]:

```
b    1.0
c    2.0
dtype: float64
```

In [38]:

```
s[s>0]
```

Out[38]:

```
b    1.0
c    2.0
dtype: float64
```

In [ ]:

```

```

In [55]:

```
# Scalar
s = pd.Series(1, index=[100,200])
```

In [56]:

```
s
```

Out[56]:

```
100    1
200    1
dtype: int64
```

In [57]:

```
pd([s<1])
```

```
---------------------------------------------------------------------------
TypeError                                 Traceback (most recent call last)
<ipython-input-57-0c7e24751427> in <module>
----> 1 pd([s<1])

TypeError: 'module' object is not callable
```

In [156]:

```
## dataframe
```

In [68]:

```
s = pd.DataFrame([[2,2],[3,3]], index=['1st', '2nd'],columns=['age','height'])#, index=rang
```

In [69]:

```
s
```

Out[69]:

|  | age | height |
|---|---|---|
| **1st** | 2 | 2 |
| **2nd** | 3 | 3 |

In [157]:

```
s = pd.DataFrame([[2,2],[3,3]])# index=['1st', '2nd'])#, index=range(10))
```

In [158]:

```
s
```

Out[158]:

|  | 0 | 1 |
|---|---|---|
| **0** | 2 | 2 |
| **1** | 3 | 3 |

In [162]:

```
pd.DataFrame([[21,15],[4,3]], index=['1st', '2nd'])#, index=range(10))
```

Out[162]:

|  | 0 | 1 |
|---|---|---|
| **1st** | 21 | 15 |
| **2nd** | 4 | 3 |

In [73]:

```
pd.DataFrame([[2,2,3,3], [1,1], [3,3,3]],index=['student#'+str(i) for i in range(3)])
```

Out[73]:

|  | 0 | 1 | 2 | 3 |
|---|---|---|---|---|
| **student#0** | 2 | 2 | 3.0 | 3.0 |
| **student#1** | 1 | 1 | NaN | NaN |
| **student#2** | 3 | 3 | 3.0 | NaN |

In [95]:

```
s = pd.DataFrame([['mohammed','20'], ['ali',19]]) ## add index and add columns
```

In [96]:

```
s
```

Out[96]:

|  | 0 | 1 |
|---|---|---|
| **0** | mohammed | 20 |
| **1** | ali | 19 |

In [100]:

```
type(s[1][0])
```

Out[100]:

```
str
```

In [ ]:

```
## values only
```

In [ ]:

```

```

In [109]:

```
data = {'Name':['Tom', 'Jack', 'Steve', 'Ricky'],'Age':[28,34,29,42]} ## type !!
```

In [167]:

```
em = pd.DataFrame()
```

In [110]:

```
pd.DataFrame(data)
```

Out[110]:

| | Name | Age |
|---|---|---|
| **0** | Tom | 28 |
| **1** | Jack | 34 |
| **2** | Steve | 29 |
| **3** | Ricky | 42 |

In [159]:

```
filename = './test.txt'
dataset = np.loadtxt(filename)#, delimiter = ",")
print(dataset.shape)
print(dataset)
```

```
(3, 4)
[[70. 75. 60. 80.]
 [40. 60. 70. 90.]
 [33. 90. 55. 90.]]
```

In [160]:

```
df =pd.DataFrame(dataset,index=['s1','s2','s3'], columns=['test1','test2','test3','test4'])
```

In [157]:

```
new_s = [ [60], [90], [99], [85]]
df1 = pd.DataFrame(new_s)
```

In [158]:

```
df.append([60, 90, 99, 85])
```

Out[158]:

|  | test1 | test2 | test3 | test4 | 0 |
|---|---|---|---|---|---|
| **s1** | 70.0 | 75.0 | 60.0 | 80.0 | NaN |
| **s2** | 40.0 | 60.0 | 70.0 | 90.0 | NaN |
| **s3** | 33.0 | 90.0 | 55.0 | 90.0 | NaN |
| **0** | NaN | NaN | NaN | NaN | 60.0 |
| **1** | NaN | NaN | NaN | NaN | 90.0 |
| **2** | NaN | NaN | NaN | NaN | 99.0 |
| **3** | NaN | NaN | NaN | NaN | 85.0 |

In [136]:

```
df.iloc[2] ### rows
```

Out[136]:

```
test1    33.0
test2    90.0
test3    55.0
test4    90.0
Name: s3, dtype: float64
```

In [268]:

```
df
```

Out[268]:

|  | 0 | 1 | 2 | 3 |
|---|---|---|---|---|
| **0** | 10.0 | 4.0 | 5.0 | 20.0 |
| **1** | 2.0 | 3.0 | 4.0 | 12.0 |

In [ ]:

```
##  append two dataframes
```

In [161]:

```
df
```

Out[161]:

| | test1 | test2 | test3 | test4 |
|---|---|---|---|---|
| **s1** | 70.0 | 75.0 | 60.0 | 80.0 |
| **s2** | 40.0 | 60.0 | 70.0 | 90.0 |
| **s3** | 33.0 | 90.0 | 55.0 | 90.0 |

In [169]:

```
em = pd.DataFrame()
em.empty
```

Out[169]:

```
True
```

In [170]:

```
df.ndim
```

Out[170]:

```
2
```

In [177]:

```
df.head(1)
```

Out[177]:

| | test1 | test2 | test3 | test4 |
|---|---|---|---|---|
| **s1** | 70.0 | 75.0 | 60.0 | 80.0 |

axes :Returns a list of the row axis labels

dtype for series and dtypes for df:Returns the dtype of the object.

empty :Returns True if series is empty.

ndim :Returns the number of dimensions of the underlying data, by definition 1.

size :Returns the number of elements in the underlying data.

values :Returns the Series as ndarray.

head() :Returns the first n rows.

tail() :Returns the last n rows.

In [281]:

```
df
```

Out[281]:

|   | 0 | 1 | 2 | 3 |
|---|---|---|---|---|
| **0** | 10.0 | 4.0 | 5.0 | 20.0 |
| **1** | 2.0 | 3.0 | 4.0 | 12.0 |

In [ ]:

```

```

In [ ]:

```

```

In [45]:

```
patients = []
for i in range(1000):
    dic = { 'id':'p'+str(i) , 'height':np.random.randint(80) + 120 , 'weight':np.random.ran
    patients.append(dic)
```

In [53]:

```
print(type(patients))
print(type(patients[1]['height']))
```

```
<class 'list'>
<class 'int'>
```

In [54]:

```
df = pd.DataFrame(patients)
```

In [61]:

```
type(df['weight'])
```

Out[61]:

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

In [ ]:

```

```

In [ ]:

In [ ]:

In [ ]:

In [ ]:

In [ ]:

In [4]:

```
# COVID example
import requests ## To get data from a website
url = 'https://api.covid19api.com/summary'
r = requests.get(url)
json = r.json()
```

In [5]:

```
json.keys()
```

Out[5]:

```
dict_keys(['Message', 'Global', 'Countries', 'Date'])
```

In [17]:

```
type(json['Countries'][1])
```

Out[17]:

```
dict
```

In [ ]:

In [ ]:

In [ ]:

```

```

In [95]:

```
filename = './test.txt'
dataset = np.loadtxt(filename)
print(dataset.shape)
print(dataset)
```

```
(2, 3)
[[10.  4.  5.]
 [ 2.  3.  4.]]
```

In [103]:

```
filename = './test_comma.txt'
dataset = np.loadtxt(filename, delimiter = ",")
print(dataset.shape)
print(dataset)
```

```
(2, 4)
[[1. 4. 6. 6.]
 [7. 7. 7. 9.]]
```

In [115]:

```
np.savetxt('tmp.txt', dataset, delimiter = "," )##, fmt='%1.4e') ## fmt='%1.2f'
```

In [ ]:

```

```