https://www.facebook.com/hcmut.ml.iot.lab

Lesson 2

Working with Numpy and Pandas

ML IoT

HCMUT EE MACHINE LEARNING & IOT LAB

MSc. Nguyen Khanh Loi
Ho Chi Minh City University of Technology

4/2021

NUMPY

HCMUT EE Machine Learning & IoT Lab https://www.facebook.com/hcmut.ml.iot.lab

Numpy Install

```
pip install numpy
pip install numpy==version
```

```
>>> import numpy as np
>>> np.__version__
'1.16.5'
```

- **List is powerful**
  - collection of value
  - hold different types
  - change, add, remove
- **BUT for Data science**
  - Mathematical operations over collection

```
x = [12, 32, 4 , 10, 36, 5]
y = [6, 8, 4, 5, 6, 5]
z = x/y # error
```

HCMUT EE Machine Learning & IoT Lab https://www.facebook.com/hcmut.ml.iot.lab

Quick start

```
>>> import numpy as np
>>> arr =
np.array([1,2,3,4,5,6,7,8,9])
>>> arr
array([1, 2, 3, 4, 5, 6, 7, 8, 9])

>>> b_arr = arr.reshape((3,3))
>>> b_arr
array([[1, 2, 3],
       [4, 5, 6],
       [7, 8, 9]])

>>> b_arr * 2
array([[ 2, 4, 6],
       [ 8, 10, 12],
       [14, 16, 18]])
```

HCMUT EE Machine Learning & IoT Lab https://www.facebook.com/hcmut.ml.iot.lab

Python List vs NumPy Array

```
>>> plist = [1,2,3,4]
>>> plist + plist
[1,2,3,4,1,2,3,4]
```

```
>>> nparray = numpy.array([1,2,3,4])
>>> nparray + nparray
array([2,4,6,8])

>>>np.array([True, 1, 2]) + np.array([3, 4, False])
# True === 1
# False === 0
array([4, 5, 2])
```

HCMUT EE Machine Learning & IoT Lab https://www.facebook.com/hcmut.ml.iot.lab

Element type (dtype)

- NumPy arrays comprise elements of a single data type
- The type object is accessible through the .dtype attribute
- Array dtypes are usually inferred automatically

```
import numpy as np

A = np.array([[1, 2, 3], [3, 4, 5]])
print(A)
[[1 2 3]
 [3 4 5]]

A = np.array([[1.1, 2, 3], [3, 4, 5]]) # Array of floats
print(A)
[[1.1 2.  3. ]
 [3.  4.  5. ]]

A = np.array([[1, 2, 3], [3, 4, 5]], dtype = complex) # Array
of complex numbers
print(A)
[[1.+0.j 2.+0.j 3.+0.j]
 [3.+0.j 4.+0.j 5.+0.j]]
```

```
a = np.array([1,2,3,4])
>>> a.dtype
dtype('int64')

a = np.array([1, 2, 3, 4.0])
>>> a.dtype
dtype('float64')

a = np.array([1,2,3,4], dtype=np.float32)
>>> a.dtype
dtype('float32')
```

HCMUT EE Machine Learning & IoT Lab https://www.facebook.com/hcmut.ml.iot.lab

Array Creation

```
>>> import numpy as np
>>> a = np.array([2,3,4])
>>> a
array([2, 3, 4])

>>> a.dtype
dtype('int64')

>>> b = np.array([1.2, 3.5, 5.1])
>>> b.dtype
dtype('float64')

>>> b = np.array([(1.5,2,3), (4,5,6)])
>>> b
array([[ 1.5,  2. ,  3. ],
       [ 4. ,  5. ,  6. ]])
```

```
>>> np.zeros( (3,4) )
array([[ 0.,  0.,  0.,  0.],
       [ 0.,  0.,  0.,  0.],
       [ 0.,  0.,  0.,  0.]])

>>> np.ones( (2,3,4), dtype=np.int16 ) # dtype can also be specified
array([[[ 1, 1, 1, 1],
        [ 1, 1, 1, 1],
        [ 1, 1, 1, 1]],
       [[ 1, 1, 1, 1],
        [ 1, 1, 1, 1],
        [ 1, 1, 1, 1]]], dtype=int16)
>>> np.empty( (2,3) )# uninitialized, output may vary
array([[  3.73603959e-262,   6.02658058e-154,   6.55490914e-260],
       [  5.30498948e-313,   3.14673309e-307,   1.00000000e+000]])

>>> np.arange( 10, 30, 5 )
array([10, 15, 20, 25])

>>> np.arange( 0, 2, 0.3 ) # it accepts float arguments
array([ 0. ,  0.3,  0.6,  0.9,  1.2,  1.5,  1.8])

>>> np.linspace( 0, 2, 9 ) # 9 numbers from 0 to 2
array([ 0.  ,  0.25,  0.5 ,  0.75,  1.  ,  1.25,  1.5 ,  1.75,  2.
])
```

HCMUT EE Machine Learning & IoT Lab https://www.facebook.com/hcmut.ml.iot.lab

Load data from text file

test.txt

1	price	sell	vat%	revenue
2	1000	634.5	123.2	1200.6
3	12.5	54.2	31.5	2100
4	313.4	1.54	31.3	313

```
>>> np.loadtxt('test.txt', skiprows=1)
array([[1.0000e+03, 6.3450e+02, 1.2320e+02, 1.2006e+03],
       [1.2500e+01, 5.4200e+01, 3.1500e+01, 2.1000e+03],
       [3.1340e+02, 1.5400e+00, 3.1300e+01, 3.1300e+02]])

>>> np.loadtxt('test.txt', skiprows=1, dtype=np.int32)
array([[1000,  634,  123, 1200],
       [  12,   54,   31, 2100],
       [ 313,    1,   31,  313]])
```

HCMUT EE Machine Learning & IoT Lab https://www.facebook.com/hcmut.ml.iot.lab

Load data from text file

test.txt

```
price,   sell,    vat%,    revenue
1000,    634.5,   123.2,   1200.6
12.5,    54.2,    ,        2100
313.4,   1.54,    31.3,    313
```

```
>>> np.genfromtxt('test.txt', skip_header=1, filling_values=0,
                  delimiter=',')
array([[1.0000e+03, 6.3450e+02, 1.2320e+02, 1.2006e+03],
       [1.2500e+01, 5.4200e+01, 0.0000e+00, 2.1000e+03],
       [3.1340e+02, 1.5400e+00, 3.1300e+01, 3.1300e+02]])

>>> np.genfromtxt('test.txt', skip_header=1, filling_values=0,
                  delimiter=',', dtype=np.int32)
array([[1000,  634,  123, 1200],
       [  12,   54,    0, 2100],
       [ 313,    1,   31,  313]])
```

HCMUT EE Machine Learning & IoT Lab https://www.facebook.com/hcmut.ml.iot.lab

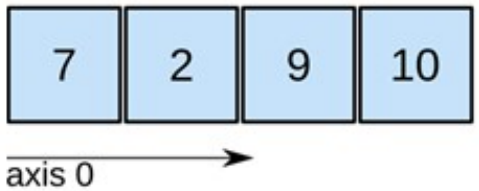
Array shape & axis

```
np.array([[[5.2, 3. , 4.5],[9.1, 0.1, 0.3]],
          [[5.2, 3. , 4.5],[9.1, 0.1, 0.3]],
          [[5.2, 3. , 4.5],[9.1, 0.1, 0.3]]])
```

```
np.array([[5.2, 3. , 4.5],
          [9.1, 0.1, 0.3]])
```

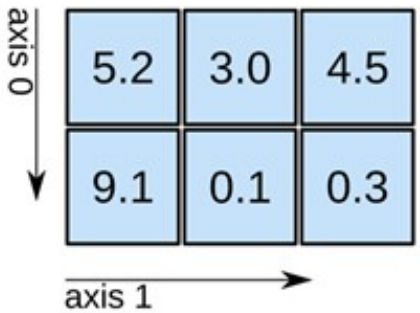
```
np.array([7, 2, 9, 10])
```

1D array



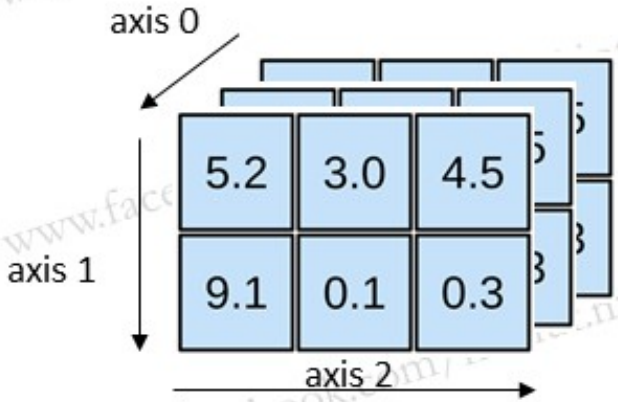
shape: (4,)

2D array



shape: (2, 3)

3D array



shape: (3, 2, 3)

reshape

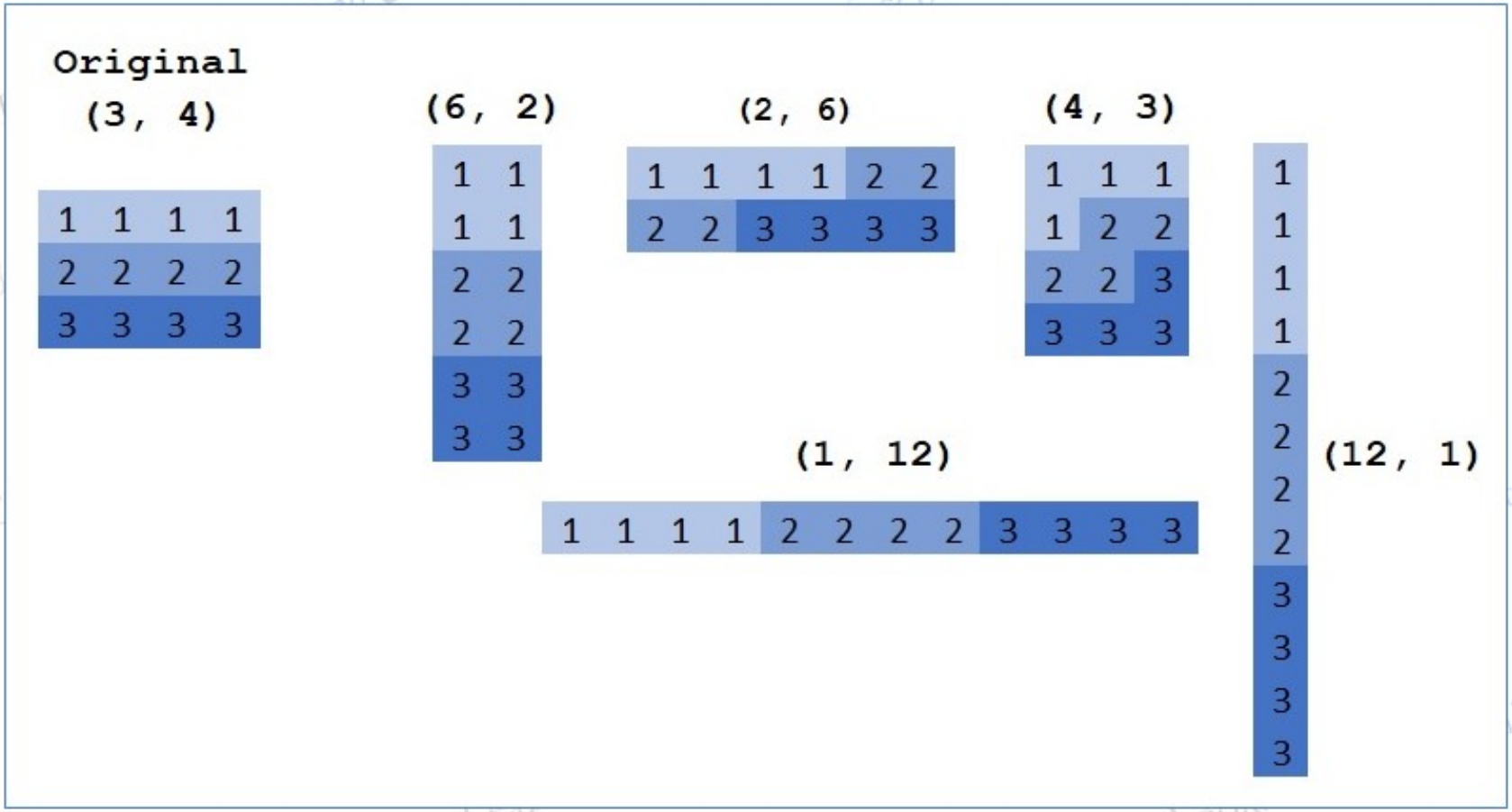
1	2	3	4	5	6	7	8	9	10	11	12

```
a = np.array(range(1,13))
b = a.reshape((3,4))
```

1	2	3	4
5	6	7	8
9	10	11	12

```
>>> a = np.array(range(1, 13))
>>> b = a.reshape((3,4))
>>> a
array([ 1,  2,  3,  4,  5,  6,  7,  8,  9, 10, 11, 12])
>>> b
array([[ 1,  2,  3,  4],
       [ 5,  6,  7,  8],
       [ 9, 10, 11, 12]])
```

reshape



HCMUT EE Machine Learning & IoT Lab https://www.facebook.com/hcmut.ml.iot.lab

Indexing

```python
# Python program for basic slicing.
import numpy as np

# Arrange elements from 0 to 19
a = np.arange(20)
print("\n Array is:\n ",a)

# a[start:stop:step]
print("\n a[-8:17:1]  = ",a[-8:17:1])

# The : operator means all elements till the end.
print("\n a[10:]  = ",a[10:])
```

```
Array is:
[ 0  1  2  3  4  5  6  7  8  9 10 11 12 13 14 15 16 17 18 19]

a[-8:17:1]  =  [12 13 14 15 16]

a[10:] = [10 11 12 13 14 15 16 17 18 19]
```

HCMUT EE Machine Learning & IoT Lab https://www.facebook.com/hcmut.ml.iot.lab

Indexing

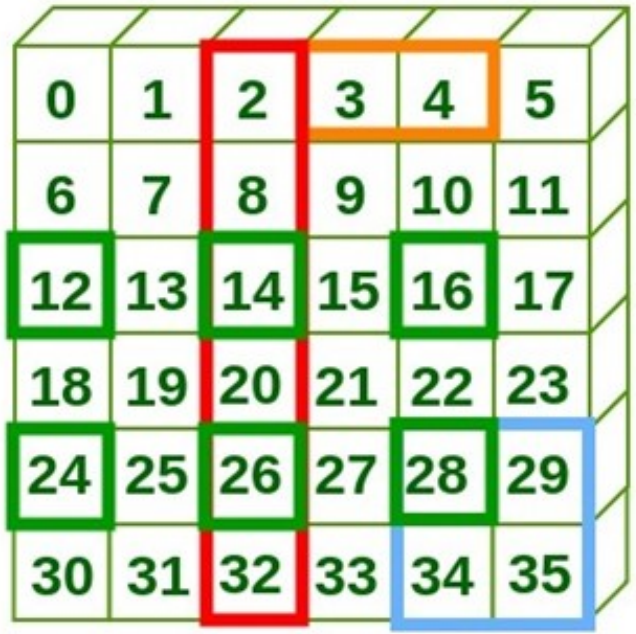
```
# A 3-Dimensional array
a = np.array([[0, 1, 2, 3, 4, 5],
              [6, 7, 8, 9, 10, 11],
              [12, 13, 14, 15, 16, 17],
              [18, 19, 20, 21, 22, 23],
              [24, 25, 26, 27, 28, 29],
              [30, 31, 32, 33, 34, 35]])
```

```
>>> a[0,3:5]
array( [3,4] )

>>> a[4:, 4:]
array( [ 28, 29],
       [ 34, 35] ] )

>>> a[ :, 2]
array( [2, 8, 14, 20, 26, 32] )

>>> a[2 : : 2, : : 2]
array( [ 12, 14, 16],
       [ 24, 26, 28] ] )
```



HCMUT EE Machine Learning & IoT Lab https://www.facebook.com/hcmut.ml.iot.lab

Iterating

```
>>> b = np.array([[ 0,  1,  2,  3],
                  [10, 11, 12, 13],
                  [20, 21, 22, 23],
                  [30, 31, 32, 33],
                  [40, 41, 42, 43]])
>>> for row in b:
...     print(row)
...
[0 1 2 3]
[10 11 12 13]
[20 21 22 23]
[30 31 32 33]
[40 41 42 43]

>>> for element in b.flat:
...     print(element)
...
0
1
2
3
10
11
...
```

HCMUT EE Machine Learning & IoT Lab https://www.facebook.com/hcmut.ml.iot.lab

Basic Operations

```
>>> a = np.array( [20,30,40,50] )
>>> b = np.arange( 4 )
>>> b
array([0, 1, 2, 3])

>>> c = a-b
>>> c
array([20, 29, 38, 47])

>>> b**2
array([0, 1, 4, 9])

>>> 10*np.sin(a)
array([ 9.12945251, -9.88031624,  7.4511316 , -2.62374854])

>>> a < 35
array([ True, True, False, False])

>>> a[a < 35]
array([20, 30])

>>> a[a > 30] = 0
>>> a
array([20, 30,  0,  0])
```

HCMUT EE Machine Learning & IoT Lab https://www.facebook.com/hcmut.ml.iot.lab

Basic Operations

```
>>> A = np.array( [[1,1],
...                [0,1]] )
>>> B = np.array( [[2,0],
...                [3,4]] )
>>> A * B    # elementwise product
array([[2, 0],
       [0, 4]])

>>> A @ B    # matrix product
array([[5, 4],
       [3, 4]])

>>> A.dot(B) # another matrix product
array([[5, 4],
       [3, 4]])
```

$$\begin{bmatrix} a_1 & a_2 & a_3 \\ a_4 & a_5 & a_6 \\ a_7 & a_8 & a_9 \end{bmatrix} \begin{bmatrix} b_1 & b_2 & b_3 \\ b_4 & b_5 & b_6 \\ b_7 & b_8 & b_9 \end{bmatrix} = \begin{bmatrix} c_1 & c_2 & c_3 \\ c_4 & c_5 & c_6 \\ c_7 & c_8 & c_9 \end{bmatrix}$$

```
>>> a = np.random.random((2,3))
>>> a
array([[ 0.18626021,  0.34556073,  0.39676747],
       [ 0.53881673,  0.41919451,  0.6852195 ]])

>>> a.sum()
2.5718191614547998

>>> a.min()
0.1862602113776709

>>> a.max()
0.6852195003967595
```

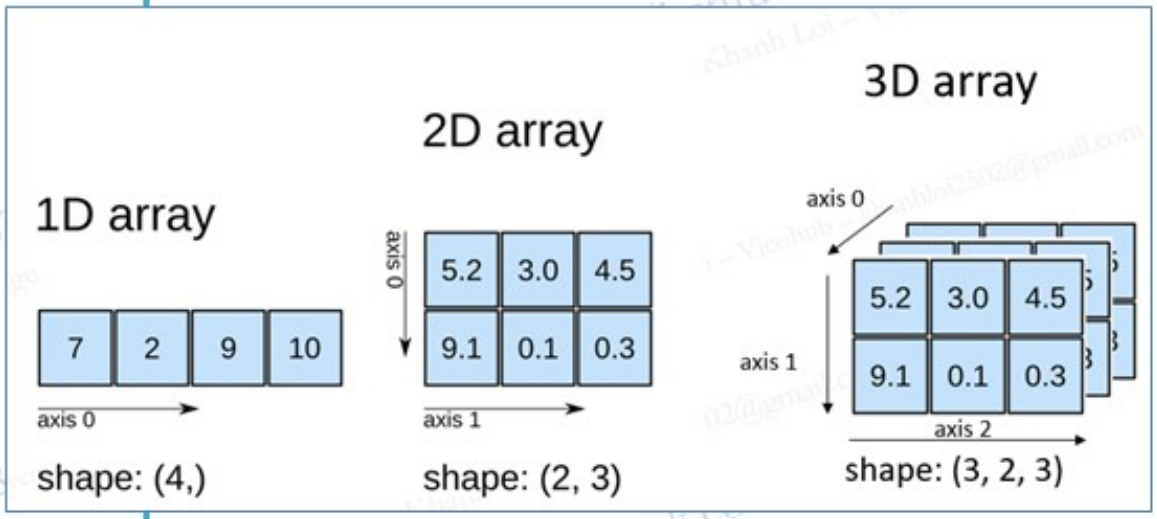
HCMUT EE Machine Learning & IoT Lab https://www.facebook.com/hcmut.ml.iot.lab

Basic Operations

```
>>> b = np.arange(12).reshape(3,4)
>>> b
array([[ 0,  1,  2,  3],
       [ 4,  5,  6,  7],
       [ 8,  9, 10, 11]])
>>>
>>> b.sum(axis=0)
# sum of each column
array([12, 15, 18, 21])

>>>
>>> b.min(axis=1)
# min of each row
array([0, 4, 8])

>>>
>>> b.cumsum(axis=1)
# cumulative sum along each row
array([[ 0,  1,  3,  6],
       [ 4,  9, 15, 22],
       [ 8, 17, 27, 38]])
```



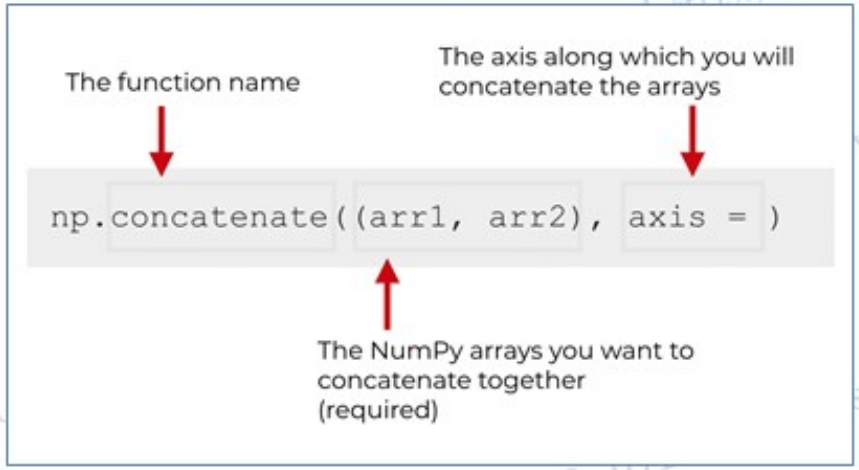
Copy

```
>>> a = np.arange(12)
>>> a
array([ 0,  1,  2,  3,  4,  5,  6,  7,  8,  9, 10, 11])
>>> b = a
>>> b[2:5] = 2
>>> b
array([ 0,  1,  2,  2,  2,  5,  6,  7,  8,  9, 10, 11])
>>> a
array([ 0,  1,  2,  2,  2,  5,  6,  7,  8,  9, 10, 11])
>>> b is a
True

>>> a = np.arange(12)
>>> b = a.copy()
>>> b[2:5] = 2
>>> b
array([ 0,  1,  2,  2,  2,  5,  6,  7,  8,  9, 10, 11])
>>> a
array([ 0,  1,  2,  3,  4,  5,  6,  7,  8,  9, 10, 11])
>>> b is a
False
```

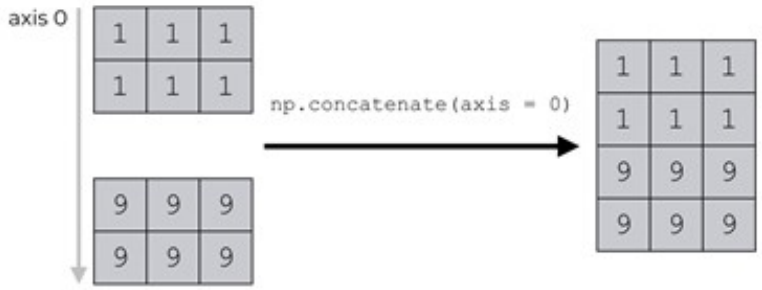
HCMUT EE Machine Learning & IoT Lab https://www.facebook.com/hcmut.ml.iot.lab

Concatenate

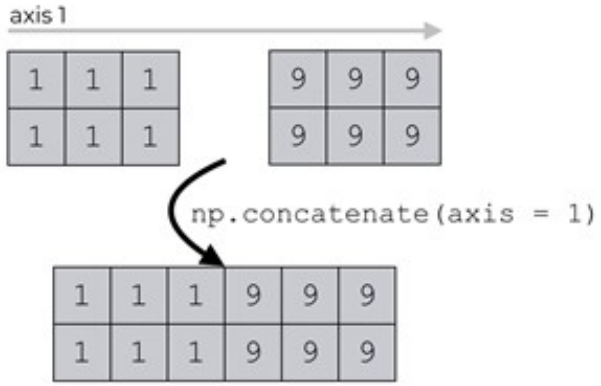



```
integer_data = np.array([[1,1,1],[1,1,1]])
float_data = np.array([[9,9,9],[9,9,9]])
np.concatenate([integer_data, float_data])
```

Setting `axis=0` concatenates along the row axis



Setting `axis=1` concatenates along the column axis



Vertical Stacking np.vstack()

```
import numpy as np

x = np.array([1, 2, 3, 4, 5])
y = np.array([16, 17, 18, 19, 20])
print('First array:\n', x)
print('Second array:\n', y)
print('Vertically stacked array:\n', np.vstack((x, y)))

First array:
 [1 2 3 4 5]
Second array:
 [16 17 18 19 20]
Vertically stacked array:
 [[ 1  2  3  4  5]
 [16 17 18 19 20]]
```

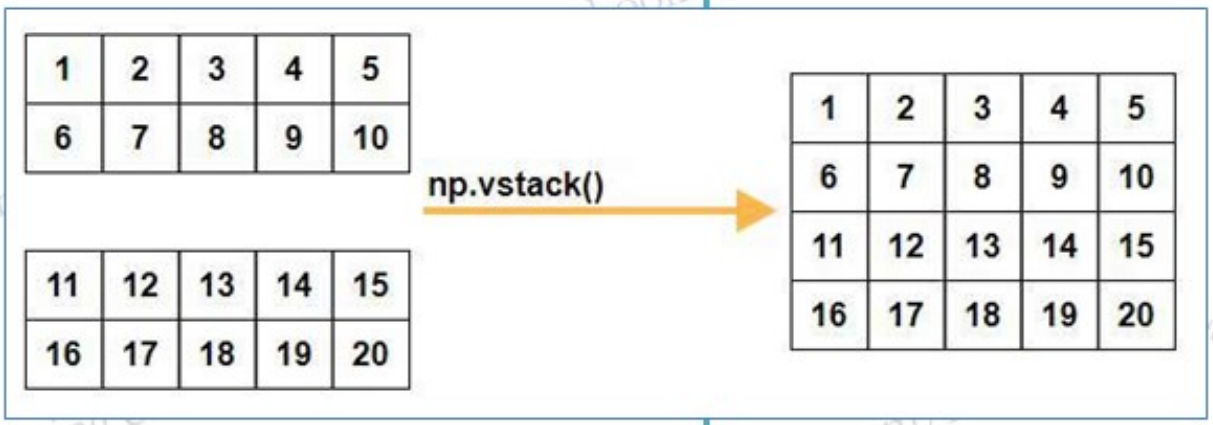
Vertical Stacking np.vstack()

```
import numpy as np

x = np.array([[1, 2, 3, 4, 5], [6, 7, 8, 9, 10]])
y = np.array([[11, 12, 13, 14, 15], [16, 17, 18, 19, 20]])
print('First array:\n', x)
print('Second array:\n', y)
print('Vertically stacked array:\n', np.vstack((x, y)))
print('Concatenated 2-D array:\n', np.concatenate((x,y),
axis=0))
```

```
First array:
 [[ 1  2  3  4  5]
 [ 6  7  8  9 10]]
Second array:
 [[11 12 13 14 15]
 [16 17 18 19 20]]
Vertically stacked array:
 [[ 1  2  3  4  5]
 [ 6  7  8  9 10]
 [11 12 13 14 15]
 [16 17 18 19 20]]
Concatenated 2-D array:
 [[ 1  2  3  4  5]
 [ 6  7  8  9 10]
 [11 12 13 14 15]
 [16 17 18 19 20]]
```

How about hstack?



PANDAS

HCMUT EE Machine Learning & IoT Lab https://www.facebook.com/hcmut.ml.iot.lab

Numpy pandas

```
pip install pandas
pip install pandas==version
```

```
>>> import pandas as pd
>>> pd.__version__
'0.25.1'
```

Pandas provides a couple of very useful datatypes, Series and DataFrame:

- **Series** represents 1D data, like time series, calendars, the output of one-variable functions, etc. A little like list and dict of Python, but more powerful.
- **DataFrame** represents 2D data, like a column-separated-values (CSV) file, a microarray, a database table, a matrix, etc. A DataFrame is composed of multiple Series, i.e., each column of a DataFrame is a Series.

HCMUT EE Machine Learning & IoT Lab https://www.facebook.com/hcmut.ml.iot.lab

DataFrame: creation pandas

DataFrame is a 2D labeled data structure with columns of potentially different types. It accepts many different kinds of input:

- Dict of 1D ndarrays, lists, dicts, or Series
- 2-D numpy.ndarray
- Structured or record ndarray
- A Series
- Another DataFrame

```
>>> import pandas as pd
>>> d = {'one':pd.Series([1., 2., 3.], index=['a', 'b', 'c']),
'two':pd.Series([1., 2., 3., 4.], index=['a', 'b', 'c', 'd'])}
>>> df = pd.DataFrame(d)
>>> df
   one  two
a  1.0  1.0
b  2.0  2.0
c  3.0  3.0
d  NaN  4.0
```

HCMUT EE Machine Learning & IoT Lab https://www.facebook.com/hcmut.ml.iot.lab

DataFrame: creation pandas

```
>>> d1 = {'one':[1., 2., 3., 4.], 'two':[4., 5., 6., 7.]}
>>> df1 = pd.DataFrame(d1)
```

```
>>> df1
   one  two
0  1.0  4.0
1  2.0  5.0
2  3.0  6.0
3  4.0  7.0
```

```
>>> d2 = [(1, 2., 'Hello'), (2, 3, True)]
>>> df2 = pd.DataFrame(d2)
```

```
>>> df2
   0    1      2
0  1  2.0  Hello
1  2  3.0   True
```

```
>>> d3 = [{'a': 1, 'b': 2}, {'a': 5, 'b': 10, 'c': 100}]
>>> df3 = pd.DataFrame(d3)
```

```
>>> df3
   a   b      c
0  1   2    NaN
1  5  10  100.0
```

```
>>> df4 = pd.DataFrame(d3, index=['row1', 'row2'])
```

```
>>> df4
      a   b      c
row1  1   2    NaN
row2  5  10  100.0
```

```
>>> df5 = pd.DataFrame(d3, columns=['a', 'b'])
```

```
>>> df5
   a   b
0  1   2
1  5  10
```

HCMUT EE Machine Learning & IoT Lab https://www.facebook.com/hcmut.ml.iot.lab

DataFrame: attribute

Row and column labels can be accessed by the index and column attributes, the hape, which describes the dimension of the table .

```
>>> df = pd.DataFrame.from_dict(dict([('A', [1, 2, 3, 4]),
                                      ('B', [4, 5, 6, 7]),
                                      ('C', [4, 5, 6, 8])]))
>>> df
   A  B  C
0  1  4  4
1  2  5  5
2  3  6  6
3  4  7  8
>>> df.index
RangeIndex(start=0, stop=4, step=1)

>>> df.columns
Index(['A', 'B', 'C'], dtype='object')

>>> df.shape
(4, 3)
```

DataFrame: load file

Load from text file (i.e., cvs)

```
df1 = pd.read_csv(file_name, [index_col=], [header=], [skiprows=])
```

```
>>> import pandas as pd
>>> df1 = pd.read_csv('iris.csv')
>>> df1.head()
   sepal_length  sepal_width  petal_length  petal_width variety
0           5.1          3.5           1.4          0.2  Setosa
1           4.9          3.0           1.4          0.2  Setosa
2           4.7          3.2           1.3          0.2  Setosa
3           4.6          3.1           1.5          0.2  Setosa
4           5.0          3.6           1.4          0.2  Setosa

>>> df1 = pd.read_csv('iris.csv', index_col = 'sepal_length')
>>> df1.head()
              sepal_width  petal_length  petal_width variety
sepal_length
5.1                   3.5           1.4          0.2  Setosa
4.9                   3.0           1.4          0.2  Setosa
4.7                   3.2           1.3          0.2  Setosa
4.6                   3.1           1.5          0.2  Setosa
5.0                   3.6           1.4          0.2  Setosa
```

DataFrame: extraction

The elementary extraction methods

Operation	Syntax	Result
Select column	df[col]	Series
Select row by label	df.loc[label]	Series
Select row by integer location	df.iloc[loc]	Series
Slice rows	df[5:10]	DataFrame
Select rows by boolean vector	df[bool_vec]	DataFrame

HCMUT EE Machine Learning & IoT Lab https://www.facebook.com/hcmut.ml.iot.lab

DataFrame: extraction

```
>>> df1 = pd.read_csv('iris.csv', index_col='variety')
>>> df1.head()
         sepal_length  sepal_width  petal_length  petal_width
variety
Setosa            5.1          3.5           1.4          0.2
Setosa            4.9          3.0           1.4          0.2
Setosa            4.7          3.2           1.3          0.2
Setosa            4.6          3.1           1.5          0.2
Setosa            5.0          3.6           1.4          0.2
>>> df1['sepal_width']
variety
Setosa       3.5
Setosa       3.0
Setosa       3.2
Setosa       3.1
Setosa       3.6
            ...
Virginica    3.0
Virginica    2.5
Virginica    3.0
Virginica    3.4
Virginica    3.0
Name: sepal_width, Length: 150, dtype: float64
>>> df1[1:10]
         sepal_length  sepal_width  petal_length  petal_width
variety
Setosa            4.9          3.0           1.4          0.2
Setosa            4.7          3.2           1.3          0.2
Setosa            4.6          3.1           1.5          0.2
Setosa            5.0          3.6           1.4          0.2
Setosa            5.4          3.9           1.7          0.4
```

```
>>> df1.loc['Setosa']
         sepal_length  sepal_width  petal_length  petal_width
variety
Setosa            5.1          3.5           1.4          0.2
Setosa            4.9          3.0           1.4          0.2
Setosa            4.7          3.2           1.3          0.2
Setosa            4.6          3.1           1.5          0.2
Setosa            5.0          3.6           1.4          0.2
...

>>> df1.iloc[:5]
         sepal_length  sepal_width  petal_length  petal_width
variety
Setosa            5.1          3.5           1.4          0.2
Setosa            4.9          3.0           1.4          0.2
Setosa            4.7          3.2           1.3          0.2
Setosa            4.6          3.1           1.5          0.2
Setosa            5.0          3.6           1.4          0.2

>>> df1[:5]
         sepal_length  sepal_width  petal_length  petal_width
variety
Setosa            5.1          3.5           1.4          0.2
Setosa            4.9          3.0           1.4          0.2
Setosa            4.7          3.2           1.3          0.2
Setosa            4.6          3.1           1.5          0.2
Setosa            5.0          3.6           1.4          0.2
```

HCMUT EE Machine Learning & IoT Lab https://www.facebook.com/hcmut.ml.iot.lab

DataFrame: extraction

```
>>> df1 = pd.read_csv('iris.csv', index_col='variety')
>>> df1.head()
         sepal_length  sepal_width  petal_length  petal_width
variety
Setosa            5.1          3.5           1.4          0.2
Setosa            4.9          3.0           1.4          0.2
Setosa            4.7          3.2           1.3          0.2
Setosa            4.6          3.1           1.5          0.2
Setosa            5.0          3.6           1.4          0.2
>>> boolean_vector = df1['sepal_length'] > 5
>>> boolean_vector
variety
Setosa       True
Setosa      False
Setosa      False
Setosa      False
Setosa      False
            ...
Virginica    True
Virginica    True
Virginica    True
Virginica    True
Virginica    True
Name: sepal_length, Length: 150, dtype: bool
```

```
>>> df1[boolean_vector]
            sepal_length  sepal_width  petal_length  petal_width
variety
Setosa               5.1          3.5           1.4          0.2
Setosa               5.4          3.9           1.7          0.4
Setosa               5.4          3.7           1.5          0.2
Setosa               5.8          4.0           1.2          0.2
Setosa               5.7          4.4           1.5          0.4
...                  ...          ...           ...          ...
Virginica            6.7          3.0           5.2          2.3
Virginica            6.3          2.5           5.0          1.9
Virginica            6.5          3.0           5.2          2.0
Virginica            6.2          3.4           5.4          2.3
Virginica            5.9          3.0           5.1          1.8

[118 rows x 4 columns]
>>> df1.sepal_length
variety
Setosa       5.1
Setosa       4.9
Setosa       4.7
Setosa       4.6
Setosa       5.0
            ...
Virginica    6.7
Virginica    6.3
Virginica    6.5
Virginica    6.2
Virginica    5.9
Name: sepal_length, Length: 150, dtype: float64
```

HCMUT EE Machine Learning & IoT Lab https://www.facebook.com/hcmut.ml.iot.lab

DataFrame: operators

Column addition, deletion

```
>>> # add column to the end
... df1['sepal_petal'] = df1['sepal_length'] - df1.petal_length
>>> df1.head()
         sepal_length  sepal_width  petal_length  petal_width  sepal_petal
variety
Setosa            5.1          3.5           1.4          0.2          3.7
Setosa            4.9          3.0           1.4          0.2          3.5
Setosa            4.7          3.2           1.3          0.2          3.4
Setosa            4.6          3.1           1.5          0.2          3.1
Setosa            5.0          3.6           1.4          0.2          3.6

>>> # delete a column
... del(df1['sepal_petal'])
>>> df1
           sepal_length  sepal_width  petal_length  petal_width
variety
Setosa              5.1          3.5           1.4          0.2
Setosa              4.9          3.0           1.4          0.2
...                 ...          ...           ...          ...
Virginica           6.2          3.4           5.4          2.3
Virginica           5.9          3.0           5.1          1.8

[150 rows x 4 columns]
```

```
>>> df1.pop('sepal_length')
variety
Setosa       5.1
Setosa       4.9
             ...
Name: sepal_length, Length: 150, dtype: float64
>>> df1
           sepal_width  petal_length  petal_width
variety
Setosa             3.5           1.4          0.2
Setosa             3.0           1.4          0.2
...                ...           ...          ...
Virginica          3.0           5.2          2.3
Virginica          2.5           5.0          1.9

[150 rows x 3 columns]
```

HCMUT EE Machine Learning & IoT Lab https://www.facebook.com/hcmut.ml.iot.lab

Assigning New Columns

The assign() method that allows you to easily create new columns that are potentially derived from existing columns. (return a copy of data)

```
>>> df1.assign(ratio=df1['sepal_width']/df1['petal_length']).head()
         sepal_width  petal_length  petal_width     ratio
variety
Setosa           3.5           1.4          0.2  2.500000
Setosa           3.0           1.4          0.2  2.142857
Setosa           3.2           1.3          0.2  2.461538
Setosa           3.1           1.5          0.2  2.066667
Setosa           3.6           1.4          0.2  2.571429
>>> df1.assign(ratio = lambda x:
       x['sepal_width']/x['petal_length']).head()
         sepal_width  petal_length  petal_width     ratio
variety
Setosa           3.5           1.4          0.2  2.500000
Setosa           3.0           1.4          0.2  2.142857
Setosa           3.2           1.3          0.2  2.461538
Setosa           3.1           1.5          0.2  2.066667
Setosa           3.6           1.4          0.2  2.571429
```

HCMUT EE Machine Learning & IoT Lab https://www.facebook.com/hcmut.ml.iot.lab

For loop

`DataFrame.iterrows` is a generator which **`yields`** both the index and row (as a Series):

```
>>> import pandas as pd
>>> import numpy as np
>>> df1 = pd.read_csv('iris.csv')
>>> df1.head()
   sepal_length  sepal_width  petal_length  petal_width variety
0           5.1          3.5           1.4          0.2  Setosa
1           4.9          3.0           1.4          0.2  Setosa
2           4.7          3.2           1.3          0.2  Setosa
3           4.6          3.1           1.5          0.2  Setosa
4           5.0          3.6           1.4          0.2  Setosa
>>> for idx, row in df1.iterrows():
...   print(idx)
...   for item in row:
...     print('Item: ', item)
...   break
...
0
Item:  5.1
Item:  3.5
Item:  1.4
Item:  0.2
Item:  Setosa
```

HCMUT EE Machine Learning & IoT Lab https://www.facebook.com/hcmut.ml.iot.lab

Apply function

There are different ways to apply a function to each row or column in DataFrame. We will learn about various ways in this post. Let's create a small dataframe first and see that.

Method 1: Applying lambda function to each row/column.

```
>>> # import pandas and numpy library
... import pandas as pd
>>> import numpy as np
>>>
... # list of tuples
... matrix = [(1,2,3,4),
...            (5,6,7,8,),
...            (9,10,11,12),
...            (13,14,15,16)
...          ]
>>>
... # Create a Dataframe object
... df = pd.DataFrame(matrix, columns = list('abcd'))
>>>
... # Output
... df
    a   b   c   d
0   1   2   3   4
1   5   6   7   8
2   9  10  11  12
3  13  14  15  16
```

```
>>>
... # Applying a lambda function to each
... # column which will add 10 to the value
... new_df = df.apply(lambda x : x + 10)
>>>
... # Output
... new_df
    a   b   c   d
0  11  12  13  14
1  15  16  17  18
2  19  20  21  22
3  23  24  25  26
```

```
>>> # Applying a lambda function to each
... # row which will add 5 to the value
... new_df = df.apply(lambda x: x + 5, axis
= 1)
>>>
... # Output
... new_df
    a   b   c   d
0   6   7   8   9
1  10  11  12  13
2  14  15  16  17
3  18  19  20  21
```

HCMUT EE Machine Learning & IoT Lab https://www.facebook.com/hcmut.ml.iot.lab

Apply function

Method 2: Applying user defined function to each row/column

```
>>> # import pandas and numpy library
... import pandas as pd
>>> import numpy as np
>>>
... # list of tuples
... matrix = [(1,2,3,4),
...           (5,6,7,8,),
...           (9,10,11,12),
...           (13,14,15,16)
...          ]
>>>
... # Create a Dataframe object
... df = pd.DataFrame(matrix, columns = list('abcd'))
>>>
... # Output
... df
    a   b   c   d
0   1   2   3   4
1   5   6   7   8
2   9  10  11  12
3  13  14  15  16

>>> # function to returns x*x
... def squareData(x):
...     return x * x
...
```

```
>>> # Applying a user defined function to
... # each column that will square the given
... # value
... new_df = df.apply(squareData)
>>>
... # Output
... new_df
     a    b    c    d
0    1    4    9   16
1   25   36   49   64
2   81  100  121  144
3  169  196  225  256
```

```
>>> # Applying a user defined function
... # to each row that will square the given
value
... new_df = df.apply(squareData, axis = 1)
>>>
... # Output
... new_df
     a    b    c    d
0    1    4    9   16
1   25   36   49   64
2   81  100  121  144
3  169  196  225  256
```

HCMUT EE Machine Learning & IoT Lab https://www.facebook.com/hcmut.ml.iot.lab

Apply function

Method 3: Applying numpy function to each row/column

```
>>> # import pandas and numpy library
... import pandas as pd
>>> import numpy as np
>>>
... # list of tuples
... matrix = [(1,2,3,4),
...           (5,6,7,8,),
...           (9,10,11,12),
...           (13,14,15,16)
...          ]
>>>
... # Create a Dataframe object
... df = pd.DataFrame(matrix, columns = list('abcd'))
>>>
... # Output
... df
    a   b   c   d
0   1   2   3   4
1   5   6   7   8
2   9  10  11  12
3  13  14  15  16
```

```
>>> # Applying a numpy function to each
... # column by squaring each value
... new_df = df.apply(np.square)
>>>
... # Output
... new_df
     a    b    c    d
0    1    4    9   16
1   25   36   49   64
2   81  100  121  144
3  169  196  225  256
```

```
>>> # Applying a user defined function
... # to each row that will square the given
value
... new_df = df.apply(np.square, axis = 1)
>>>
... # Output
... new_df
     a    b    c    d
0    1    4    9   16
1   25   36   49   64
2   81  100  121  144
3  169  196  225  256
```

HCMUT EE Machine Learning & IoT Lab https://www.facebook.com/hcmut.ml.iot.lab

Apply function

Method 4: Applying a Reducing function to each row/column

A Reducing function will take row or column as series and returns either a series of same size as that of input row/column or it will return a single variable depending upon the function we use.

```
>>> # import pandas and numpy library
... import pandas as pd
>>> import numpy as np
>>>
... # list of tuples
... matrix = [(1,2,3,4),
...           (5,6,7,8,),
...           (9,10,11,12),
...           (13,14,15,16)
...          ]
>>>
... # Create a Dataframe object
... df = pd.DataFrame(matrix, columns = list('abcd'))
>>>
... # Output
... df
    a   b   c   d
0   1   2   3   4
1   5   6   7   8
2   9  10  11  12
3  13  14  15  16
```

```
>>> # Applying a numpy function to get the sum
... # of all values in each column
... new_df = df.apply(np.sum)
>>>
... # Output
... new_df
a    28
b    32
c    36
d    40
dtype: int64
```

```
>>> # Applying a numpy function to get t
... # he sum of all values in each row
... new_df = df.apply(np.sum, axis = 1)
>>>
... # Output
... new_df
0    10
1    26
2    42
3    58
dtype: int64
```

DataFrame: statistics

- The statistical functions such as: `min()`, `max()`, `mean()`, `median()`, `sum()`, `cumsum()`, `var()`, `std()`… also work on DataFrame.

- Correlation Matrix Of Values: df.corr()

```
>>> df1 = pd.read_csv('iris.csv', index_col='variety')
>>> df1.head()
         sepal_length  sepal_width  petal_length  petal_width
variety
Setosa            5.1          3.5           1.4          0.2
Setosa            4.9          3.0           1.4          0.2
Setosa            4.7          3.2           1.3          0.2
Setosa            4.6          3.1           1.5          0.2
Setosa            5.0          3.6           1.4          0.2
>>> df1.corr()
              sepal_length  sepal_width  petal_length  petal_width
sepal_length      1.000000    -0.117570      0.871754     0.817941
sepal_width      -0.117570     1.000000     -0.428440    -0.366126
petal_length      0.871754    -0.428440      1.000000     0.962865
petal_width       0.817941    -0.366126      0.962865     1.000000
```

HCMUT EE Machine Learning & IoT Lab https://www.facebook.com/hcmut.ml.iot.lab

DataFrame – GroupBy

```
obj.groupby('key')
obj.groupby(['key1','key2'])
obj.groupby(key,axis=1)
```

```
>>> #import the pandas library
... import pandas as pd
>>>
>>> ipl_data = {'Team': ['Riders', 'Riders', 'Devils', 'Devils', 'Kings',
...     'kings', 'Kings', 'Kings', 'Riders', 'Royals', 'Royals', 'Riders'],
...     'Rank': [1, 2, 2, 3, 3,4 ,1 ,1,2 , 4,1,2],
...     'Year': [2014,2015,2014,2015,2014,2015,2016,2017,2016,2014,2015,2017],
...     'Points':[876,789,863,673,741,812,756,788,694,701,804,690]}
>>> df = pd.DataFrame(ipl_data)
>>> df
      Team  Rank  Year  Points
0   Riders     1  2014     876
1   Riders     2  2015     789
2   Devils     2  2014     863
3   Devils     3  2015     673
4    Kings     3  2014     741
5    kings     4  2015     812
6    Kings     1  2016     756
7    Kings     1  2017     788
8   Riders     2  2016     694
9   Royals     4  2014     701
10  Royals     1  2015     804
11  Riders     2  2017     690
```

```
>>> df.groupby('Team').groups
{
      'Devils': Int64Index([2, 3], dtype='int64'),
      'Kings': Int64Index([4, 6, 7], dtype='int64'),
      'Riders': Int64Index([0, 1, 8, 11], dtype='int64'),
      'Royals': Int64Index([9, 10], dtype='int64'),
      'kings': Int64Index([5], dtype='int64')
}
```

HCMUT EE Machine Learning & IoT Lab https://www.facebook.com/hcmut.ml.iot.lab

DataFrame – GroupBy

```
obj.groupby('key')
obj.groupby(['key1','key2'])
obj.groupby(key,axis=1)
```

Iterating through Groups

```
>>> #import the pandas library
... import pandas as pd
>>>
>>> ipl_data = {'Team': ['Riders', 'Riders', 'Devils', 'Devils', 'Kings',
...     'kings', 'Kings', 'Kings', 'Riders', 'Royals', 'Royals', 'Riders'],
...     'Rank': [1, 2, 2, 3, 3,4 ,1 ,1,2 , 4,1,2],
...     'Year':
[2014,2015,2014,2015,2014,2015,2016,2017,2016,2014,2015,2017],
...     'Points':[876,789,863,673,741,812,756,788,694,701,804,690]}
>>> df = pd.DataFrame(ipl_data)
>>> df
      Team  Rank  Year  Points
0   Riders     1  2014     876
1   Riders     2  2015     789
2   Devils     2  2014     863
3   Devils     3  2015     673
4    Kings     3  2014     741
5    kings     4  2015     812
6    Kings     1  2016     756
7    Kings     1  2017     788
8   Riders     2  2016     694
9   Royals     4  2014     701
10  Royals     1  2015     804
11  Riders     2  2017     690
```

```
>>> grouped = df.groupby('Year')
>>> for name,group in grouped:
...     print(name)
...     print(group)
...
2014
     Team  Rank  Year  Points
0  Riders     1  2014     876
2  Devils     2  2014     863
4   Kings     3  2014     741
9  Royals     4  2014     701
2015
      Team  Rank  Year  Points
1   Riders     2  2015     789
3   Devils     3  2015     673
5    kings     4  2015     812
10  Royals     1  2015     804
2016
     Team  Rank  Year  Points
6   Kings     1  2016     756
8  Riders     2  2016     694
2017
      Team  Rank  Year  Points
7    Kings     1  2017     788
11  Riders     2  2017     690
```

HCMUT EE Machine Learning & IoT Lab https://www.facebook.com/hcmut.ml.iot.lab

DataFrame – GroupBy

```
obj.groupby('key')
obj.groupby(['key1','key2'])
obj.groupby(key,axis=1)
```

Using the **get_group()** method, we can select a single group.

```
>>> #import the pandas library
... import pandas as pd
>>>
>>> ipl_data = {'Team': ['Riders', 'Riders', 'Devils', 'Devils', 'Kings',
...     'kings', 'Kings', 'Kings', 'Riders', 'Royals', 'Royals', 'Riders'],
...     'Rank': [1, 2, 2, 3, 3,4 ,1 ,1,2 , 4,1,2],
...     'Year': [2014,2015,2014,2015,2014,2015,2016,2017,2016,2014,2015,2017],
...     'Points':[876,789,863,673,741,812,756,788,694,701,804,690]}
>>> df = pd.DataFrame(ipl_data)
>>> df
      Team  Rank  Year  Points
0   Riders     1  2014     876
1   Riders     2  2015     789
2   Devils     2  2014     863
3   Devils     3  2015     673
4    Kings     3  2014     741
5    kings     4  2015     812
6    Kings     1  2016     756
7    Kings     1  2017     788
8   Riders     2  2016     694
9   Royals     4  2014     701
10  Royals     1  2015     804
11  Riders     2  2017     690
```

```
>>> grouped = df.groupby('Year')
>>> grouped.get_group(2014)
     Team  Rank  Year  Points
0  Riders     1  2014     876
2  Devils     2  2014     863
4   Kings     3  2014     741
9  Royals     4  2014     701
```

HCMUT EE Machine Learning & IoT Lab https://www.facebook.com/hcmut.ml.iot.lab

DataFrame – GroupBy

```
obj.groupby('key')
obj.groupby(['key1','key2'])
obj.groupby(key,axis=1)
```

Aggregations

```
>>> #import the pandas library
... import pandas as pd
>>>
>>> ipl_data = {'Team': ['Riders', 'Riders', 'Devils', 'Devils', 'Kings',
...     'kings', 'Kings', 'Kings', 'Riders', 'Royals', 'Royals', 'Riders'],
...     'Rank': [1, 2, 2, 3, 3,4 ,1 ,1,2 , 4,1,2],
...     'Year': [2014,2015,2014,2015,2014,2015,2016,2017,2016,2014,2015,2017],
...     'Points':[876,789,863,673,741,812,756,788,694,701,804,690]}
>>> df = pd.DataFrame(ipl_data)
>>> df
      Team  Rank  Year  Points
0   Riders     1  2014     876
1   Riders     2  2015     789
2   Devils     2  2014     863
3   Devils     3  2015     673
4    Kings     3  2014     741
5    kings     4  2015     812
6    Kings     1  2016     756
7    Kings     1  2017     788
8   Riders     2  2016     694
9   Royals     4  2014     701
10  Royals     1  2015     804
11  Riders     2  2017     690
```

```
>>> grouped = df.groupby('Year')
>>> grouped['Points'].agg(np.mean)
Year
2014    795.25
2015    769.50
2016    725.00
2017    739.00
Name: Points, dtype: float64
```

HCMUT EE Machine Learning & IoT Lab https://www.facebook.com/hcmut.ml.iot.lab

DataFrame – GroupBy

```
obj.groupby('key')
obj.groupby(['key1','key2'])
obj.groupby(key,axis=1)
```

Aggregations

```
>>> #import the pandas library
... import pandas as pd
>>>
>>> ipl_data = {'Team': ['Riders', 'Riders', 'Devils', 'Devils', 'Kings',
...     'kings', 'Kings', 'Kings', 'Riders', 'Royals', 'Royals', 'Riders'],
...     'Rank': [1, 2, 2, 3, 3,4 ,1 ,1,2 , 4,1,2],
...     'Year': [2014,2015,2014,2015,2014,2015,2016,2017,2016,2014,2015,2017],
...     'Points':[876,789,863,673,741,812,756,788,694,701,804,690]}
>>> df = pd.DataFrame(ipl_data)
>>> df
      Team  Rank  Year  Points
0   Riders     1  2014     876
1   Riders     2  2015     789
2   Devils     2  2014     863
3   Devils     3  2015     673
4    Kings     3  2014     741
5    kings     4  2015     812
6    Kings     1  2016     756
7    Kings     1  2017     788
8   Riders     2  2016     694
9   Royals     4  2014     701
10  Royals     1  2015     804
11  Riders     2  2017     690
```

```
>>> grouped = df.groupby('Team')
>>> grouped.agg(np.size)
        Rank  Year  Points
Team
Devils     2     2       2
Kings      3     3       3
Riders     4     4       4
Royals     2     2       2
kings      1     1       1
```

DataFrame – GroupBy

```
obj.groupby('key')
obj.groupby(['key1','key2'])
obj.groupby(key,axis=1)
```

Applying Multiple Aggregation Functions at Once

```
>>> #import the pandas library
... import pandas as pd
>>>
>>> ipl_data = {'Team': ['Riders', 'Riders', 'Devils', 'Devils', 'Kings',
...     'kings', 'Kings', 'Kings', 'Riders', 'Royals', 'Royals', 'Riders'],
...     'Rank': [1, 2, 2, 3, 3,4 ,1 ,1,2 , 4,1,2],
...     'Year': [2014,2015,2014,2015,2014,2015,2016,2017,2016,2014,2015,2017],
...     'Points':[876,789,863,673,741,812,756,788,694,701,804,690]}
>>> df = pd.DataFrame(ipl_data)
>>> df
      Team  Rank  Year  Points
0   Riders     1  2014     876
1   Riders     2  2015     789
2   Devils     2  2014     863
3   Devils     3  2015     673
4    Kings     3  2014     741
5    kings     4  2015     812
6    Kings     1  2016     756
7    Kings     1  2017     788
8   Riders     2  2016     694
9   Royals     4  2014     701
10  Royals     1  2015     804
11  Riders     2  2017     690
```

```
>>> grouped = df.groupby('Team')
>>> grouped['Points'].agg([np.sum, np.mean, np.std])
         sum         mean         std
Team
Devils  1536  768.000000  134.350288
Kings   2285  761.666667   24.006943
Riders  3049  762.250000   88.567771
Royals  1505  752.500000   72.831998
kings    812  812.000000         NaN
```

HCMUT EE Machine Learning & IoT Lab https://www.facebook.com/hcmut.ml.iot.lab

DataFrame – GroupBy

```
obj.groupby('key')
obj.groupby(['key1','key2'])
obj.groupby(key,axis=1)
```

Filtration

```
>>> #import the pandas library
... import pandas as pd
>>>
>>> ipl_data = {'Team': ['Riders', 'Riders', 'Devils', 'Devils', 'Kings',
...     'kings', 'Kings', 'Kings', 'Riders', 'Royals', 'Royals', 'Riders'],
...     'Rank': [1, 2, 2, 3, 3,4 ,1 ,1,2 , 4,1,2],
...     'Year': [2014,2015,2014,2015,2014,2015,2016,2017,2016,2014,2015,2017],
...     'Points':[876,789,863,673,741,812,756,788,694,701,804,690]}
>>> df = pd.DataFrame(ipl_data)
>>> df
      Team  Rank  Year  Points
0   Riders     1  2014     876
1   Riders     2  2015     789
2   Devils     2  2014     863
3   Devils     3  2015     673
4    Kings     3  2014     741
5    kings     4  2015     812
6    Kings     1  2016     756
7    Kings     1  2017     788
8   Riders     2  2016     694
9   Royals     4  2014     701
10  Royals     1  2015     804
11  Riders     2  2017     690
```

```
>>> df.groupby('Team').filter(lambda x: len(x) >= 3)
      Team  Rank  Year  Points
0   Riders     1  2014     876
1   Riders     2  2015     789
4    Kings     3  2014     741
6    Kings     1  2016     756
7    Kings     1  2017     788
8   Riders     2  2016     694
11  Riders     2  2017     690
```

HCMUT EE Machine Learning & IoT Lab https://www.facebook.com/hcmut.ml.iot.lab