# Objective of the class

## Introduction to Numpy

- What is numpy?
- numpy performance test
- Introduction to numpy arrays
- Introduction to numpy function
- Dealing with Flat files using numpy
- Mathematical functions
- Statisticals function
- Operations with arrays

# Other Array Methods

arange

```
arange(start,stop=None,step=1, dtype=None)

In [63]: array
Out[63]: array([0, 1, 2, 3, 4, 5, 6, 7, 8, 9])

In [64]: array = np.arange(10).reshape((5,2))

In [65]: array
Out[65]:
array([[0, 1],
       [2, 3],
       [4, 5],
       [6, 7],
       [8, 9]])
```

# Other Array Methods

ones and zeros

```
In [67]: array = np.ones((3,2))

In [68]: array
Out[68]:
array([[ 1.,  1.],
       [ 1.,  1.],
       [ 1.,  1.]])

In [69]: array = np.zeros((3,2))

In [70]: array
Out[70]:
array([[ 0.,  0.],
       [ 0.,  0.],
       [ 0.,  0.]])
```

# Other Array Methods

Identity and fill

```
In [77]: array = np.identity(5)

In [78]: array
Out[78]:
array([[ 1.,  0.,  0.,  0.,  0.],
       [ 0.,  1.,  0.,  0.,  0.],
       [ 0.,  0.,  1.,  0.,  0.],
       [ 0.,  0.,  0.,  1.,  0.],
       [ 0.,  0.,  0.,  0.,  1.]])

In [79]: array.fill(5.0)
array([[ 5.,  5.,  5.,  5.,  5.],
       [ 5.,  5.,  5.,  5.,  5.],
       [ 5.,  5.,  5.,  5.,  5.],
       [ 5.,  5.,  5.,  5.,  5.],
       [ 5.,  5.,  5.,  5.,  5.]])
```

# NumPy Functions

## Diagonal Function

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

In [7]: array.diagonal()
Out[7]: array([1, 5, 9])

In [8]: array.diagonal(offset=1)
Out[8]: array([2, 6])

In [9]: array.diagonal(offset=2)
Out[9]: array([3])

In [10]: array.diagonal(offset=-1)
Out[10]: array([4, 8])

In [11]: array.diagonal(offset=-2)
Out[11]: array([7])
```

# NumPy Functions

Sum Function

```
array([[1, 2, 3],
       [4, 5, 6],
       [7, 8, 9]])

In [13]: array.sum()
Out[13]: 45

In [14]: array.sum(axis=1)
Out[14]: array([ 6, 15, 24])

In [15]: array.sum(axis=0)
Out[15]: array([12, 15, 18])
```

# NumPy Functions

## Prod Function

```
In [12]: array
array([[1, 2, 3],
       [4, 5, 6],
       [7, 8, 9]])



In [112]: array.prod()
Out[112]: 362880

In [113]: array.prod(axis=1)
Out[113]: array([  6, 120, 504])

In [114]: array.prod(axis=0)
Out[114]: array([ 28,  80, 162])
```

# NumPy Functions

Min Function and max function

```
In [12]: array
array([[1, 2, 3],
       [4, 5, 6],
       [7, 8, 9]])



In [24]: array.min()
Out[24]: 1

In [25]: array.min(axis=1)
Out[25]: array([1, 4, 7])

In [26]: array.min(axis=0)
Out[26]: array([1, 2, 3])
```

# NumPy Functions

argmin Function

```
In [31]: array
Out[31]:
array([[10,  2,  1],
       [ 1,  0,  3]])

In [32]: array.argmin()
Out[32]: 4

In [33]: array.argmin(axis=1)
Out[33]: array([2, 1])

In [34]: array.argmin(axis=0)
Out[34]: array([1, 1, 0])
```

# Statistics Array Methods

Mean Function

```
array([[1, 2, 3],
       [4, 5, 6],
       [7, 8, 9]])

In [40]: array.mean()
Out[40]: 5.0

In [41]: array.mean(axis=1)
Out[41]: array([ 2.,  5.,  8.])

In [42]: array.mean(axis=0)
Out[42]: array([ 4.,  5.,  6.])
```

# Statistics Array Methods

## Standard Deviation and Variance

```
array([[1, 2, 3],
       [4, 5, 6],
       [7, 8, 9]])

In [44]: array.std()
Out[44]: 2.5819888974716112

In [45]: array.std(axis=1)
Out[45]: array([ 0.81649658,  0.81649658,  0.81649658])

In [46]: array.std(axis=0)
Out[46]: array([ 2.44948974,  2.44948974,  2.44948974])

In [47]:

In [47]: array.var()
Out[47]: 6.666666666666667
```

# Other Array Methods

Clip

```
array([[1, 2, 3],
       [4, 5, 6],
       [7, 8, 9]])

In [50]: array.clip(2, 6)
Out[50]:
array([[2, 2, 3],
       [4, 5, 6],
       [6, 6, 6]])
```

# Other Array Methods

PTP - Peak to Peak

```
array([[1, 2, 3],
      [4, 5, 6],
      [7, 8, 9]])

In [52]: array.ptp()
Out[52]: 8

In [53]: array.ptp(axis=1)
Out[53]: array([2, 2, 2])

In [54]: array.ptp(axis=0)
Out[54]: array([6, 6, 6])
```

# Other Array Methods

Round

```
In [56]: array = np.array([1.5, 6.7, 2.1])

In [57]: array.round()
Out[57]: array([ 2.,  7.,  2.])

In [59]: array.round(decimals=1)
Out[59]: array([ 1.5,  6.7,  2.1])
```

# Other Array Methods

Linspace (Equaly Spaces)

```
In [82]: array = np.linspace(0,5,5)

In [83]: array
Out[83]: array([ 0.  ,  1.25,  2.5 ,  3.75,  5.  ])

In [84]: array = np.linspace(0,5,10)

In [85]: array
Out[85]:
array([ 0.        ,  0.55555556,  1.11111111,  1.66666667,  2.22222222,
        2.77777778,  3.33333333,  3.88888889,  4.44444444,  5.        ])
```

# Other Array Methods

Linspace (Equaly Spaces)

```
In [82]: array = np.linspace(0,5,5)

In [83]: array
Out[83]: array([ 0.  ,  1.25,  2.5 ,  3.75,  5.  ])

In [84]: array = np.linspace(0,5,10)

In [85]: array
Out[85]:
array([ 0.        ,  0.55555556,  1.11111111,  1.66666667,  2.22222222,
        2.77777778,  3.33333333,  3.88888889,  4.44444444,  5.        ])
```

# Other Array Methods

Linspace (Equaly Spaces)

```
In [82]: array = np.linspace(0,5,5)

In [83]: array
Out[83]: array([ 0.  ,  1.25,  2.5 ,  3.75,  5.  ])

In [84]: array = np.linspace(0,5,10)

In [85]: array
Out[85]:
array([ 0.        ,  0.55555556,  1.11111111,  1.66666667,  2.22222222,
        2.77777778,  3.33333333,  3.88888889,  4.44444444,  5.        ])
```

# Other Array Methods

Random Generation

```
In [94]: array = np.random.rand(10)

array([ 0.66604191,  0.0626421 ,  0.23912655,  0.76221033,  0.27270546,
        0.71345773,  0.14292213,  0.94666562,  0.07804906,  0.64202557])

In [96]: array = np.random.randint(10)

In [97]: array
Out[97]: 4
```

# Other Array Methods

Shuffling

```
In [98]: array = np.arange(10)

In [99]: array
Out[99]: array([0, 1, 2, 3, 4, 5, 6, 7, 8, 9])

In [101]: np.random.shuffle(array)

In [102]: array
Out[102]: array([7, 1, 4, 2, 3, 9, 5, 0, 8, 6])
```

# Other Array Methods

Sorting

```
In [105]: array
Out[105]: array([7, 1, 4, 2, 3, 9, 5, 0, 8, 6])

In [106]: array.argsort()
Out[106]: array([7, 1, 3, 4, 2, 6, 9, 0, 8, 5])

In [107]: array.sort()

In [108]: array
Out[108]: array([0, 1, 2, 3, 4, 5, 6, 7, 8, 9])
```

# Other Array Methods

Tile Function

```
>>> a = np.array([0, 1, 2])
>>> np.tile(a, 2)
array([0, 1, 2, 0, 1, 2])

>>> np.tile(a, (2, 2))
array([[0, 1, 2, 0, 1, 2],
       [0, 1, 2, 0, 1, 2]])

>>> b = np.array([[1, 2], [3, 4]])
>>> np.tile(b, 2)

array([[1, 2, 1, 2],
       [3, 4, 3, 4]])

>>> np.tile(b, (2, 1))
array([[1, 2],
       [3, 4],
       [1, 2],
       [3, 4]])
```

# Other Array Methods

Mathmatical Function You Should Try

- exp(x)
- log(x)
- log10(x)
- sqrt(x)
- absolute(x)
- conjugate(x)
- negative(x)
- ceil(x)
- floor(x)
- fabs(x)
- hypot(x,y)
- fmod(x,y)
- maximum(x,y)
- minimum(x,y)

- sin(x)
- sinh(x)
- cos(x)
- cosh(x)
- arccos(x)
- arccosh(x)
- arctan(x)
- arctanh(x)
- arcsin(x)
- arcsinh(x)
- arctan2(x,y)