Numpy:

In [46]:

```
### 1-d deminsional array:

import numpy as np

a = np.array([1,2,3,4])
a
```

Out[46]:

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

In [87]:

```
a = np.array([1,2,3,4])
a
```

Out[87]:

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

In [49]:

```
#### 2-d dimensional array:

c = np.array([[1,2,3,4],[1,2,3,4]])
c
```

Out[49]:

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

In [50]:

```
#### 3-d dimensional array:

d = np.array([[[1,2,3,4],[1,2,3,4]],[[1,2,3,4],[1,2,3,4]]])
d
```

Out[50]:

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

       [[1, 2, 3, 4],
        [1, 2, 3, 4]]])
```

In [54]:

```
c.ndim #### ndim is used to find the dimensions of your given array.
```

Out[54]:

```
2
```

In [58]:

```
a = np.array([1,2,43,4],ndmin =10)
a
```

Out[58]:

```
array([[[[[[[[[[ 1,  2, 43,  4]]]]]]]]]])
```

In [63]:

```
### accessing of elements from an array:

a = np.array([1,2,3,4])
a
```

Out[63]:

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

In [62]:

```
a[0]
```

Out[62]:

```
1
```

In [64]:

```
a[-1]
```

Out[64]:

```
4
```

In [65]:

```
a[3]
```

Out[65]:

```
4
```

In [68]:

```
### accessing elements from a 2-d array:

c
```

Out[68]:

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

In [70]:

```
c[0:2,0:2]
```

Out[70]:

```
array([[1, 2],
       [1, 2]])
```

In [71]:

```
c[1:3,1:3]
```

Out[71]:

```
array([[2, 3]])
```

In [72]:

```
c[0:2, 0:3]
```

Out[72]:

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

In [77]:

```
c[1,1:3]
```

Out[77]:

```
array([2, 3])
```

In [78]:

```
c = np.array([[0,0,0,0,0],[0,1,1,1,0],[0,1,5,1,0],[0,1,1,1,0],[0,0,0,0,0]])
```

In [79]:

```
c
```

Out[79]:

```
array([[0, 0, 0, 0, 0],
       [0, 1, 1, 1, 0],
       [0, 1, 5, 1, 0],
       [0, 1, 1, 1, 0],
       [0, 0, 0, 0, 0]])
```

In [80]:

```
### accessing elemnts from 3-d array:

d
```

Out[80]:

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

       [[1, 2, 3, 4],
        [1, 2, 3, 4]]])
```

In [82]:

```
d[0,0,2:] ### array_name[2-d array index,row_index,slicing/particular element index]
```

Out[82]:

```
array([3, 4])
```

In [83]:

```
d[1,1,2:]
```

Out[83]:

```
array([3, 4])
```

In [84]:

```
d[:,:,2:]
```

Out[84]:

```
array([[[3, 4],
        [3, 4]],

       [[3, 4],
        [3, 4]]])
```

In [91]:

```
### shape of an array:

a = np.array([[1,2,4,5],[1,2,3,4]])
```

In [94]:

```
a
```

Out[94]:

```
array([[1, 2, 4, 5],
       [1, 2, 3, 4]])
```

In [95]:

```
a.shape
```

Out[95]:

```
(2, 4)
```

In [98]:

```
a = np.array([1,2,3,4])
a
```

Out[98]:

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

In [97]:

```
a.shape
```

Out[97]:

```
(4,)
```

In [99]:

```
d.shape
```

Out[99]:

```
(2, 2, 4)
```

In [100]:

```
d
```

Out[100]:

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

       [[1, 2, 3, 4],
        [1, 2, 3, 4]]])
```

In [109]:

```
### Reshape of an array:

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

Out[109]:

```
array([ 1,  2,  3,  4,  5,  6,  7,  8,  9, 10, 11, 12])
```

In [103]:

```
a.reshape(2,5) ### array_name.reshape(rows,columns)
```

Out[103]:

```
array([[ 1,  2,  3,  4,  5],
       [ 6,  7,  8,  9, 10]])
```

In [104]:

```
a.reshape(5,2)
```

Out[104]:

```
array([[ 1,  2],
       [ 3,  4],
       [ 5,  6],
       [ 7,  8],
       [ 9, 10]])
```

In [108]:

```
a.reshape(1,10)
```

Out[108]:

```
array([[ 1,  2,  3,  4,  5,  6,  7,  8,  9, 10]])
```

In [110]:

```
### converting 1-d to 3-d :

a.reshape(2,3,2)
```

Out[110]:

```
array([[[ 1,  2],
        [ 3,  4],
        [ 5,  6]],

       [[ 7,  8],
        [ 9, 10],
        [11, 12]]])
```

In [111]:

```
a = np.array([1,2,3,4,5,6,7,8,9,10])
```

In [112]:

```
a.reshape(10,1)
```

Out[112]:

```
array([[ 1],
       [ 2],
       [ 3],
       [ 4],
       [ 5],
       [ 6],
       [ 7],
       [ 8],
       [ 9],
       [10]])
```

In [117]:

```
### if the unknown dimension:

a = np.array([1,2,3,4,5,6,7,8])
a.reshape(2,2,-1)
```

Out[117]:

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

       [[5, 6],
        [7, 8]]])
```

In [119]:

```
a.reshape(2,-1)
```

Out[119]:

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

In [124]:

```
a = np.array([1,2,3,4,5,6,7])
```

In [127]:

```
#### iterating arrays:
### 1-d array:

a = np.array([1,2,3])

for x in a:
    print(x)
```

```
1
2
3
```

In [134]:

```
### 2-d array:

b = np.arange(1,11)
b = b.reshape(2,5)

for x in b:
    print(x)
```

```
[1 2 3 4 5]
[ 6  7  8  9 10]
```

In [135]:

```
for x in b:
    for y in x:
        print(y)
```

```
1
2
3
4
5
6
7
8
9
10
```

In [136]:

```
d
```

Out[136]:

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

       [[1, 2, 3, 4],
        [1, 2, 3, 4]]])
```

In [137]:

```
for x in d:
    print(x)
```

```
[[1 2 3 4]
 [1 2 3 4]]
[[1 2 3 4]
 [1 2 3 4]]
```

In [140]:

```
for x in d:
    for y in x:
        for z in y:
            print(z,end = " ")
```

```
1 2 3 4 1 2 3 4 1 2 3 4 1 2 3 4
```

In [144]:

```
#### nditer()

for x in np.nditer(d):
    print(x,end = ",")
```

```
1,2,3,4,1,2,3,4,1,2,3,4,1,2,3,4,
```

In [145]:

```
np.arange(10,20,2)
```

Out[145]:

```
array([10, 12, 14, 16, 18])
```

In [148]:

```
### Joining of arrays:

## 1-d array:

a = np.arange(1,6)
b = np.arange(6,11)
c = np.concatenate((a,b))
c
```

Out[148]:

```
array([ 1,  2,  3,  4,  5,  6,  7,  8,  9, 10])
```

In [157]:

```
np.ones((2,3),dtype = int)
```

Out[157]:

```
array([[1, 1, 1],
       [1, 1, 1]])
```

In [164]:

```
### 2-d array:

a = np.array([[1,2],[3,4]])
b = np.array([[5,6],[7,8]])
c = np.concatenate((a,b))
c
```

Out[164]:

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

In [166]:

```
a = np.arange(1,11).reshape(2,5)
a
```

Out[166]:

```
array([[ 1,  2,  3,  4,  5],
       [ 6,  7,  8,  9, 10]])
```

In [168]:

```
### stastical concepts in numpy:

a = np.array([[3,7,5],[8,4,3],[2,4,9]])

a
```

Out[168]:

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

In [170]:

```
np.amin(a)
```

Out[170]:

```
2
```

In [171]:

```
np.amin(a,1)
```

Out[171]:

```
array([3, 3, 2])
```

In [172]:

```
np.min(a)
```

Out[172]:

```
2
```

In [173]:

```
np.max(a)
```

Out[173]:

```
9
```

In [174]:

```
np.amax(a)
```

Out[174]:

```
9
```

In [175]:

```
a.mean() ### mean
```

Out[175]:

```
5.0
```

In [176]:

```
a.var() ### variance
```

Out[176]:

```
5.333333333333333
```

In [178]:

```
a.std() ### standard deviation
```

Out[178]:

```
2.309401076758503
```

In [181]:

```
np.median(a) ## median
```

Out[181]:

```
4.0
```

In [182]:

```
np.average(a)
```

Out[182]:

```
5.0
```

In [188]:

```
np.average(a,1)   ### calculating average from each row
np.average(a,0)  ### calculating average from each column
```

Out[188]:

```
array([4.33333333, 5.        , 5.66666667])
```

In [189]:

```
### math operations:

a
```

Out[189]:

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

In [195]:

```
a.sum() ### sum of all elements
```

Out[195]:

```
45
```

In [194]:

```
np.sqrt(a) ### square root
```

Out[194]:

```
array([[1.73205081, 2.64575131, 2.23606798],
       [2.82842712, 2.        , 1.73205081],
       [1.41421356, 2.        , 3.        ]])
```

In [197]:

```
np.ceil(a)
```

Out[197]:

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

In [198]:

```
np.floor(a)
```

Out[198]:

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

In [199]:

```
np.exp(a)
```

Out[199]:

```
array([[2.00855369e+01, 1.09663316e+03, 1.48413159e+02],
       [2.98095799e+03, 5.45981500e+01, 2.00855369e+01],
       [7.38905610e+00, 5.45981500e+01, 8.10308393e+03]])
```

In [200]:

```
np.log(a)
```

Out[200]:

```
array([[1.09861229, 1.94591015, 1.60943791],
       [2.07944154, 1.38629436, 1.09861229],
       [0.69314718, 1.38629436, 2.19722458]])
```

In [201]:

```
np.sin(a)
```

Out[201]:

```
array([[ 0.14112001,  0.6569866 , -0.95892427],
       [ 0.98935825, -0.7568025 ,  0.14112001],
       [ 0.90929743, -0.7568025 ,  0.41211849]])
```

In [202]:

```
np.cos(a)
```

Out[202]:

```
array([[-0.9899925 ,  0.75390225,  0.28366219],
       [-0.14550003, -0.65364362, -0.9899925 ],
       [-0.41614684, -0.65364362, -0.91113026]])
```

In [203]:

```
np.tan(a)
```

Out[203]:

```
array([[-0.14254654,  0.87144798, -3.38051501],
       [-6.79971146,  1.15782128, -0.14254654],
       [-2.18503986,  1.15782128, -0.45231566]])
```

In [204]:

```
import math
```

In [205]:

```python
dir(math)
```

Out[205]:

```
['__doc__',
 '__loader__',
 '__name__',
 '__package__',
 '__spec__',
 'acos',
 'acosh',
 'asin',
 'asinh',
 'atan',
 'atan2',
 'atanh',
 'ceil',
 'copysign',
 'cos',
 'cosh',
 'degrees',
 'e',
 'erf',
 'erfc',
 'exp',
 'expm1',
 'fabs',
 'factorial',
 'floor',
 'fmod',
 'frexp',
 'fsum',
 'gamma',
 'gcd',
 'hypot',
 'inf',
 'isclose',
 'isfinite',
 'isinf',
 'isnan',
 'ldexp',
 'lgamma',
 'log',
 'log10',
 'log1p',
 'log2',
 'modf',
 'nan',
 'pi',
 'pow',
 'radians',
 'remainder',
 'sin',
 'sinh',
 'sqrt',
 'tan',
 'tanh',
 'tau',
 'trunc']
```

In [208]:

```
np.remainder(a,1)
```

Out[208]:

```
array([[0, 0, 0],
       [0, 0, 0],
       [0, 0, 0]], dtype=int32)
```

In [209]:

```
np.log2(a)
```

Out[209]:

```
array([[1.5849625 , 2.80735492, 2.32192809],
       [3.        , 2.        , 1.5849625 ],
       [1.        , 2.        , 3.169925  ]])
```

In [210]:

```
np.log10(a)
```

Out[210]:

```
array([[0.47712125, 0.84509804, 0.69897   ],
       [0.90308999, 0.60205999, 0.47712125],
       [0.30103   , 0.60205999, 0.95424251]])
```

In [211]:

```
np.degrees(a)
```

Out[211]:

```
array([[171.88733854, 401.07045659, 286.47889757],
       [458.3662361 , 229.18311805, 171.88733854],
       [114.59155903, 229.18311805, 515.66201562]])
```

In [214]:

```
dir(math)
```

Out[214]:

```
['__doc__',
 '__loader__',
 '__name__',
 '__package__',
 '__spec__',
 'acos',
 'acosh',
 'asin',
 'asinh',
 'atan',
 'atan2',
 'atanh',
 'ceil',
 'copysign',
 'cos',
 'cosh',
 'degrees',
 'e',
 'erf',
 'erfc',
 'exp',
 'expm1',
 'fabs',
 'factorial',
 'floor',
 'fmod',
 'frexp',
 'fsum',
 'gamma',
 'gcd',
 'hypot',
 'inf',
 'isclose',
 'isfinite',
 'isinf',
 'isnan',
 'ldexp',
 'lgamma',
 'log',
 'log10',
 'log1p',
 'log2',
 'modf',
 'nan',
 'pi',
 'pow',
 'radians',
 'remainder',
 'sin',
 'sinh',
 'sqrt',
 'tan',
 'tanh',
 'tau',
 'trunc']
```

In [215]:

```python
help(math.acos)
```

```
Help on built-in function acos in module math:

acos(x, /)
    Return the arc cosine (measured in radians) of x.
```

In [ ]:

```python

```