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

In [2]:
```
#Create an array.
a = np.array([ [0,1,2,3], [4,5,6,7], [2,1,6,4] ])
a
```

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

In [3]:
```
print(a.ndim)

print(a.shape)

print(a.size)

print(a.dtype.name)
```

```
2
(3, 4)
12
int32
```

In [4]:
```
a = np.array([ [0,1,2,3], [4,5,6,7], [2,1,6,4] ])
b = np.array([ [0,1,2,3], [4,5,6,7], [2,1,6,4] ])
```

In [5]:
```
#Subtraction
a-b
```

Out[5]:
```
array([[0, 0, 0, 0],
       [0, 0, 0, 0],
       [0, 0, 0, 0]])
```

In [6]:
```
#Addition
a+b
```

Out[6]:
```
array([[ 0,  2,  4,  6],
       [ 8, 10, 12, 14],
       [ 4,  2, 12,  8]])
```

In [17]:
```
#Dot Product
c = np.arange(10,22).reshape(4,3)
print(c)

np.dot(a,c)
```

```
[[10 11 12]
 [13 14 15]
 [16 17 18]
 [19 20 21]]
```

Out[17]:
```
array([[102, 108, 114],
       [334, 356, 378],
       [205, 218, 231]])
```

In [7]:
```
#Multiplication
a*b
```

Out[7]:
```
array([[ 0,  1,  4,  9],
       [16, 25, 36, 49],
       [ 4,  1, 36, 16]])
```

In [8]:
```
#Division
a/b
```

```
C:\Users\acer\PycharmProjects\untitled\venv\lib\site-packages\ipykernel_launche
r.py:2: RuntimeWarning: invalid value encountered in true_divide
```

Out[8]:
```
array([[nan,  1.,  1.,  1.],
       [ 1.,  1.,  1.,  1.],
       [ 1.,  1.,  1.,  1.]])
```

In [9]:
```
#Squaring
a**2
```

Out[9]:
```
array([[ 0,  1,  4,  9],
       [16, 25, 36, 49],
       [ 4,  1, 36, 16]], dtype=int32)
```

In [10]:
```
#Trigonometric function

np.cos(a)
```

Out[10]:
```
array([[ 1.        ,  0.54030231, -0.41614684, -0.9899925 ],
       [-0.65364362,  0.28366219,  0.96017029,  0.75390225],
       [-0.41614684,  0.54030231,  0.96017029, -0.65364362]])
```

In [13]:
```
np.cos(90)       #Takes in input angle in terms of radians.
```

Out[13]:
```
-0.4480736161291701
```

In [14]:
```
#Conditional Operations.

a<3
```

Out[14]:
```
array([[ True,  True,  True, False],
       [False, False, False, False],
       [ True,  True, False, False]])
```

Indexing and Slicing

In [18]:
```
aa = np.arange(10,30)
ab = np.arange(10,30).reshape(5,4)

print(aa)
print(ab)
```

```
[10 11 12 13 14 15 16 17 18 19 20 21 22 23 24 25 26 27 28 29]
[[10 11 12 13]
 [14 15 16 17]
 [18 19 20 21]
 [22 23 24 25]
 [26 27 28 29]]
```

In [19]:
```
aa[4: 10: 1]
```

Out[19]:
```
array([14, 15, 16, 17, 18, 19])
```

In [20]:
```
aa[10:4:-1]
```

Out[20]:
```
array([20, 19, 18, 17, 16, 15])
```

In [21]:
```
aa[10]
```

Out[21]:
```
20
```

In [22]:
```
aa[4]
```

Out[22]:
```
14
```

In [25]:
```
#Selecting the first column of ab.

ab[:, 0]
```

Out[25]:
```
array([10, 14, 18, 22, 26])
```

In [27]:
```
#Selecting the first and last column of ab.

ab[:, 0:4:3]
```

Out[27]:
```
array([[10, 13],
       [14, 17],
       [18, 21],
       [22, 25],
       [26, 29]])
```

In [36]:
```
#Slicing the alst row and the second row.

ab[-1:-5:-3, :]    #Negative indexing.
```

Out[36]:
```
array([[26, 27, 28, 29],
       [14, 15, 16, 17]])
```

In [38]:
```
#Slicing all the rows after 2nd row.
ab[2:]
```

Out[38]:
```
array([[18, 19, 20, 21],
       [22, 23, 24, 25],
       [26, 27, 28, 29]])
```

In [42]:
```
#Slicing the first 3 rows.
print(ab[:,:])

ab[:3]
```

```
[[10 11 12 13]
 [14 15 16 17]
 [18 19 20 21]
 [22 23 24 25]
 [26 27 28 29]]
```

Out[42]:
```
array([[10, 11, 12, 13],
       [14, 15, 16, 17],
       [18, 19, 20, 21]])
```

In [69]:
```
#Slice items that are greater than 15 and less than 20.
ab[(ab>15) & (ab<20) ]
```

Out[69]:
```
array([16, 17, 18, 19])
```

Shape manipulation

In [43]:
```
#Flatten the array.
print(ab)

ab.ravel()
```

```
[[10 11 12 13]
 [14 15 16 17]
 [18 19 20 21]
 [22 23 24 25]
 [26 27 28 29]]
```

Out[43]:
```
array([10, 11, 12, 13, 14, 15, 16, 17, 18, 19, 20, 21, 22, 23, 24, 25, 26,
       27, 28, 29])
```

In [47]:
```
print(np.shape(ab))
```

```
(5, 4)
```

In [53]:
```
#Reshaping the array into 5X2.

ab1 = ab.reshape(10,2)
```

In [54]:
```
ab1
```

Out[54]:
```
array([[10, 11],
       [12, 13],
       [14, 15],
       [16, 17],
       [18, 19],
       [20, 21],
       [22, 23],
       [24, 25],
       [26, 27],
       [28, 29]])
```

In [55]:
```
ab
```

Out[55]:
```
array([[10, 11, 12, 13],
       [14, 15, 16, 17],
       [18, 19, 20, 21],
       [22, 23, 24, 25],
       [26, 27, 28, 29]])
```

In [56]:
```
#Transpose ab1.

ab1 = ab1.transpose()
ab1
```

Out[56]:
```
array([[10, 12, 14, 16, 18, 20, 22, 24, 26, 28],
       [11, 13, 15, 17, 19, 21, 23, 25, 27, 29]])
```

numpy random numbers.

In [57]:
```
#Generate a sequesnce of random numbers.

np.random.random(5)
```

Out[57]:
```
array([0.74212233, 0.17272467, 0.19815145, 0.39641591, 0.69388813])
```

In [64]:
```
#Generate a series of random numbers of particular order array.

np.random.random(6).reshape(3,2)
```

Out[64]:
```
array([[0.62299691, 0.35229494],
       [0.73987795, 0.30585124],
       [0.52969894, 0.69227502]])
```

In [59]:
```
# Generate a random integer between 10 and 50.

np.random.randint(10,50)
```

Out[59]:
```
34
```

In [61]:
```
#Generate random integers of particular array order.

np.random.randint(20,30, size = (5,2))
```

Out[61]:
```
array([[27, 23],
       [28, 22],
       [23, 21],
       [27, 29],
       [26, 28]])
```

Rounding off.

In [71]:
```
g = np.arange(0,180,30)
g
```

Out[71]:
```
array([  0,  30,  60,  90, 120, 150])
```

In [72]:
```
np.sin(g)
```

Out[72]:
```
array([ 0.        , -0.98803162, -0.30481062,  0.89399666,  0.58061118,
       -0.71487643])
```

In [74]:
```
g_inDegrees = ((np.pi) / 180) * g
g_inDegrees
```

Out[74]:
```
array([0.        , 0.52359878, 1.04719755, 1.57079633, 2.0943951 ,
       2.61799388])
```

In [76]:
```
result = np.sin(g_inDegrees)
result
```

Out[76]:
```
array([0.       , 0.5      , 0.8660254, 1.       , 0.8660254, 0.5      ])
```

In [77]:
```
#Rounding off to two decimal digits.

result = np.around(result, decimals = 2)
result
```

Out[77]:
```
array([0.  , 0.5 , 0.87, 1.  , 0.87, 0.5 ])
```

In [78]:
```
np.around(result, decimals = -1)
```

Out[78]:
```
array([0., 0., 0., 0., 0., 0.])
```

In [80]:
```
#Direct method.
np.around( np.log(g [1:] ), 2)     #Excluding log(0) as it gives infinity.
```

Out[80]:
```
array([3.4 , 4.09, 4.5 , 4.79, 5.01])
```

In [82]:
```
np.around(np.pi, 4)
```

Out[82]:
```
3.1416
```