In [1]:

```
import numpy as np
```

In [4]:

```
a = np.array([1,2,3], dtype='int32')
print(a)
```

```
[1 2 3]
```

In [5]:

```
print("1 :",a.ndim)
print("2 :",a.shape)
print("3 :",a.dtype)
print("4 :",a.itemsize)
print("5 :",a.nbytes )
print("6 :",a.size)
print("7 :", type(a))
```

```
1 : 1
2 : (3,)
3 : int32
4 : 4
5 : 12
6 : 3
7 : <class 'numpy.ndarray'>
```

In [9]:

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

```
[[1 2 3]
 [4 5 6]]
```

In [10]:

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

Out[10]:

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

In [12]:

```
v1=np.array([[1,2,3],[4,5,6]])
v2=np.array([[7,8,9],[10,11,12]])
v3=np.vstack((v1,v2))
print(v3)
```

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

In [13]:

```
v1=np.array([[1,2,3],[4,5,6]])
v2=np.array([[7,8,9],[10,11,12]])
v3=np.hstack((v1,v2))
print(v3)
```

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

In [14]:

```
a = np.array([1,2,3])  #Tuple
print(a.shape)
print(a.ndim)
print(a)
```

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

In [15]:

```
a = np.array([1,2,3]).reshape(1,3) #Row Vector
print(a.shape)
print(a.ndim)
print(a)
```

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

In [16]:

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

```
[[[ 1  2]
  [ 3  4]
  [ 5  6]]

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

In [19]:

```
a = np.array([[1,2,3],[4,5,6]]).reshape(-1,3) #Row Vector, -1 automatically finds another d
print(a.shape)
print(a.ndim)
print(a)
```

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

In [20]:

```
a=np.array([[1, 2, 3], [4, 5, 6], [7, 8, 9]])
print(a,"\n")
print("After Transpose")
print(a.transpose())
```

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

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

In [21]:

```
a = np.array([1, 2, 3])
print(a.shape)
print(a.ndim)
print(a)
print("After Transpose")
b=a.transpose()
print(b.shape)
print(b.ndim)
print(b)
```

```
(3,)
1
[1 2 3]
After Transpose
(3,)
1
[1 2 3]
```

In [24]:

```
v1=[1,2,3]
v2=[4,5,6]
v3=np.add(v1,v2)
print(v3)
```

```
[5 7 9]
```

In [25]:

```
x=np.array([1, 2, 3, 4])
y=np.array([1, 2, 3, 4])
z=x+y
print(z)
```

```
[2 4 6 8]
```

In [26]:

```
x = [1, 2, 3, 4]
y = [1, 2, 3, 4]
 #Vectorization
print("1 :",np.add(x, y))
print("2 :",np.subtract(x,y))
print("3 :",np.multiply(x,y))
print("4 :", np.divide(x,y))
print("5 :",np.power(x,y))
print("6 :",np.mod(x,y))
print("7 :",np.remainder(x,y))
print("8 :",np.divmod(x,y))
```

```
1 : [2 4 6 8]
2 : [0 0 0 0]
3 : [ 1  4  9 16]
4 : [1. 1. 1. 1.]
5 : [  1   4  27 256]
6 : [0 0 0 0]
7 : [0 0 0 0]
8 : (array([1, 1, 1, 1], dtype=int32), array([0, 0, 0, 0], dtype=int32))
```

In [27]:

```
x= np.arange(15).reshape(3, 5)
print(x,"\n")
print(x+2,"\n")
print(np.sin(x))
```

```
[[ 0  1  2  3  4]
 [ 5  6  7  8  9]
 [10 11 12 13 14]]

[[ 2  3  4  5  6]
 [ 7  8  9 10 11]
 [12 13 14 15 16]]

[[ 0.          0.84147098  0.90929743  0.14112001 -0.7568025 ]
 [-0.95892427 -0.2794155   0.6569866   0.98935825  0.41211849]
 [-0.54402111 -0.99999021 -0.53657292  0.42016704  0.99060736]]
```

In [28]:

```
arr = np.ceil([-3.1666, 3.6667])
print(arr)
```

```
[-3.  4.]
```

In [31]:

```
def myadd(x, y):
  return x+y

myadd = np.frompyfunc(myadd, 2, 1)          #add function to your NumPy ufunc library with t
print(myadd([1, 2, 3, 4], [5, 6, 7, 8]))
```

```
[6 8 10 12]
```

In [ ]:

```

```