Experiment - 4 | NumPy

```
In [2]: import numpy as np
```

```
In [3]: a = [1,2,3,4,5,6]
        print(a)
        print(type(a))

[1, 2, 3, 4, 5, 6]
<class 'list'>
```

```
In [4]: a = np.array([1,2,3,4,5,6])
        print(a)
        print(type(a))

[1 2 3 4 5 6]
<class 'numpy.ndarray'>
```

ndim

```
In [6]: a.ndim

Out[6]: 1
```

```
In [7]: a = np.array([[1,2,3],[4,5,6],[7,8,9]])
        print(a)
        print(type(a))
        a.ndim

[[1 2 3]
 [4 5 6]
 [7 8 9]]
<class 'numpy.ndarray'>

Out[7]: 2
```

```
In [8]: a = np.array([[[1,2,3],[4,5,6],[7,8,9]],[[1,1,1],[2,2,2],[3,3,3]]])
        print(a)
        print(type(a))
        a.ndim

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

 [[1 1 1]
  [2 2 2]
  [3 3 3]]]
<class 'numpy.ndarray'>

Out[8]: 3
```

```
In [9]: a = np.array([[1,2,3,2,3],[4,5,6,5,6],[7,8,9,8,9],[1,1,1,1,1],[2,2,2,3,3]])
        print(a)
        print(type(a))
        a.ndim

[[1 2 3 2 3]
 [4 5 6 5 6]
 [7 8 9 8 9]
 [1 1 1 1 1]
 [2 2 2 3 3]]
<class 'numpy.ndarray'>

Out[9]: 2
```

```
In [10]: a = np.array([[[1,3],[4,5]],[[1,3],[4,5]],[[1,3],[4,5]]])
         print(a)
         print(type(a))
         a.ndim

[[[1 3]
  [4 5]]

 [[1 3]
  [4 5]]

 [[1 3]
  [4 5]]]
<class 'numpy.ndarray'>

Out[10]: 3
```

size - show no. of elements

```
In [12]: a.size

Out[12]: 12
```

shape - show no. of rows & column

```
In [14]: a.shape

Out[14]: (3, 2, 2)
```

dtype - show datatype of array

```
In [16]: a.dtype

Out[16]: dtype('int64')
```

```
In [17]: a = np.array([1.5,2.3,3,4.5,5.6,6.4])
         print(a)
print(type(a))
```

```
[1.5 2.3 3.  4.5 5.6 6.4]
<class 'numpy.ndarray'>
```

In [18]:
```
a.dtype
```

Out[18]:
```
dtype('float64')
```

ones() and zeros()

In [20]:
```
z = np.zeros((4,4), dtype='int32')
z
```

Out[20]:
```
array([[0, 0, 0, 0],
       [0, 0, 0, 0],
       [0, 0, 0, 0],
       [0, 0, 0, 0]], dtype=int32)
```

In [21]:
```
o = np.ones((4,4), dtype=int)
o
```

Out[21]:
```
array([[1, 1, 1, 1],
       [1, 1, 1, 1],
       [1, 1, 1, 1],
       [1, 1, 1, 1]])
```

In [22]:
```
o = np.ones((4,4), dtype=str)
o
```

Out[22]:
```
array([['1', '1', '1', '1'],
       ['1', '1', '1', '1'],
       ['1', '1', '1', '1'],
       ['1', '1', '1', '1']], dtype='<U1')
```

In [23]:
```
o = np.zeros((4,4), dtype=str)
o
```

Out[23]:
```
array([['', '', '', ''],
       ['', '', '', ''],
       ['', '', '', ''],
       ['', '', '', '']], dtype='<U1')
```

In [24]:
```
o = np.ones((4,4), dtype=bool)
o
```

Out[24]:
```
array([[ True,  True,  True,  True],
       [ True,  True,  True,  True],
       [ True,  True,  True,  True],
       [ True,  True,  True,  True]])
```

In [25]:
```
o = np.zeros((4,4), dtype=bool)
o
```

Out[25]:
```
array([[False, False, False, False],
       [False, False, False, False],
       [False, False, False, False],
       [False, False, False, False]])
```

empty()

In [27]:
```
e = np.empty((4,4))
e
```

Out[27]:
```
array([[2.93873566e-316, 0.00000000e+000, 2.41907520e-312,
        2.14321575e-312],
       [2.46151512e-312, 2.31297541e-312, 2.35541533e-312,
        2.05833592e-312],
       [2.22809558e-312, 2.56761491e-312, 2.48273508e-312,
        2.05833592e-312],
       [2.05833592e-312, 2.29175545e-312, 2.07955588e-312,
        2.14321575e-312]])
```

arange()

In [29]:
```
arr = np.arange(50)
print(arr)
```

```
[ 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 36 37 38 39 40 41 42 43 44 45 46 47
 48 49]
```

In [30]:
```
arr = np.arange(2,13)
print(arr)
```

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

In [31]:
```
arr = np.arange(2,13,2)
print(arr)
```

```
[ 2  4  6  8 10 12]
```

In [32]:
```
arr = np.arange(2,13,1.2)
print(arr)
```

```
[ 2.   3.2  4.4  5.6  6.8  8.   9.2 10.4 11.6 12.8]
```

linspace()

In [34]:
```
l = np.linspace(1,100,5)
print(l)
```

```
[  1.    25.75  50.5   75.25 100.  ]
```

In [35]:
```
l = np.linspace(1,2,50)
print(l)
```

```
[1.         1.02040816 1.04081633 1.06122449 1.08163265 1.10204082
 1.12244898 1.14285714 1.16326531 1.18367347 1.20408163 1.2244898
 1.24489796 1.26530612 1.28571429 1.30612245 1.32653061 1.34693878
 1.36734694 1.3877551  1.40816327 1.42857143 1.44897959 1.46938776
 1.48979592 1.51020408 1.53061224 1.55102041 1.57142857 1.59183673
 1.6122449  1.63265306 1.65306122 1.67346939 1.69387755 1.71428571
 1.73469388 1.75510204 1.7755102  1.79591837 1.81632653 1.83673469
 1.85714286 1.87755102 1.89795918 1.91836735 1.93877551 1.95918367
 1.97959184 2.        ]
```

```
In [36]: l = np.linspace(100,200,3)
         print(l)
```

```
[100. 150. 200.]
```

reshape()

```
In [38]: demo = np.linspace(1,10,25)
         demo
```

```
Out[38]: array([ 1.   ,  1.375,  1.75 ,  2.125,  2.5  ,  2.875,  3.25 ,  3.625,
                 4.   ,  4.375,  4.75 ,  5.125,  5.5  ,  5.875,  6.25 ,  6.625,
                 7.   ,  7.375,  7.75 ,  8.125,  8.5  ,  8.875,  9.25 ,  9.625,
                10.   ])
```

```
In [39]: demo.reshape((5,5))
```

```
Out[39]: array([[ 1.   ,  1.375,  1.75 ,  2.125,  2.5  ],
                [ 2.875,  3.25 ,  3.625,  4.   ,  4.375],
                [ 4.75 ,  5.125,  5.5  ,  5.875,  6.25 ],
                [ 6.625,  7.   ,  7.375,  7.75 ,  8.125],
                [ 8.5  ,  8.875,  9.25 ,  9.625, 10.   ]])
```

```
In [40]: # q1

         arr_3d = np.arange(1,13).reshape((3,2,2))
         print(arr_3d)
```

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

 [[ 5  6]
  [ 7  8]]

 [[ 9 10]
  [11 12]]]
```

ravel()

```
In [42]: np.ravel(arr_3d)
```

```
Out[42]: array([ 1,  2,  3,  4,  5,  6,  7,  8,  9, 10, 11, 12])
```

```
In [43]: arr_3d.ravel()
```

```
Out[43]: array([ 1,  2,  3,  4,  5,  6,  7,  8,  9, 10, 11, 12])
```

transpose()

```
In [45]: arr = np.arange(1,13).reshape((4,3))
         print(arr)
```

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

```
In [46]: arr.transpose()
```

```
Out[46]: array([[ 1,  4,  7, 10],
                [ 2,  5,  8, 11],
                [ 3,  6,  9, 12]])
```

```
In [47]: arr.T
```

```
Out[47]: array([[ 1,  4,  7, 10],
                [ 2,  5,  8, 11],
                [ 3,  6,  9, 12]])
```

```
In [48]: arr1 = np.arange(1,17).reshape((4,4))
         arr2 = np.arange(1,17).reshape((4,4))
         print(arr1, arr2, sep='\n\n')
```

```
[[ 1  2  3  4]
 [ 5  6  7  8]
 [ 9 10 11 12]
 [13 14 15 16]]

[[ 1  2  3  4]
 [ 5  6  7  8]
 [ 9 10 11 12]
 [13 14 15 16]]
```

mathemetical operation using numpy

```
In [50]: print(arr1 + arr2)
```

```
[[ 2  4  6  8]
 [10 12 14 16]
 [18 20 22 24]
 [26 28 30 32]]
```

```
In [51]: print(arr1 - arr2)
```

```
[[0 0 0 0]
 [0 0 0 0]
 [0 0 0 0]
 [0 0 0 0]]
```

```
In [52]: print(arr1 * arr2)
```

```
[[  1   4   9  16]
 [ 25  36  49  64]
 [ 81 100 121 144]
 [169 196 225 256]]
```

```
In [53]: print(arr1 @ arr2)  # matrix multiplication
```

```
[[ 90 100 110 120]
 [202 228 254 280]
 [314 356 398 440]
 [426 484 542 600]]
```

```
In [54]: np.dot(arr1, arr2)  # matrix multiplication
```

```
Out[54]: array([[ 90, 100, 110, 120],
                [202, 228, 254, 280],
                [314, 356, 398, 440],
                [426, 484, 542, 600]])
```

```
In [55]: np.subtract(arr1, arr2)
```

```
Out[55]: array([[0, 0, 0, 0],
                [0, 0, 0, 0],
                [0, 0, 0, 0],
                [0, 0, 0, 0]])
```

```
In [56]: np.multiply(arr1, arr2)  # element multiplication
```

```
Out[56]: array([[  1,   4,   9,  16],
                [ 25,  36,  49,  64],
                [ 81, 100, 121, 144],
                [169, 196, 225, 256]])
```

```
In [57]: np.divide(arr1, arr2)
```

```
Out[57]: array([[1., 1., 1., 1.],
                [1., 1., 1., 1.],
                [1., 1., 1., 1.],
                [1., 1., 1., 1.]])
```

```
In [ ]:
```