Assignment-1

Now that we've learned about NumPy let's test your knowledge. We'll start off with a few simple tasks, and then you'll be asked some more complicated questions.

Import NumPy as np

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

Create an array of 10 zeros

In [2]:
```
a=np.zeros(10)
a
```

Out[2]:
```
array([0., 0., 0., 0., 0., 0., 0., 0., 0., 0.])
```

Create an array of 10 ones

In [3]:
```
b=np.ones(10)
b
```

Out[3]:
```
array([1., 1., 1., 1., 1., 1., 1., 1., 1., 1.])
```

Create an array of 10 fives

In [4]:
```
d=[5,5,5,5,5,5,5,5,5,5]
e=np.array(d)
e
```

Out[4]:
```
array([5, 5, 5, 5, 5, 5, 5, 5, 5, 5])
```

Create an array of the integers from 10 to 50

In [5]:
```
arr=np.arange(10,51)
arr
```

Out[5]:
```
array([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, 50])
```

Create an array of all the even integers from 10 to 50

In [6]:
```
even = np.arange(10,51,2)
even
```

Out[6]:
```
array([10, 12, 14, 16, 18, 20, 22, 24, 26, 28, 30, 32, 34, 36, 38, 40, 42,
       44, 46, 48, 50])
```

Create a 3x3 matrix with values ranging from 0 to 8

In [7]:
```
matrix=np.arange(9).reshape(3,3)
matrix
```

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

Create a 3x3 identity matrix

In [8]:
```
id=np.eye(3)
id
```

Out[8]:
```
array([[1., 0., 0.],
       [0., 1., 0.],
       [0., 0., 1.]])
```

Use NumPy to generate a random number between 0 and 1

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

Out[10]:
```
0.48259674616312764
```

Use NumPy to generate an array of 25 random numbers sampled from a standard normal distribution

In [12]:
```
random_numbers = np.random.normal(0, 1, 25)
print(random_numbers)
```

```
[-1.19890496 -0.91568012  0.52514504 -0.3440117  -1.45190724 -0.64407574
 -0.68530531 -0.07752886  0.2360832   0.11701989 -0.7372775  -0.60007743
  1.06961671 -1.53285789  0.21046886 -0.58215235  0.89202933  0.9836785
 -0.5074968  -1.05441372  1.37045741  0.00518766 -1.55933551  1.66516686
  1.78825644]
```

Create the following matrix:

In [13]:
```
sequence = np.arange(0.01, 1.01, 0.01)
result_array = sequence.reshape(10, 10)
print(result_array)
```

```
[[0.01 0.02 0.03 0.04 0.05 0.06 0.07 0.08 0.09 0.1 ]
 [0.11 0.12 0.13 0.14 0.15 0.16 0.17 0.18 0.19 0.2 ]
 [0.21 0.22 0.23 0.24 0.25 0.26 0.27 0.28 0.29 0.3 ]
 [0.31 0.32 0.33 0.34 0.35 0.36 0.37 0.38 0.39 0.4 ]
 [0.41 0.42 0.43 0.44 0.45 0.46 0.47 0.48 0.49 0.5 ]
 [0.51 0.52 0.53 0.54 0.55 0.56 0.57 0.58 0.59 0.6 ]
 [0.61 0.62 0.63 0.64 0.65 0.66 0.67 0.68 0.69 0.7 ]
 [0.71 0.72 0.73 0.74 0.75 0.76 0.77 0.78 0.79 0.8 ]
 [0.81 0.82 0.83 0.84 0.85 0.86 0.87 0.88 0.89 0.9 ]
 [0.91 0.92 0.93 0.94 0.95 0.96 0.97 0.98 0.99 1.  ]]
```

Create an array of 20 linearly spaced points between 0 and 1:

In [14]:
```
ly = np.linspace(0, 1, 20)
print(ly)
```

```
[0.         0.05263158 0.10526316 0.15789474 0.21052632 0.26315789
 0.31578947 0.36842105 0.42105263 0.47368421 0.52631579 0.57894737
 0.63157895 0.68421053 0.73684211 0.78947368 0.84210526 0.89473684
 0.94736842 1.        ]
```

Numpy Indexing and Selection

Now you will be given a few matrices, and be asked to replicate the resulting matrix outputs:

In [15]:
```
mat = np.arange(1,26).reshape(5,5)
mat
```

Out[15]:
```
array([[ 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]])
```

In [0]:
```
# WRITE CODE HERE THAT REPRODUCES THE OUTPUT OF THE CELL BELOW
# BE CAREFUL NOT TO RUN THE CELL BELOW, OTHERWISE YOU WON'T
# BE ABLE TO SEE THE OUTPUT ANY MORE
```

In [16]:
```
a=np.array([[12,13,14,15],[17,18,19,20],[22,23,24,25]])
a
```

Out[16]:
```
array([[12, 13, 14, 15],
       [17, 18, 19, 20],
       [22, 23, 24, 25]])
```

In [0]:
```
# WRITE CODE HERE THAT REPRODUCES THE OUTPUT OF THE CELL BELOW
# BE CAREFUL NOT TO RUN THE CELL BELOW, OTHERWISE YOU WON'T
# BE ABLE TO SEE THE OUTPUT ANY MORE
```

In [17]:
```
a[1][3]
```

Out[17]:
```
20
```

In [0]:
```
# WRITE CODE HERE THAT REPRODUCES THE OUTPUT OF THE CELL BELOW
# BE CAREFUL NOT TO RUN THE CELL BELOW, OTHERWISE YOU WON'T
# BE ABLE TO SEE THE OUTPUT ANY MORE
```

In [18]:
```
output_array = np.array([[2], [7], [12]])
print(output_array)
```

```
[[ 2]
 [ 7]
 [12]]
```

In [0]:
```
# WRITE CODE HERE THAT REPRODUCES THE OUTPUT OF THE CELL BELOW
# BE CAREFUL NOT TO RUN THE CELL BELOW, OTHERWISE YOU WON'T
# BE ABLE TO SEE THE OUTPUT ANY MORE
```

In [19]:
```
c=np.array([21,22,23,24,25])
c
```

Out[19]:
```
array([21, 22, 23, 24, 25])
```

In [0]:
```
# WRITE CODE HERE THAT REPRODUCES THE OUTPUT OF THE CELL BELOW
# BE CAREFUL NOT TO RUN THE CELL BELOW, OTHERWISE YOU WON'T
# BE ABLE TO SEE THE OUTPUT ANY MORE
```

In [20]:
```
d=np.array([[16,17,18,19,20],[21,22,23,24,25]])
d
```

Out[20]:
```
array([[16, 17, 18, 19, 20],
       [21, 22, 23, 24, 25]])
```

Now do the following

Get the sum of all the values in mat

In [21]:
```
mat.sum()
```

Out[21]:
```
325
```

Get the standard deviation of the values in mat

In [22]:
```
dev = np.std(mat)
dev
```

Out[22]:
```
7.211102550927978
```

Get the sum of all the columns in mat

In [23]:
```
col_sum = np.sum(mat,axis=0)
col_sum
```

Out[23]:
```
array([55, 60, 65, 70, 75])
```