NumPy Exercises

Import NumPy as np In

[1]:

```
import numpy as npp
```

Create an array of 10 zeros In

[4]:

```
z1=np.zeros(10)
z1
```

Out[4]: array([0., 0., 0., 0., 0., 0., 0., 0.,

0., 0.])

Create an array of 10 ones In

[0]:

```
z=np.ones(10)
z
```

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

1., 1.])

Create an array of 10 fives In

[7]:

```
z3=np.full(10,5.0)
z3
```

Out[7]: array([5., 5., 5., 5., 5., 5., 5., 5.,

5., 5.])

Create an array of the integers from 10 to 50

In [10]:

```
a=np.arange(10,51)
a
```

Out[10]:

```
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 [16]:

```
'''
el=[]
for i in a:
    if i%2==0:
        el.append(i)
el_arr=np.array(el)
el_arr
'''
ev_arr=np.arange(10,51,2)
ev_arr
```

Out[16]:

```
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

[19]:

```
a1=np.array([[0,1,2,],[3,4,5],[6,7,8]])
a1
```

Out[19]:

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

Create a 3x3 identity matrix In

[21]:

```
a2=np.eye(3)
a2
```

Out[21]:

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

Use NumPy to generate a random number between 0 and 1 In

[28]:

```
ran_num=np.random.rand()
ran_num
```

Out[28]:

```
0.9483808282587929
```

Use NumPy to generate an array of 25 random numbers sampled from a standard normal distribution In [29]:

```
a=np.random.randn(25)
a
```

Out[29]:

```
array([-1.59708033,  0.63240268, -0.45939039, -0.60963869, -1.35633054,
       -0.71346668, -0.1903171 , -0.24412923,  1.95484375,  0.3337913 ,
       -1.30823977,  0.58113653,  0.29769696,  1.5221738 ,  1.84084109,
       -0.93919947,  0.91771739, -0.34159515, -0.67488164,  0.90681744,
        1.11333238, -1.05780533,  0.69298557,  0.79006997, -0.66977284])
```

Create the following matrix:

In [30]:

```
ar=np.arange(0.01,1.0,0.01)
ar
```

Out[30]:

```
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])
```

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

In [31]:

```
la=np.linspace(0,1,20)
la
```

Out[31]:

```
array([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 [51]:

```
mat = np.arange(1,26).reshape(5,5)
mat
```

Out[51]:

```
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 [52]:

```
mat[2:6,1:6]
```

Out[52]:

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

In [41]:

```
# 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 [61]:

```
mat[3:4,4:6]
```

Out[61]:

```
array([[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 [66]:

```
mat[0:3,1:2]
```

Out[66]:

```
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 [69]:

```
mat[4:6,0:6]
```

Out[69]:

```
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 [70]:

```
mat[3:6,0:6]
```

Out[70]:

```
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

[73]:

```
sum1=np.sum(mat)
sum1
```

Out[73]:

```
325
```

Get the standard deviation of the values in mat In

[76]:

```
sd=np.std(mat)
sd
```

Out[76]:

7.211102550927978

Get the sum of all the columns in mat In

[78]:

```
col_sum=np.sum(mat,axis=0)
col_sum
```

Out[78]: array([55, 60, 65,

70, 75]) Type *Markdown* and

LaTeX: $\square^2$