NumPy Exercises

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.

AI AND ML MORNING SLOT 10AM TO 12 PM

ASSIGNMNET-1(01-09-2023)

Name: srikal kakula

Reg.no: 21BCE7457

Mail id: srikalkakula@gmail.com

University mail id:srikal.21bce7457@Vitapstudent.ac.in

Import NumPy as np

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

Create an array of 10 zeros

In [0]:
```

```

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

In [2]:
```
z0=np.zeros(10)
z0
```

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

Create an array of 10 ones

In [0]:
```

```

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

In [3]:
```
z1=np.ones(10)
z1
```

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

Create an array of 10 fives

In [0]:
```

```

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

In [4]:
```
z5=np.full(10,5.0)
z5
```

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

Create an array of the integers from 10 to 50

In [0]:
```
```

Out[0]:
```
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])
```

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

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 [0]:
```
```

Out[0]:
```
array([10, 12, 14, 16, 18, 20, 22, 24, 26, 28, 30, 32, 34, 36, 38, 40, 42,
       44, 46, 48, 50])
```

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

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 [0]:
```
```

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

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

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

Create a 3x3 identity matrix

In [0]:
```
```

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

In [8]:
```
a2=np.eye(3)
a2
```

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

Use NumPy to generate a random number between 0 and 1

In [0]:

Out[0]:
```
array([ 0.42829726])
```

In [9]:
```
ran_num=np.random.rand()
ran_num
```

Out[9]:
```
0.5258450236055247
```

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

In [0]:

Out[0]:
```
array([ 1.32031013,  1.6798602 , -0.42985892, -1.53116655,  0.85753232,
        0.87339938,  0.35668636, -1.47491157,  0.15349697,  0.99530727,
       -0.94865451, -1.69174783,  1.57525349, -0.70615234,  0.10991879,
       -0.49478947,  1.08279872,  0.76488333, -2.3039931 ,  0.35401124,
       -0.45454399, -0.64754649, -0.29391671,  0.02339861,  0.38272124])
```

In [10]:
```
a=np.random.randn(25)
a
```

Out[10]:
```
array([ 0.15610585,  0.25283433,  0.31746579, -0.25534745,  0.59874684,
       -0.89898983, -0.1963885 ,  0.51139689,  0.80836141,  0.97359719,
        0.69308413, -0.55256524,  0.8917299 ,  0.4442093 ,  0.23144886,
       -0.30295675,  0.33286162, -0.38102484,  1.62447541,  0.04365653,
       -0.56713725, -0.91418852,  1.22684553, -2.10886193, -0.0696941 ])
```

Create the following matrix:

In [0]:

Out[0]:
```
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.  ]])
```

In [11]:
```
arr=np.arange(0.01,1.0,0.01)
arr
```

Out[11]:
```
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 [0]:

Out[0]:
```
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.        ])
```

In [12]:
```
la=np.linspace(0,1,20)
la
```

Out[12]:
```
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 [0]:
```
mat = np.arange(1,26).reshape(5,5)
mat
```

Out[0]:
```
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 [13]:
```
mat = np.arange(1,26).reshape(5,5)
mat
```

Out[13]:
```
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 [0]:
```

```

Out[0]:
```
array([[12, 13, 14, 15],
       [17, 18, 19, 20],
       [22, 23, 24, 25]])
```

In [14]:
```
mat[2:6,1:6]
```

Out[14]:
```
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 [0]:
```

```

Out[0]:
```
20
```

In [15]:
```
mat[3:4,4:6]
```

Out[15]:
```
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 [0]:
```

```

Out[0]:
```
array([[ 2],
       [ 7],
       [12]])
```

In [16]:
```
mat[0:3,1:2]
```

Out[16]:
```
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 [0]:
```

```

Out[0]:
```
array([21, 22, 23, 24, 25])
```

In [17]:
```
mat[4:6,0:6]
```

Out[17]:
```
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 [0]:
```

```

Out[0]:
```
array([[16, 17, 18, 19, 20],
       [21, 22, 23, 24, 25]])
```

In [18]:
```
mat[3:6,0:6]
```

Out[18]:
```
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 [0]:
```

```

Out[0]:
```
325
```

In [19]:
```
sum1=np.sum(mat)
sum1
```

Out[19]:
```
325
```

Get the standard deviation of the values in mat

In [0]:

```
```

Out[0]:

```
7.2111025509279782
```

In [20]:

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

Out[20]:

```
7.211102550927978
```

Get the sum of all the columns in mat

In [0]:

```
```

Out[0]:

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

In [21]:

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

Out[21]:

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