NumPy Exercises (kartik-A9)

Import NumPy as np

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

Create an array of 10 zeros

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

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

Create an array of 10 ones

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

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

Create an array of 10 fives

In [4]:
```
np.fives(10)
```

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]:
```
np.array(10,50)
```

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]:
```
np.arange(10,50,2)
```

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]:
```
x=np.arange(0,9)
x.reshape((3,3))
```

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

Create a 3x3 identity matrix

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

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

Use NumPy to generate a random number between 0 and 1

In [15]:
```
np.random.rand(1)
```

Out[15]:
```
array([ 0.42829726])
```

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

In [33]:
```
np.random.random_sample((25))
```

Out[33]:
```
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])
```

Create the following matrix:

In [35]:
```
x=np.arange(0.01,1.01,0.01)
x
y=x.reshape(10,10)
y
```

Out[35]:
```
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 [36]:
```
x=np.linspace(0,1,20)
x
```

Out[36]:
```
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 [38]:
```
mat = np.arange(1,26).reshape(5,5)
mat
```

Out[38]:
```
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 [39]:
```
# 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 [40]:
```
x=np.arange(12,26)
new=np.delete(x,(4,9))
new.reshape((3,4))
```

Out[40]:
```
array([[12, 13, 14, 15],
       [17, 18, 19, 20],
       [22, 23, 24, 25]])
```

In [29]:
```
# 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 [41]:
```
new[1][3]
```

Out[41]:
```
20
```

In [30]:
```
# 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 [42]:
```
x=np.array([2,7,12]).reshape(3,1)
x
```

Out[42]:
```
array([[ 2],
       [ 7],
       [12]])
```

In [31]:
```
# 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 [46]:
```
x=np.arange(21,26)
x
```

Out[46]:
```
array([21, 22, 23, 24, 25])
```

In [32]:
```
# 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 [49]:
```
x=np.arange(16,26).reshape(2,5)
x
```

Out[49]:
```
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 [50]:
```
x=np.arange(0,100)
s=np.sum(x)
x = x/s*325
x
np.sum(x)
```

Out[50]:
```
325
```

Get the standard deviation of the values in mat

In [51]:
```
st=np.std(new)   # first we find std of new matrix
st # then i print its std  :4.232808366400098
new_std=new/st*7.21110255092797 # divide new by ratio of desired and actual st
new_std
np.std(new_std) # now new std will have desired std
```

Out[51]:
```
7.2111025509279782
```

Get the sum of all the columns in mat

In [1]:
```
# first , create a random matrix of integers .
new= np.random.randint(0, 10, size=(3, 5))

asum=np.sum(new,axis=0)  # then finding sum of all coloumns in array

#changing row 1
r1=new[0:3,0]
nr1=r1/asum[0]*55

# changing row 2
r2=new[0:3,1]
nr2=r2/asum[1]*60

# changing row 3
r3=new[0:3,2]
nr3=r3/asum[2]*65

#changing row 4
r4=new[0:3,3]
nr4=r4/asum[3]*70

#changing row 5
r5=new[0:3,4]
nr5=r5/asum[4]*75

# now printing the sum //
new_mat= np.column_stack((nr1,nr2,nr3,nr4,nr5 )) # joining all columns back int
np.sum(new_mat,axis=0)  # now finding the sum of all coloums ..with all rows i
```

Out[53]:
```
array([55, 60, 65, 70, 75])
```

In [ ]:
```

```