NumPy Assignment

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

Q1.

Create a null vector of size 10 but the fifth value is 1

In [2]:
```
a= np.zeros(10)
a[4] = 1
a
```

Out[2]:
```
array([0., 0., 0., 0., 1., 0., 0., 0., 0., 0.])
```

Q2.

Create a vector with values ranging from 10 to 49

In [3]:
```
a=np.arange(10,50)
a
```

Out[3]:
```
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])
```

Q3.

Reverse a vector (first element becomes last)

In [4]:
```
a=a[::-1]
a
```

Out[4]:
```
array([49, 48, 47, 46, 45, 44, 43, 42, 41, 40, 39, 38, 37, 36, 35, 34, 33,
       32, 31, 30, 29, 28, 27, 26, 25, 24, 23, 22, 21, 20, 19, 18, 17, 16,
       15, 14, 13, 12, 11, 10])
```

Q4.

Create a 3x3 matrix with values ranging from 0 to 8 `hint: reshape`

In [5]:
```
a=np.arange(0,9)
b=a.reshape([3,3])
b
```

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

Q5.

Find indices of non-zero elements from [1,2,0,0,4,0] `hint: np.nonzero`

In [6]:
```
a=np.nonzero([1,2,0,0,4,0])
a
```

Out[6]:
```
(array([0, 1, 4], dtype=int64),)
```

Q6.

Create a 3x3x3 array with random values `hint: np.random.random`

In [7]:
```
b=np.random.random([3,3,3])
b
```

Out[7]:
```
array([[[0.250978  , 0.21027033, 0.46136938],
        [0.74568807, 0.15067108, 0.6954368 ],
        [0.50398082, 0.76690294, 0.14723029]],

       [[0.92230913, 0.04445247, 0.28032136],
        [0.20353078, 0.11474088, 0.12665368],
        [0.25481191, 0.87429448, 0.17657862]],

       [[0.34821689, 0.99185715, 0.9709972 ],
        [0.08296744, 0.54523818, 0.10162944],
        [0.93803362, 0.6012834 , 0.25661411]]])
```

Q7.

Create a 10x10 array with random values and find the minimum and maximum values `hint: min, max`

In [8]:
```
b=np.random.random([10,10])
print(b.max())
print(b.min())
print(b.mean())
```

```
0.99421976957989
0.017328309402178665
0.5191858049984805
```

Q8.

Create a random vector of size 30 and find the mean value `hint: mean`

In [9]:
```
a=np.arange(30)
b=a.mean()
b
```

Out[9]:
```
14.5
```

Q9.

Create a 2d array with 1 on the border and 0 inside `hint: array[1:-1, 1:-1]`

In [10]:
```
x = np.ones((4,4))
x[1:-1,1:-1] = 0
x
```

Out[10]:
```
array([[1., 1., 1., 1.],
       [1., 0., 0., 1.],
       [1., 0., 0., 1.],
       [1., 1., 1., 1.]])
```

Q10.

10. Normalize a 5x5 random matrix `hint: (x -mean)/std`

In [11]:
```
x=np.random.random([5,5])
m=x.mean()
s=x.std()
x=((x-m)/s)
x
```

Out[11]:
```
array([[-1.18803342, -1.23041605, -0.75605151,  0.68895608,  2.07064328],
       [-0.87942495, -1.47960093, -0.57072835,  1.17800572,  0.75144319],
       [-0.49123904,  0.7959511 , -1.20106923,  1.31154758,  0.82936831],
       [-0.03390826,  0.27589882, -1.04468793,  1.3065326 ,  0.07071726],
       [-1.46687778, -0.4580565 ,  0.05862311,  0.25177068,  1.21063622]])
```

Q11.

Multiply a 5x3 matrix by a 3x2 matrix (real matrix product)

In [12]:
```
x=np.random.random([5,3])
y=np.random.random([3,2])
z=x@y
z
```

Out[12]:
```
array([[0.68654119, 0.55538974],
       [0.91720878, 0.30974857],
       [0.46871398, 0.4834697 ],
       [0.3528873 , 0.65598819],
       [0.6583626 , 0.27785005]])
```

Q12.

Given a 1D array, negate all elements which are between 3 and 8, in place.

In [13]:
```
Z = np.arange(11)
Z
Z[(3 < Z) & (Z <= 8)] *= -1
Z
```

Out[13]:
```
array([ 0,  1,  2,  3, -4, -5, -6, -7, -8,  9, 10])
```

Q13.

Find the eigenvalues and eigenvectors of a square matrix.
`hint: np.linalg.eig`

In [14]:
```
import numpy as np
a = np.array([[3, 1], [2, 2]])
np.linalg.eig(a)
```

Out[14]:
```
(array([4., 1.]),
 array([[ 0.70710678, -0.4472136 ],
        [ 0.70710678,  0.89442719]]))
```

Q14.

Find the inverse of a square matrix.
`hint: np.linalg.inv`

In [19]:
```
ar=np.random.random((4,4))
np.linalg.inv(ar)
```

Out[19]:
```
array([[  5.00452968, -10.3424529 ,  16.55826292,  -8.75313217],
       [  0.39183492,  -0.87846095,   4.27107968,  -2.34074096],
       [ -2.46392865,   8.85076922, -15.41383336,   6.78334774],
       [ -0.08715052,  -1.06419128,   0.39008852,   1.59525206]])
```

In [ ]: