Q1-> Create a null vector of size 10 but the fifth value which is 1 (★☆☆)

```
# QUESTION 1
import numpy as np
a = np.zeros(10)   #main thing
print(a)
print("update 5th element to 1")
a[5] = 1
print(a)
```

```
[0. 0. 0. 0. 0. 0. 0. 0. 0. 0.]
update 5th element to 1
[0. 0. 0. 0. 0. 1. 0. 0. 0. 0.]
```

2. Create a vector with values ranging from 10 to 49

```
# QUESTION 2
a = np.arange(10,49)    #main thing
print("Original vector: ")
print(a)
```

```
Original vector:
[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]
```

3. Reverse a vector (first element becomes last)

```
# Question 3
x = np.arange(12, 20)
print("Original array:")
print(x)
print("Reverse array:")
x = x[::-1]
print(x)
```

```
Original array:
[12 13 14 15 16 17 18 19]
Reverse array:
[19 18 17 16 15 14 13 12]
```

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

```
# Question 4
#how to use flip
x =  np.arange(0, 8)
print(x)
x = np.flip
print(x)
```

```
[0 1 2 3 4 5 6 7]
<function flip at 0x7c0d107312d0>
```

5. Find indices of non-zero elements from [1,2,0,0,4,0]

```
# Question 5
arr = np.array([[1, 2, 0, 0, 4, 0]])

print ("Input  array : \n", arr)

out_tpl = np.nonzero(arr)
print ("Indices of non zero elements : ", out_tpl)
```

```
Input  array :
 [[1 2 0 0 4 0]]
Indices of non zero elements :  (array([0, 0, 0]), array([0, 1, 4]))
```

6. Create a 3x3 identity matrix

```
# Question 6
import numpy as np
a = np.eye(3)
print(a)
```

```
[[1. 0. 0.]
 [0. 1. 0.]
 [0. 0. 1.]]
```

7. Create a 3x3x3 array with random values

```
# QUESTION 7
import numpy as np
x= np.random.random((3,3,3))
print(x)
```

```
[[[0.41387864 0.26031502 0.01636567]
  [0.4692166  0.1020946  0.09455012]
  [0.37924782 0.07342208 0.57283107]]

 [[0.07281738 0.4966104  0.09412524]
  [0.86557783 0.24452944 0.15501885]
  [0.84810286 0.35469765 0.99930264]]

 [[0.8299177  0.26850112 0.09401298]
  [0.06091298 0.32933583 0.38955136]
  [0.4610016  0.99775532 0.59726457]]]
```

8. Create a 10x10 array with random values and find the minimum and maximum values

```
# Question 8
import numpy as np
x = np.random.random((10,10))
print("Original Array:")
print(x)
xmin, xmax = x.min(), x.max()
print("Minimum and Maximum Values:")
print(xmin, xmax)
```

```
Original Array:
[[0.1978596  0.05132007 0.93138042 0.77704272 0.2696221  0.34972261
  0.49997206 0.12626462 0.00154513 0.7592794 ]
 [0.14188369 0.50140765 0.29715195 0.50022681 0.81128163 0.85312077
  0.46531029 0.10516874 0.50353979 0.28951298]
 [0.95056371 0.93822617 0.70235653 0.80056629 0.86403222 0.70123721
  0.65217115 0.33481278 0.79516663 0.56247726]
 [0.26113865 0.52521586 0.20170587 0.63835616 0.61853267 0.72681869
  0.13747446 0.02462499 0.57972693 0.23218284]
 [0.53577469 0.7971974  0.14105529 0.91466426 0.17934351 0.43180858
  0.82948748 0.28146642 0.09002346 0.76370069]
 [0.38092565 0.76164647 0.87412517 0.11981419 0.3432374  0.35037731
  0.9134666  0.82120737 0.06008297 0.99293106]
 [0.18219733 0.39346481 0.16163615 0.65149688 0.6344811  0.84844119
  0.3260104  0.30088347 0.52439711 0.99485253]
 [0.74458979 0.61400395 0.31224546 0.15210342 0.64992819 0.21738637
  0.69381985 0.26333308 0.87001456 0.38962699]
 [0.42091105 0.72750295 0.77842881 0.24429506 0.2745828  0.16290181
  0.01035131 0.49025782 0.74861342 0.89709985]
 [0.36745666 0.08550592 0.80317259 0.83432774 0.72637416 0.34334499
  0.43951304 0.07085603 0.70631095 0.15068813]]
Minimum and Maximum Values:
0.001545129926052824 0.9948525283731123
```

9. Create a random vector of size 30 and find the mean value

```
# Question 9
import numpy as np
x = np.random.random(30)
print("Original array:")
print(x)
#mean value
```

```
Original array:
[0.60606051 0.14972189 0.21709637 0.84058311 0.69661229 0.78059306
 0.63229099 0.90948012 0.27757938 0.89617014 0.99653091 0.29377059
 0.526636   0.95325853 0.98117249 0.68446769 0.63061962 0.24802968
 0.77886573 0.86109143 0.32121744 0.29038062 0.71541179 0.82364819
 0.41729262 0.47503043 0.73714474 0.47585049 0.60313099 0.35959291]
```

10. Create a 2d array with 1 on the border and 0 inside

```
#Question 10
import numpy as np
x = np.ones((5,5))
print("Original array:")
print(x)
print("1 on the border and 0 inside in the array")
x[1:-1,1:-1] = 0
print(x)
```

```
Original array:
[[1. 1. 1. 1. 1.]
 [1. 1. 1. 1. 1.]
 [1. 1. 1. 1. 1.]
 [1. 1. 1. 1. 1.]
 [1. 1. 1. 1. 1.]]
1 on the border and 0 inside in the array
[[1. 1. 1. 1. 1.]
 [1. 0. 0. 0. 1.]
 [1. 0. 0. 0. 1.]
 [1. 0. 0. 0. 1.]
 [1. 1. 1. 1. 1.]]
```

11. How to add a border (filled with 0's) around an existing array?

```
#Question 11
# Importing the NumPy library with an alias 'np'
import numpy as np

# Creating a 3x3 NumPy array filled with ones
x = np.ones((3, 3))

# Printing the original array
print("Original array:")
print(x)

# Modifying the array 'x' to set 0s on the border and 1s inside the array using the np.pad function
print("0 on the border and 1 inside in the array")
x = np.pad(x, pad_width=1, mode='constant', constant_values=0)

# Printing the desired result
print(x)
```

```
Original array:
[[1. 1. 1.]
 [1. 1. 1.]
 [1. 1. 1.]]
0 on the border and 1 inside in the array
[[0. 0. 0. 0. 0.]
 [0. 1. 1. 1. 0.]
 [0. 1. 1. 1. 0.]
 [0. 1. 1. 1. 0.]
 [0. 0. 0. 0. 0.]]
```

12. Create a 5x5 matrix with values 1,2,3,4 just below the diagonal

```
#Question 12
Z = np.diag(1+np.arange(4),k=-1)
print(Z)
```

```
[[0 0 0 0 0]
 [1 0 0 0 0]
 [0 2 0 0 0]
 [0 0 3 0 0]
 [0 0 0 4 0]]
```

13. Create a 8x8 matrix and fill it with a checkerboard pattern Consider a (6,7,8) shape array, what is the index (x,y,z) of the 100th element?

```
# Question 13
import numpy as np
x = np.ones((3,3))
print(x)
print("Checkerboard pattern:")
x = np.zeros((8,8),dtype=int)
x[1::2,::2] = 1
x[::2,1::2] = 1
print(x)
```

```
[[1. 1. 1.]
 [1. 1. 1.]
 [1. 1. 1.]]
Checkerboard pattern:
[[0 1 0 1 0 1 0 1]
 [1 0 1 0 1 0 1 0]
 [0 1 0 1 0 1 0 1]
 [1 0 1 0 1 0 1 0]
 [0 1 0 1 0 1 0 1]
 [1 0 1 0 1 0 1 0]
 [0 1 0 1 0 1 0 1]
 [1 0 1 0 1 0 1 0]]
```

Double-click (or enter) to edit

```
import numpy as np
print(np.unravel_index(100,(6,7,8)))
```

```
(1, 5, 4)
```

14. Create a checkerboard 8x8 matrix using the tile function

```
#Question 14
import numpy as np
Z = np.tile( np.array([[0,1],[1,0]]), (4,4))
print(Z)
```

```
[[0 1 0 1 0 1 0 1]
 [1 0 1 0 1 0 1 0]
 [0 1 0 1 0 1 0 1]
 [1 0 1 0 1 0 1 0]
 [0 1 0 1 0 1 0 1]
 [1 0 1 0 1 0 1 0]
 [0 1 0 1 0 1 0 1]
 [1 0 1 0 1 0 1 0]]
```

15. Normalize a 5x5 random matrix

```
# QUESTION 15
import numpy as np
x= np.random.random((3,3))
print("Original Array:")
print(x)
xmax, xmin = x.max(), x.min()
print(" max values -->",x.max())
print(" min values -->", x.min())
x = (x - xmin)/(xmax - xmin)
print("After normalization:")
print(x)
```

```
Original Array:
[[0.23876583 0.87436549 0.19046614]
 [0.92049115 0.32005732 0.69055766]
 [0.8096536  0.67128882 0.5620462 ]]
 max values --> 0.9204911499494861
 min values --> 0.19046613836400184
After normalization:
[[0.06616169 0.93681633 0.        ]
 [1.         0.17751609 0.6850334 ]
 [0.84817294 0.65863863 0.50899635]]
```

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

```
# Question 16
import numpy as np
x = np.random.random((5,3))
print("First array:")
print(x)
y = np.random.random((3,2))
print("Second array:")
print(y)
z = np.dot(x, y)
print("Dot product of two arrays:")
print(z)
```

```
First array:
[[0.82342856 0.03206594 0.76621971]
 [0.51474909 0.41592724 0.24465447]
 [0.39626584 0.09298379 0.22489758]
 [0.51650515 0.9142299  0.70272492]
 [0.71574744 0.23838115 0.17418275]]
Second array:
[[0.98118469 0.66716825]
 [0.11625127 0.73597966]
 [0.89517734 0.01561955]]
Dot product of two arrays:
[[1.49756572 0.58493328]
 [0.77242513 0.65335963]
 [0.60094268 0.33632297]
 [1.24213076 1.0284267 ]
 [0.88591699 0.65568831]]
```

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

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

```
# Question 17
import numpy as np
arr = np.arange(10)
arr[4:8] = np.multiply(arr[4:8],-1)
print(arr)
```

```
[ 0  1  2  3 -4 -5 -6 -7  8  9]
```

```
# Question 19
import numpy as np
array1 = np.array([0, 10, 20, 40, 60, 80])
print("Array1: ",array1)
array2 = [10, 30, 40, 60]
print("Array2: ",array2)
print("Common values between two arrays:")
print(np.intersect1d(array1, array2))
```

```
Array1:  [ 0 10 20 40 60 80]
Array2:  [10, 30, 40, 60]
Common values between two arrays:
[10 40 60]
```