In [ ]:
```
# i)    Create a one dimensional array (vector) iterate through all the elements and find the mean of the vector.
# importing numpy
import numpy as np

# creating a one-Dimensional list (Horizontal)
first_list = [2, 3, 5,7,8]

# vector as row using the first list
first_vector = np.array(first_list)
sum =0
for a in first_vector:
  sum += a
mean = sum/len(first_vector)
mean
```

Out[ ]:
```
5.0
```

In [ ]:
```
# ii)   Create a 3X3 array and find the sum of all elements in the second column
three_three =np.array([(1,2,3),(4,5,6),(7,8,9)])
three_three
sum =0
for row in three_three:
  sum += row[1]
sum
```

Out[ ]:
```
15
```

In [ ]:
```
# iii)  Create array using other standard methods like ones and empty by taking the zeros as reference
# ones
ones_array = np.ones((4, 1))
print("Using Ones:\n",ones_array)
print("---------------------------")
# Zeros
zero_array = np.zeros(7)
print("Using Zero:\n",zero_array)
print("---------------------------")
# arrange
arrange_arr  = np.arange(20)
print("Using Arrange:\n",arrange_arr)
print("---------------------------")
```

```
# full
full_arr = np.full((2,5), 3)
print("Using Full:\n",full_arr)
print("---------------------------")
# EYE
eye_arr  = np.eye(3,3)
print("Using eye:\n",eye_arr)
print("---------------------------")
```

```
Using Ones:
 [[1.]
 [1.]
 [1.]
 [1.]]
---------------------------
Using Zero:
 [0. 0. 0. 0. 0. 0. 0.]
---------------------------
Using Arrange:
 [ 0  1  2  3  4  5  6  7  8  9 10 11 12 13 14 15 16 17 18 19]
---------------------------
Using Full:
 [[3 3 3 3 3]
 [3 3 3 3 3]]
---------------------------
Using eye:
 [[1. 0. 0.]
 [0. 1. 0.]
 [0. 0. 1.]]
---------------------------
```

In [ ]:

```
# 2.i)  Get the diagonal elements and calculate the sum of all the diagonal elements.
diag_arr = np.arange(16).reshape((4, 4))

print("Eye Array:\n",diag_arr)
print("-------------------------")
# get daigonal element
diag_elem = diag_arr.diagonal()
print("Diagonale Element: ",diag_elem)
print("-------------------------------")
# sum of diagonal element
sum_diag_elem = np.sum(diag_elem)
print("Sum of Diagonale Element: ",sum_diag_elem)
print("-------------------------------")
```

```
Eye Array:
 [[ 0  1  2  3]
 [ 4  5  6  7]
 [ 8  9 10 11]
 [12 13 14 15]]
-------------------------
Diagonale Element:  [ 0  5 10 15]
-------------------------------
Sum of Diagonale Element:  30
-------------------------------
```

In [ ]:

```
# 2.ii) Calculate the column wise and row wise mean of the created matrix
diag_arr = np.arange(16).reshape((4, 4))
print("Matxix:\n",diag_arr)
row_mean =np.mean(diag_arr,axis=0)
print("-------------------------------")
print("Row Wise Mean:",row_mean)


column_mean =np.mean(diag_arr,axis=1)
print("Column Wise Mean:",column_mean)
print("-------------------------------")
```

```
Matxix:
 [[ 0  1  2  3]
 [ 4  5  6  7]
 [ 8  9 10 11]
 [12 13 14 15]]
-------------------------------
Row Wise Mean: [6. 7. 8. 9.]
Column Wise Mean: [ 1.5  5.5  9.5 13.5]
-------------------------------
```

3.Can we reshape the array into any shape? If not, then what are the cases?

Answer:We can reshape any array into any shape as long as the elements required for reshaping are equal in both shapes. Interestingly, we are allowed to have one “unknown” dimension. What that means is that you don’t have to specify an example number for one of the dimensions in the reshape method.

In [ ]:

```
# 4. Use arange and create a sequence of 6 numbers and try to reshape into all the possible ways and combinations
ar_6=np.arange(6)
print("ar_6: ",ar_6)
print("-----------------------")
```

```
print("Reshape 3*2:\n",ar_6.reshape(2,3))
print("Reshape 2 *3:\n",ar_6.reshape(3,2))
print("Reshape 6 *1\n",ar_6.reshape(1,6))
print("Reshape 1 *6\n",ar_6.reshape(6,-1))
```

```
ar_6:  [0 1 2 3 4 5]
----------------------
Reshape 3*2:
 [[0 1 2]
 [3 4 5]]
Reshape 2 *3:
 [[0 1]
 [2 3]
 [4 5]]
Reshape 6 *1
 [[0 1 2 3 4 5]]
Reshape 1 *6
 [[0]
 [1]
 [2]
 [3]
 [4]
 [5]]
```

In [ ]: