Solution 2

September 10, 2019

0.0.1 Question 1:

Perform the following actions:

1. Use range(25) to create an array with a dimension of 5X5, and use a for loop to calculate the sum of all elements in the diagonal of the array. (25 points)

```
In [1]: import numpy as np
        np.random.seed(89)
        arr1 = np.random.randn(25).reshape(5, 5)
        arr1
```

```
Out[1]: array([[-1.69373003, -0.00212133, -0.42589184, -0.25559046, -0.30577884],
               [ 0.04054968, -0.06835443, -1.2022241 , -0.18253337, -0.87957607],
               [-0.24324777,  0.67403074,  0.34494632, -0.62528484, -0.20645163],
               [-0.87159742, -0.87010581,  0.39814764, -0.66525522,  0.13367724],
               [ 0.78621468, -0.00531721, -0.54656608,  0.55457963,  0.46951714]])
```

```
In [2]: res1 = 0
        for i in range(len(arr1)):
            res1 += arr1[i, i]
        res1
```

```
Out[2]: -1.6128762124920417
```

Differnt approach:

```
In [3]: res1 = 0
        for i in arr1.diagonal():
            res1 += i
        res1
```

```
Out[3]: -1.6128762124920417
```

2. Choose three functions to apply to this array. (25 points)

```
In [4]: np.var(arr1)
```

```
Out[4]: 0.35353162137795124
```

```
In [5]: np.std(arr1, axis = 1)
```

```
Out[5]: array([0.5948308 , 0.49149818, 0.46161693, 0.53525632, 0.47501008])
```

```
In [6]: np.mean(arr1, axis = 0)
```

```
Out[6]: array([-0.39636217, -0.05437361, -0.28631761, -0.23481685, -0.15772243])
```

0.0.2 Question 2:

Perform the following actions: #### 1. Use x = np.random.randint(0, 1000, size = (10, 10)) to generate 10x10 array and use a for loop to find out how many even numbers are in it. (25 points)

```
In [7]: np.random.seed(100)
        arr2 = np.random.randint(0, 1000, size = (10, 10))
        arr2
```

```
Out[7]: array([[520, 792, 835, 871, 855,  79, 944, 906, 350, 948],
               [866,  53, 578, 738, 526, 802, 752, 280, 655, 228],
               [875, 316, 570, 912, 507, 649,  93,  86, 386, 667],
               [876, 900, 415, 897, 141, 757, 723, 612,   4, 603],
               [955, 835, 135,  49, 431, 705, 317, 782, 695, 967],
               [763, 336,   2, 889, 617, 478, 403, 994,  63, 181],
               [283, 824, 238, 369, 926, 944, 303, 679, 877, 806],
               [172, 274, 192, 952, 930, 437, 714, 273, 584, 525],
               [618,  30,  17,  53,  68, 946, 488, 347, 475, 979],
               [693, 846,   0,  13, 185, 460, 362, 131, 582, 643]])
```

Approach 1:

```
In [8]: evensum1 = 0
        for i in range(arr2.shape[0]):
            for j in range(arr2.shape[1]):
                if arr2[i, j]%2 == 0:
                    evensum1 += 1
        evensum1
```

```
Out[8]: 50
```

Approach 2:

```
In [9]: evensum2 = 0
        for row in arr2:
            for element in row:
                if element %2 == 0:
                    evensum2 += 1
        evensum2
```

```
Out[9]: 50
```

Approach 3: Please refer to https://docs.scipy.org/doc/numpy-1.15.0/reference/generated/numpy.nditer.html about how ndeiter function works.

```
In [10]: evensum2 = 0
         for value in np.nditer(arr2):
             if value %2 == 0:
                 evensum2 += 1
         evensum2
```

```
Out[10]: 50
```

2. Randomly generate a 8x9 array from a normal distribute with mean = 1, sigma = 0.5. Calculate the mean of elements whose indices i and j has a relation of (i+j)%5 == 0

Approach 1

```
In [11]: np.random.seed(200)
         arr3 = np.random.normal(1, 0.5, 72)
         arr3 = arr3.reshape(8, 9)
```

```
In [12]: res2 = 0
         count = 0
         for i in range(arr3.shape[0]):
             for j in range(arr3.shape[1]):
                 if (i+j)%5 == 0:    # This is different from i+j % 5 == 0, add print(i, j) to
                     print(i, j)
                     res2 += arr3[i, j]
                     count += 1
         res2/count
```

```
0 0
0 5
1 4
2 3
2 8
3 2
3 7
4 1
4 6
5 0
5 5
6 4
7 3
7 8
```

```
Out[12]: 1.1771991417476577
```

Approach 2: Please refer to https://docs.scipy.org/doc/numpy-1.15.0/reference/generated/numpy.ndenumerate.html about how ndenumerate function works.

```
In [13]: res2 = 0
         count = 0
         for indexs, value in np.ndenumerate(arr3):
             if (indexs[0] + indexs[1]) % 5 == 0:
                 res2 += value
                 count += 1

         res2/count
```

```
Out[13]: 1.1771991417476577
```

Approach 3:

```
In [14]: # Create an array to record column index for each element.
         ColumnArray = np.repeat([np.arange(0, 9),], 8, axis = 0)
         ColumnArray
```

```
Out[14]: array([[0, 1, 2, 3, 4, 5, 6, 7, 8],
                [0, 1, 2, 3, 4, 5, 6, 7, 8],
                [0, 1, 2, 3, 4, 5, 6, 7, 8],
                [0, 1, 2, 3, 4, 5, 6, 7, 8],
                [0, 1, 2, 3, 4, 5, 6, 7, 8],
                [0, 1, 2, 3, 4, 5, 6, 7, 8],
                [0, 1, 2, 3, 4, 5, 6, 7, 8],
                [0, 1, 2, 3, 4, 5, 6, 7, 8]])
```

```
In [15]: # Create an array to record row index for each element.
         RowArray = np.repeat([np.arange(0,8),], 9, axis = 0).transpose()
         RowArray
```

```
Out[15]: array([[0, 0, 0, 0, 0, 0, 0, 0, 0],
                [1, 1, 1, 1, 1, 1, 1, 1, 1],
                [2, 2, 2, 2, 2, 2, 2, 2, 2],
                [3, 3, 3, 3, 3, 3, 3, 3, 3],
                [4, 4, 4, 4, 4, 4, 4, 4, 4],
                [5, 5, 5, 5, 5, 5, 5, 5, 5],
                [6, 6, 6, 6, 6, 6, 6, 6, 6],
                [7, 7, 7, 7, 7, 7, 7, 7, 7]])
```

```
In [16]: # find the position with (i+j)%5 == 0
         Sumindex = (ColumnArray + RowArray) % 5 == 0
         Sumindex
```

```
Out[16]: array([[ True, False, False, False, False,  True, False, False, False],
                [False, False, False, False,  True, False, False, False, False],
```

```
               [False, False, False,  True, False, False, False, False,  True],
               [False, False,  True, False, False, False, False,  True, False],
               [False,  True, False, False, False, False,  True, False, False],
               [ True, False, False, False, False,  True, False, False, False],
               [False, False, False, False,  True, False, False, False, False],
               [False, False, False,  True, False, False, False, False,  True]])
```

In [17]:
```
# Select those elements.
arr3[Sumindex]
```

Out[17]:
```
array([0.27452588, 0.98352516, 1.50867448, 1.60749202, 1.10926891,
       1.1337302 , 0.66723288, 1.30218793, 1.85553437, 0.87955861,
       2.09304747, 0.8041994 , 0.98495979, 1.27685089])
```

In [18]:
```
# Result
np.mean(arr3[Sumindex])
```

Out[18]:
```
1.1771991417476577
```