



Indian Institute of Technology Mandi
भारतीय प्रौद्योगिकी संस्थान मण्डी
IC252-Data Science 2
Assignment– 0

Name: Vinamra Garg Roll No: B23302 Department: SCEE(Electrical Engineering)

Problem 1

Find out the size of array 'a', 'b', and 'c'. In the following code block.

Listing 1: Code for problem

```
1 import numpy as np
2
3 a = np . array (42)
4 b = np . array ([1 , 2, 3 , 4 , 5])
5 c = np . array ([[1 , 2, 3] , [4 , 5, 6]])
6
7 #solution
8 def arr_size(arr):
9     return arr.nbytes/4
10 # Print size of all arrays
11 print("size of array a = ",arr_size(a))
12 print("size of array b = ",arr_size(b))
13 print("size of array c = ",arr_size(c))
14
15 #alternate approach#
16
17 # Print size of all arrays
18 print("size of array a = ",np.size(a))
19 print("size of array b = ",np.size(b))
20 print("size of array c = ",np.size(c))
21
22 #to find shape of array 'c' as it is a 2D array
23 print("shape (rows,columns) of array c = ",np.shape(c))
```

Output:

```
size of array a = 1.0
size of array b = 5.0
size of array c = 6.0
size of array a = 1
size of array b = 5
size of array c = 6
shape (rows,columns) of array c = (2, 3)
```

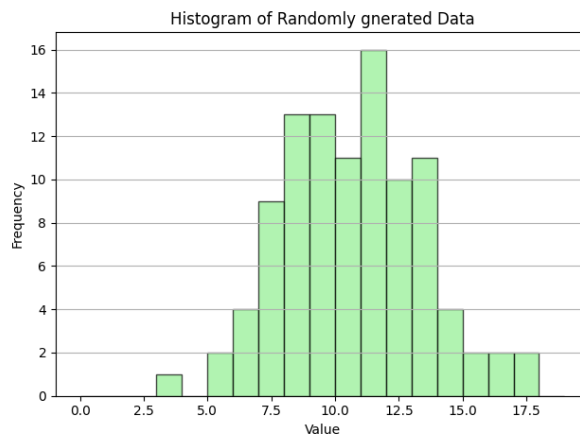
Problem 2

Bins! Yes binning might be tricky if you want to plot meaningful plots, in the below histogram choose an appropriate binning size, experiment with the bins style, location, borders and colors. Explore the np.arange function. Add meaningful labels to both axes.

Listing 2: Code for problem

```
1
2
3 import numpy as np
4 import matplotlib.pyplot as plt
5
6 hbins = np.arange(0,20,1)
7 plt.hist(val, bins=hbins, color='lightgreen', edgecolor='black', alpha=
    =0.7) #histogram plot
8 #defining labels and title
9 plt.xlabel('Value')
10 plt.ylabel('Frequency')
11 plt.title('Histogram of Randomly generated Data')
12 plt.grid(1,axis='y')
13 plt.tight_layout() #adjust layout
14
15 # save/show plot
16 plt.savefig("q2.png")
17 plt.show()
```

Output:



Problem 3

Array addressing and slicing. As the dimensions of the array increase it becomes very important how one indexes those arrays, slicing becomes important when one wants to get some desired subpart of a whole array. This Question has two parts.

3.1. Print the sum of left diagonal and right diagonal of a square matrix.

Listing 3: Code for problem

```

1 import numpy as np
2 import matplotlib . pyplot as plt
3
4 mat = np . array ([[1 ,3 ,4 ,5 ,2] ,[1 ,5 ,2 ,4 ,3] ,[5 ,2 ,3 ,4 ,1 ,] ,[1↵
    ,4 ,2 ,6 ,9] ,[4 ,5 ,2 ,1 ,7]])
5 def left_diagonal_sum ( mat : np . ndarray ) ->float :
6     sum=0
7     for i in range(mat.shape[0]):
8         sum+=mat[i][i]
9     # sum=np.trace(mat)    #alternate approach
10    return sum
11
12 def right_diagonal_sum ( mat : np . ndarray ) ->float :
13     sum=0
14     r,c=mat.shape
15     for i in range(r):
16         sum+=mat[i][r-i-1]
17     # flip_mat = np.fliplr(mat)    #alternate approach
18     # sum = np.trace(flip_mat)
19     return sum
20

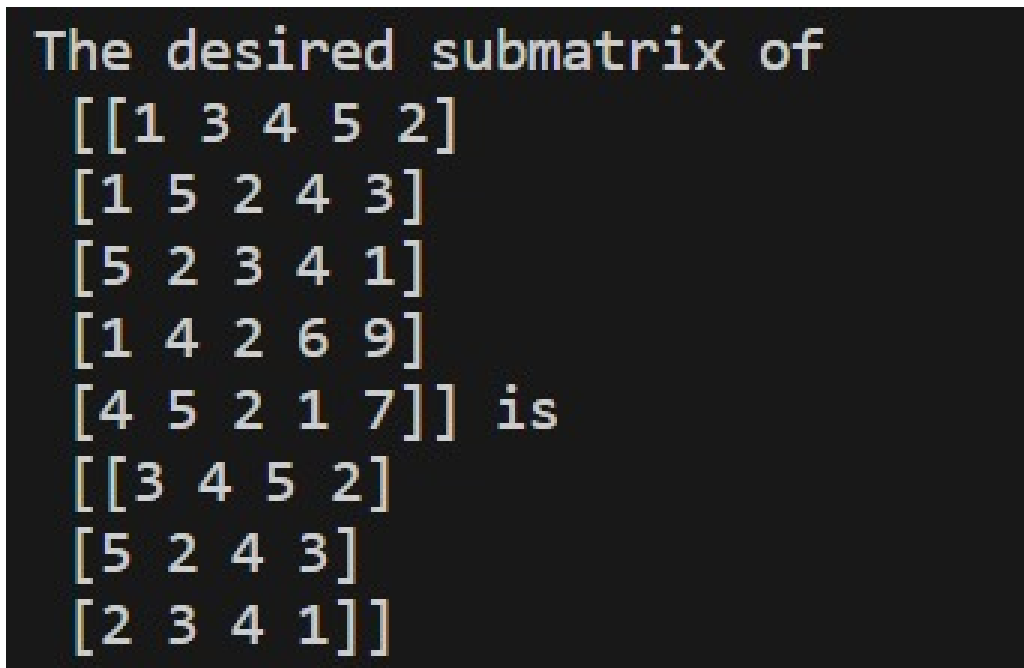
```

```

21 print (f'Left Diagonal Sum of { mat =} is { left_diagonal_sum (mat)}')
22 print (f'Right Diagonal Sum of { mat =} is { right_diagonal_sum ( mat )}')

```

Output:



```

The desired submatrix of
[[1 3 4 5 2]
 [1 5 2 4 3]
 [5 2 3 4 1]
 [1 4 2 6 9]
 [4 5 2 1 7]] is
[[3 4 5 2]
 [5 2 4 3]
 [2 3 4 1]]

```

3.2. From the matrix of previous part, print the 3x4 sub-matrix from 1st row 2nd column to 3rd row 5th column. Make use of the array slicing operations preferably , use of for loops is discouraged.

Listing 4: Code for problem

```

1
2 import numpy as np
3
4 def submatrix_3x4 ( mat : np . ndarray ) -> np . ndarray :
5     sub_matrix=mat[0:3,1:5]
6     return sub_matrix
7
8 print (f'The desired submatrix of \n { mat } is \n { submatrix_3x4 ( mat )}↵
    ')

```

Output:

```

Left Diagonal Sum of mat =array([[1, 3, 4, 5, 2],
    [1, 5, 2, 4, 3],
    [5, 2, 3, 4, 1],
    [1, 4, 2, 6, 9],
    [4, 5, 2, 1, 7]]) is 22
Right Diagonal Sum of mat =array([[1, 3, 4, 5, 2],
    [1, 5, 2, 4, 3],
    [5, 2, 3, 4, 1],
    [1, 4, 2, 6, 9],
    [4, 5, 2, 1, 7]]) is 17

```

Problem 4

Question 4: array operations in np.array and np math functions. Plot the function

$$f(x) = \frac{(\sin(x))^7 + (\cos(x))^5}{e^x}$$

Domain: $x \in [0, 4]$

Do not use for loops for the same

Listing 5: Code for problem

```

1 import numpy as np
2 import matplotlib.pyplot as plt
3
4 def f(x: np.ndarray) -> np.ndarray :
5     return ( np.sin(x)*7 + np.cos(x)*5 )/ np.exp(x)
6
7 steps =1000 #note: 4/steps generates 1000 evenly spaced points between 0 and 4)
8 x= np.arange(0 ,4 , 4/steps )
9 y= f(x)
10 plt.plot(x, y, color='black')
11 plt.title('f(x) = (sin^7(x) + cos^5(x))/(e^x)')
12 plt.xlabel('x-values')
13 plt.ylabel('y-values')
14
15 plt.tight_layout() #adjust layout
16

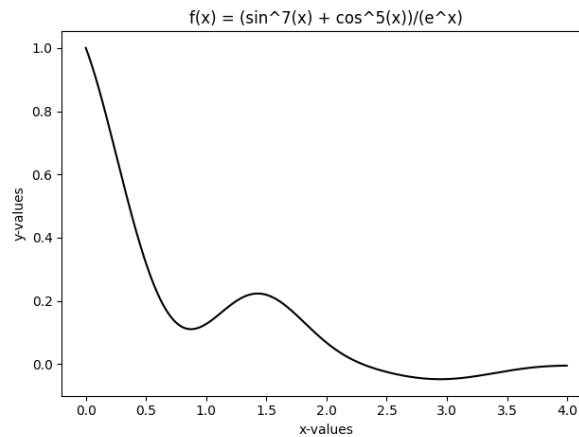
```

```

17 # save/show plot
18 plt.savefig("q4.png")
19 plt.show()

```

Output:



Problem 5

Create a subplot with two plots side by side. Plot a sine wave in the first subplot. Plot a cosine wave in the second subplot. Add labels, titles, and a legend to the plots

Listing 6: Code for problem

```

1 import numpy as np
2 import matplotlib.pyplot as plt
3
4 x = np.linspace(0, 2 * np.pi, 100) #x values for both sin and cos
5 y_sin = np.sin(x) #y values for sine
6 y_cos = np.cos(x) #y values for cosine
7 fig, axs = plt.subplots(1, 2, figsize=(12, 4)) #layout 2 subplots
8
9 # Plot1 sine wave
10 axs[0].plot(x, y_sin, label='Sine Wave', color='black')
11 axs[0].set_title('Sine Wave (y=sin(x))')
12 axs[0].set_xlabel('x-values')
13 axs[0].set_ylabel('y-values')
14 axs[0].legend()
15
16 # Plot2 cosine wave
17 axs[1].plot(x, y_cos, label='Cosine Wave', color='black')
18 axs[1].set_title('Cosine Wave (y=cos(x))')
19 axs[1].set_xlabel('x-values')

```

```
20 axs[1].set_ylabel('y-values')
21 axs[1].legend()
22
23 plt.tight_layout() #adjust layout
24
25 # save/show plot
26 plt.savefig("q5.png")
27 plt.show()
```

Output:

