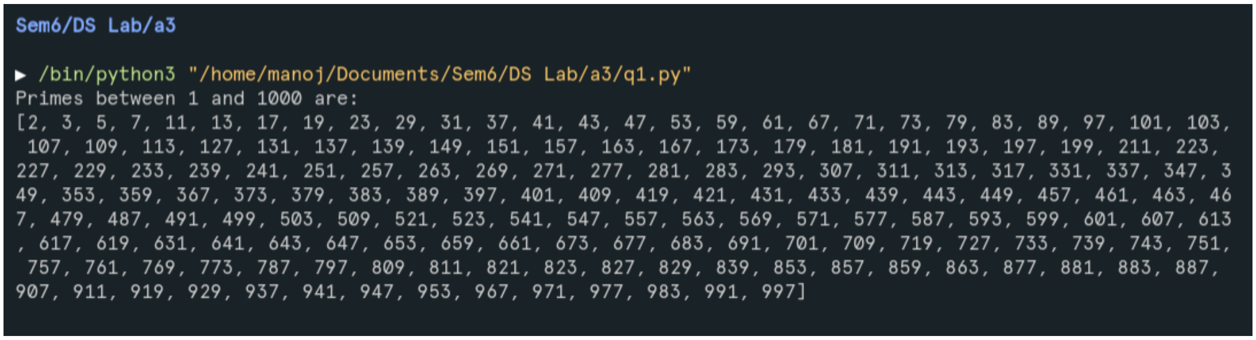
Data Science - Lab 3

☰ Student Name	Venkata Sai Manoj Boganadham
# Roll no	197121
☰ Section	A
▼ Type	Assignment
▼ Subject	DS Lab

1. Write a python program to print all the prime numbers between 1 to 1000 using loop.

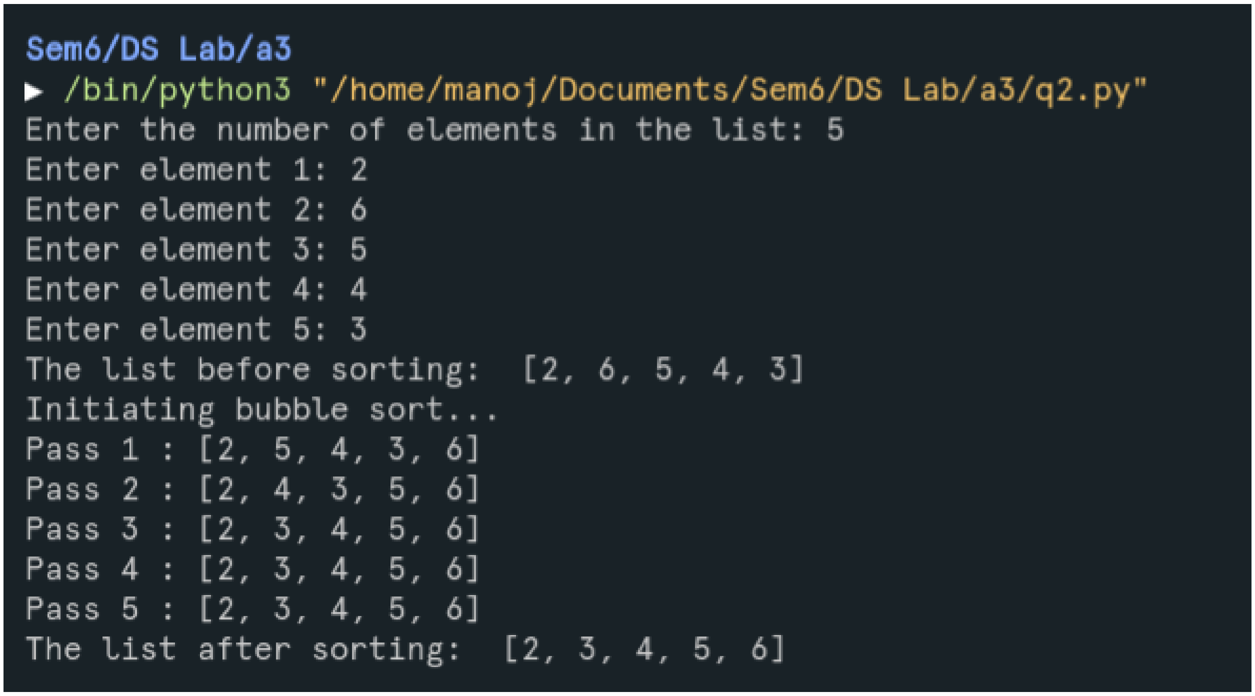
```
primes = [];

# python program to print the prime numbers between 1 to 1000 using for loop
for i in range(2,1000):
    isPrime = 1;
    for j in range(2,i):
        # if a number is divisible by any number between 2 and itself then it is not prime
        if(i%j==0):
            isPrime = 0;
            break
    # if the number is prime then it is appended to the list
    if(isPrime==1):
        primes.append(i);

print("Primes between 1 and 1000 are: ")
print(primes);
```



```
Sem6/DS Lab/a3

▶ /bin/python3 "/home/manoj/Documents/Sem6/DS Lab/a3/q1.py"
Primes between 1 and 1000 are:
[2, 3, 5, 7, 11, 13, 17, 19, 23, 29, 31, 37, 41, 43, 47, 53, 59, 61, 67, 71, 73, 79, 83, 89, 97, 101, 103,
 107, 109, 113, 127, 131, 137, 139, 149, 151, 157, 163, 167, 173, 179, 181, 191, 193, 197, 199, 211, 223,
227, 229, 233, 239, 241, 251, 257, 263, 269, 271, 277, 281, 283, 293, 307, 311, 313, 317, 331, 337, 347, 3
49, 353, 359, 367, 373, 379, 383, 389, 397, 401, 409, 419, 421, 431, 433, 439, 443, 449, 457, 461, 463, 46
7, 479, 487, 491, 499, 503, 509, 521, 523, 541, 547, 557, 563, 569, 571, 577, 587, 593, 599, 601, 607, 613
, 617, 619, 631, 641, 643, 647, 653, 659, 661, 673, 677, 683, 691, 701, 709, 719, 727, 733, 739, 743, 751,
 757, 761, 769, 773, 787, 797, 809, 811, 821, 823, 827, 829, 839, 853, 857, 859, 863, 877, 881, 883, 887,
907, 911, 919, 929, 937, 941, 947, 953, 967, 971, 977, 983, 991, 997]
```

2. Use python programming to implement bubble sort. [define a function to perform the sorting and take the input from the user; for each passes display pass number and the respective sorted array]

```
# function to implement bubble sort on a list
def bubbleSort(list):
    n = len(list);
    for i in range(n):
        # Traverse through all array elements
        for j in range(0,n-i-1):
            # Swap if the element found is greater
            if(list[j]>list[j+1]):
                list[j], list[j+1] = list[j+1], list[j];
        # printing the list at every pass
        print("Pass",i+1,":", list);


# taking input from the user
n = int(input("Enter the number of elements in the list: "));
list = [];
# taking the list from the user
for i in range(n):
    list.append(int(input("Enter element " + str(i+1) + ": ")));
print("The list before sorting: ", list);
print("Initiating bubble sort...");
# calling the function
bubbleSort(list);
print("The list after sorting: ", list);
```

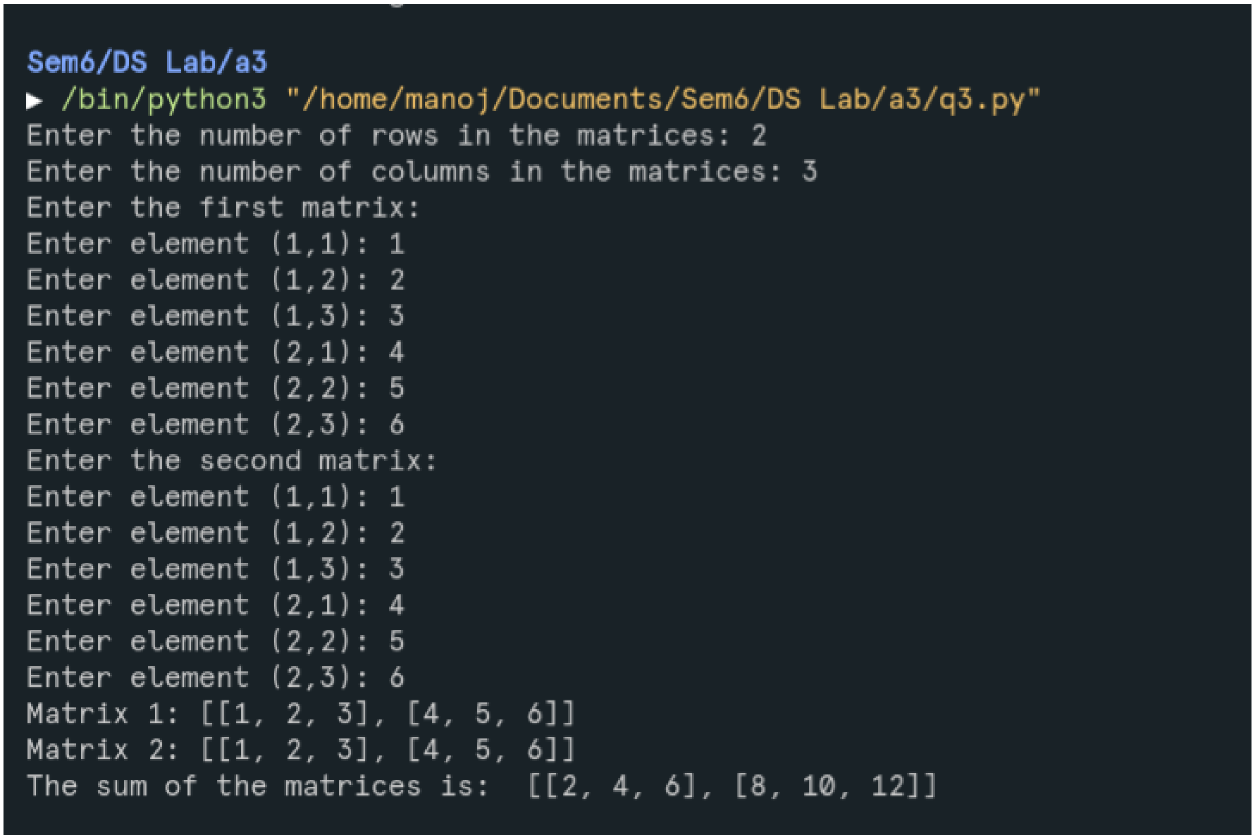


```
Sem6/DS Lab/a3
▶ /bin/python3 "/home/manoj/Documents/Sem6/DS Lab/a3/q2.py"
Enter the number of elements in the list: 5
Enter element 1: 2
Enter element 2: 6
Enter element 3: 5
Enter element 4: 4
Enter element 5: 3
The list before sorting:  [2, 6, 5, 4, 3]
Initiating bubble sort...
Pass 1 : [2, 5, 4, 3, 6]
Pass 2 : [2, 4, 3, 5, 6]
Pass 3 : [2, 3, 4, 5, 6]
Pass 4 : [2, 3, 4, 5, 6]
Pass 5 : [2, 3, 4, 5, 6]
The list after sorting:  [2, 3, 4, 5, 6]
```

3. Write a python program to compute the sum of two matrices and display the result. [take the input from the user]

```
# program to compute the sum of two matrices
mat1 = []
mat2 = []

# get the dimensions of the lists from the user
rows = int(input("Enter the number of rows in the matrices: "))
cols = int(input("Enter the number of columns in the matrices: "))

# get the matrix 1
print("Enter the first matrix: ")
for i in range(rows):
    row = []
    for j in range(cols):
        row.append(int(input("Enter element (" + str(i+1) + "," + str(j+1) + "): ")))
    mat1.append(row)

# get the matrix 2
print("Enter the second matrix: ")
for i in range(rows):
    row = []
    for j in range(cols):
        row.append(int(input("Enter element (" + str(i+1) + "," + str(j+1) + "): ")))
    mat2.append(row)

# add the matrices
```

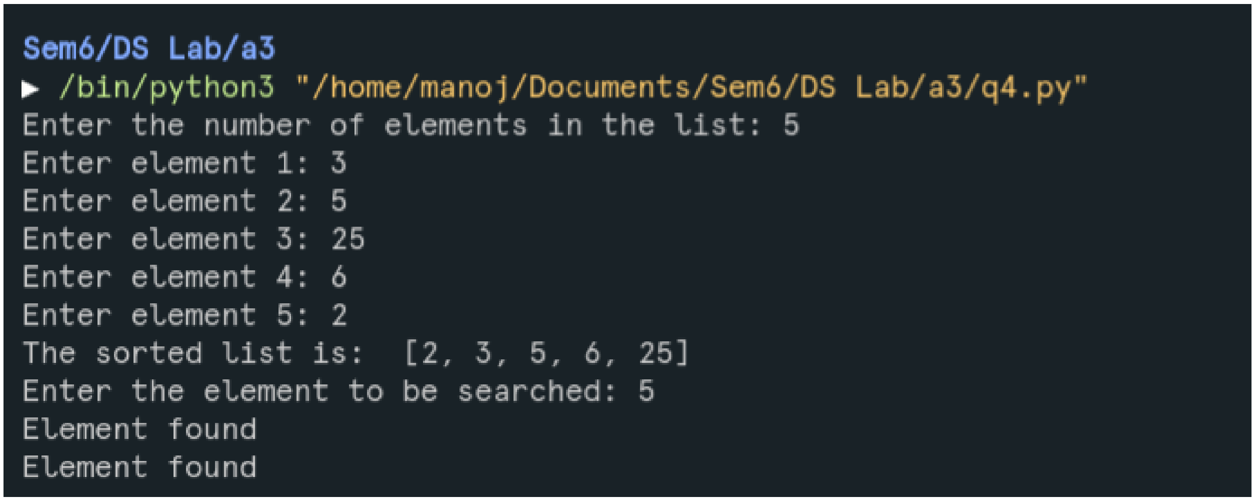
```
mat3 = []
for i in range(rows):
    row = []
    for j in range(cols):
        row.append(mat1[i][j] + mat2[i][j])
    mat3.append(row)

# print the matrices
print("Matrix 1:", mat1)
print("Matrix 2:", mat2)
print("The sum of the matrices is: ", mat3)
```



```
Sem6/DS Lab/a3
▶ /bin/python3 "/home/manoj/Documents/Sem6/DS Lab/a3/q3.py"
Enter the number of rows in the matrices: 2
Enter the number of columns in the matrices: 3
Enter the first matrix:
Enter element (1,1): 1
Enter element (1,2): 2
Enter element (1,3): 3
Enter element (2,1): 4
Enter element (2,2): 5
Enter element (2,3): 6
Enter the second matrix:
Enter element (1,1): 1
Enter element (1,2): 2
Enter element (1,3): 3
Enter element (2,1): 4
Enter element (2,2): 5
Enter element (2,3): 6
Matrix 1: [[1, 2, 3], [4, 5, 6]]
Matrix 2: [[1, 2, 3], [4, 5, 6]]
The sum of the matrices is:  [[2, 4, 6], [8, 10, 12]]
```

4.Use python programming to implement the binary search by using the methods[take the input from the user]:
a. Recursive method
b. Iterative method

```
# function to perform recursive binary search
def binary_search(arr, l, r, x):
    # check if the array is empty
    if r >= l:
        # performing integer division
        mid = l + (r - l) // 2
        # check if the mid element is equal to x
        if arr[mid] == x:
            return mid
        # if the mid element is greater than x, perform search in the left half
        elif arr[mid] > x:
            return binary_search(arr, l, mid - 1, x)
        # if the mid element is less than x, perform search in the right half
        else:
            return binary_search(arr, mid + 1, r, x)
    # if the array is empty, return -1
    return -1

# function to perform binary search
def binary_search_itr(arr, x):
    # initialize left and right indices
    l = 0
    r = len(arr) - 1
    # loop till left index is less than right index
    while l <= r:
        # performing integer division
        mid = l + (r - l) // 2
        # check if the mid element is equal to x
        if arr[mid] == x:
            return mid
        # if the mid element is greater than x, perform search in the left half
        elif arr[mid] > x:
            r = mid - 1
        # if the mid element is less than x, perform search in the right half
        else:
            l = mid + 1
    # if the element is not found, return -1
    return -1

list = []
# length of the list from user
n = int(input("Enter the number of elements in the list: "))
# taking the list from the user
for i in range(n):
    list.append(int(input("Enter element " + str(i+1) + ": ")));
# sort the list
list.sort()
# print the list
print("The sorted list is: ", list);



# search for an element in the list
x = int(input("Enter the element to be searched: "))
# calling the recursive function
result = binary_search(list, 0, len(list) - 1, x)
# check if the element is found
if result != -1:
    print("Element found")
else:
    print("Element not found")


# calling the iterative function
result = binary_search_itr(list, x)
# check if the element is found
if result != -1:
    print("Element found")
else:
    print("Element not found")
```



```
Sem6/DS Lab/a3
▶ /bin/python3 "/home/manoj/Documents/Sem6/DS Lab/a3/q4.py"
Enter the number of elements in the list: 5
Enter element 1: 3
Enter element 2: 5
Enter element 3: 25
Enter element 4: 6
Enter element 5: 2
The sorted list is:  [2, 3, 5, 6, 25]
Enter the element to be searched: 5
Element found
Element found
```

5.Write a python program using NumPy:

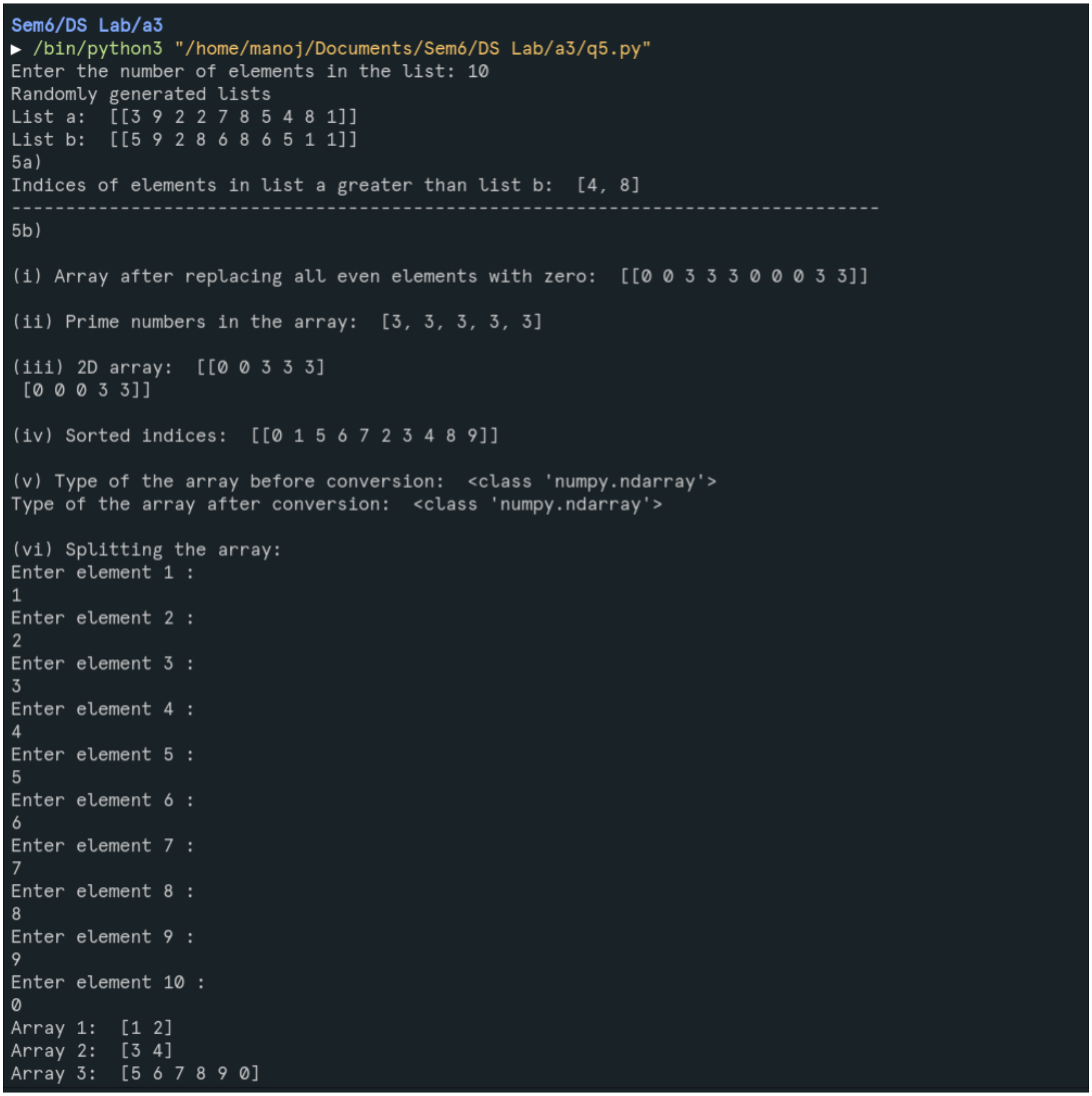
a. Create two 1-D arrays of same size with n number of elements and display the index of the arrays where the value of elements in 1st array is more than and equal to its corresponding element in 2nd array.

b. Create a 1-D array and perform the following:

i. Replace all even numbers in the array with 0
ii. Extract the prime numbers from the array
iii. Convert the 1D array to a 2D array in 2 rows Input
iv. Display the array element indices such that array elements are sorted in ascending order [without the changing the position of elements]
v. Convert a binary NumPy array (holding only 0s and 1s) to a Boolean NumPy array.
vi. Take an input of 10 elements and split the array into 3 arrays, where 1st two arrays should have 2 elements each and the rest of the elements in the last array. Display the arrays.

```
from random import random
import numpy as np

# get number of elements from the user
n = int(input("Enter the number of elements in the list: "))

a = np.random.randint(1,10,size=(1,n))
b = np.random.randint(1,10,size=(1,n))

# print the lists a and b
print("Randomly generated lists")
print("List a: ", a)
print("List b: ", b)

indices = []

for i in range(n):
    if(a[0][i] > b[0][i]):
        indices.append(i);

# print indices
print("5a)")
print("Indices of elements in list a greater than list b: ", indices)
print("--------------------------------------------------------------------------------");

# create a numpy array of random numbers
arr = np.random.randint(1,10,size=(1,n))

print("5b)");
# replace all the even elements with zero
arr[arr%2==0] = 0
# print the array
print("\n(i) Array after replacing all even elements with zero: ", arr)
# extract all the prime numbers from the array
primes = []
for i in arr[0]:
    isPrime = 1
    entered = 0
    if(i>=2):
        entered = 1
        for j in range(2,i):
            if(i%j==0):
                isPrime = 0
                break
    if(isPrime==1 and entered==1):
        # insert i into the set primes
        primes.append(i);

#print the prime numbers
print("\n(ii) Prime numbers in the array: ", primes)

# Convert the 1D array to a 2D array in 2 rows Input
```

```
dim2 = n//2
newarr = arr.reshape(2,dim2);
# print the array
print("\n(iii) 2D array: ", newarr)

# Display the array element indices such that array elements are sorted in ascending order
[ without the changing the position of elements]
sortedIndices = np.argsort(arr)
#print the sorted indices
print("\n(iv) Sorted indices: ", sortedIndices)

# Create a numpy array containing 0s and 1s
arr = np.random.randint(0,2,size=(1,n))
# convert the array to boolean
# type of the array before conversion
print("\n(v) Type of the array before conversion: ", type(arr))
boolarr = np.array(arr, dtype=bool)
# type of the array after conversion
print("Type of the array after conversion: ", type(boolarr))


# create a numpy array of 10 elements from user input
print("\n(vi) Splitting the array: ")
arr = np.random.randint(1,10,size=(1,10))
for i in range(len(arr[0])):
    print("Enter element", i+1, ": ")
    arr[0][i] = int(input())

a1 = arr[0][0:2]
a2 = arr[0][2:4]
a3 = arr[0][4:]

# print the arrays
print("Array 1: ", a1)
print("Array 2: ", a2)
print("Array 3: ", a3)
```



```
Sem6/DS Lab/a3
▶ /bin/python3 "/home/manoj/Documents/Sem6/DS Lab/a3/q5.py"
Enter the number of elements in the list: 10
Randomly generated lists
List a:  [[3 9 2 2 7 8 5 4 8 1]]
List b:  [[5 9 2 8 6 8 6 5 1 1]]
5a)
Indices of elements in list a greater than list b:  [4, 8]
--------------------------------------------------------------------------------
5b)

(i) Array after replacing all even elements with zero:  [[0 0 3 3 3 0 0 0 3 3]]

(ii) Prime numbers in the array:  [3, 3, 3, 3, 3]

(iii) 2D array:  [[0 0 3 3 3]
 [0 0 0 3 3]]

(iv) Sorted indices:  [[0 1 5 6 7 2 3 4 8 9]]

(v) Type of the array before conversion:  <class 'numpy.ndarray'>
Type of the array after conversion:  <class 'numpy.ndarray'>

(vi) Splitting the array:
Enter element 1 :
1
Enter element 2 :
2
Enter element 3 :
3
Enter element 4 :
4
Enter element 5 :
5
Enter element 6 :
6
Enter element 7 :
7
Enter element 8 :
8
Enter element 9 :
9
Enter element 10 :
0
Array 1:  [1 2]
Array 2:  [3 4]
Array 3:  [5 6 7 8 9 0]
```

6. There are 190 students in a class of Data Science Theory. The subject is taught every day (Monday to Sunday) in a week for an hour. Create and display a series of data as a count of attendance of the total number of students attending the subject every day in a week. [Hint: Use pandas to create the dataset, create the dataset for a week i.e. for all 7 days in a week, for each respective day mention the number of attendees.] Perform the following with the series dataset created.

a. Display the dataset

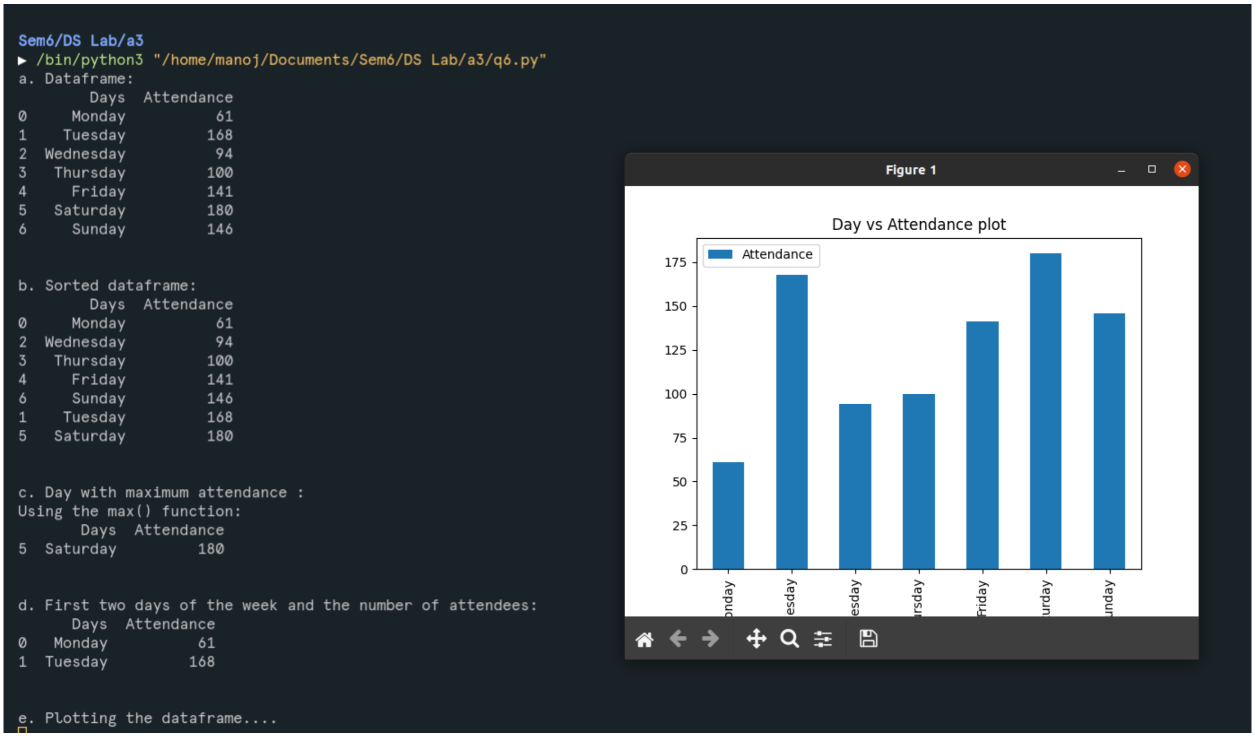
b. Display the sorted dataset with least number of attendees at first

c. Show the day with maximum number of attendees

d. Display the 1st two days of the week and the number of attendees

e. Plot the dataset for each day in the week.

```
import numpy as np
import matplotlib.pyplot as plt
import pandas as pd

# create a dataframe with the days in a week and their corresponding attendance count
attendance = np.random.randint(60, 190, size=(1,7))
attendanceList = attendance[0].tolist()
df = pd.DataFrame({'Days':['Monday','Tuesday','Wednesday','Thursday','Friday','Saturda
y','Sunday'],
                     'Attendance':attendanceList})
# print the dataframe
print("a. Dataframe: ")
print(df)
# sorting the dataframe according to the attendance count
sortedDf = df.sort_values(by=['Attendance'], ascending=True)
print("\n\nb. Sorted dataframe: ")
print(sortedDf)

# day with maximum attendance
print("\n\nc. Day with maximum attendance :")
print("Using the max() function: ")
print(df[df.Attendance == df.Attendance.max()])

# first two days of the week and the number of attendees
print("\n\nd. First two days of the week and the number of attendees: ")
print(df.head(2))

# plot the dataframe
print("\n\ne. Plotting the dataframe....")
df.plot(kind='bar', x='Days', y='Attendance')
plt.title('Day vs Attendance plot')
plt.show()
```



```
Sem6/DS Lab/a3
▶ /bin/python3 "/home/manoj/Documents/Sem6/DS Lab/a3/q6.py"
a. Dataframe:
        Days  Attendance
0     Monday          61
1    Tuesday         168
2  Wednesday          94
3   Thursday         100
4     Friday         141
5   Saturday         180
6     Sunday         146


b. Sorted dataframe:
        Days  Attendance
0     Monday          61
2  Wednesday          94
3   Thursday         100
4     Friday         141
6     Sunday         146
1    Tuesday         168
5   Saturday         180


c. Day with maximum attendance :
Using the max() function:
       Days  Attendance
5  Saturday         180


d. First two days of the week and the number of attendees:
      Days  Attendance
0   Monday          61
1  Tuesday         168


e. Plotting the dataframe....
```

Consider the data set in Kaggle and perform the following:

a. Read the dataset

b. Display the information related to the dataset such as the number of rows and columns

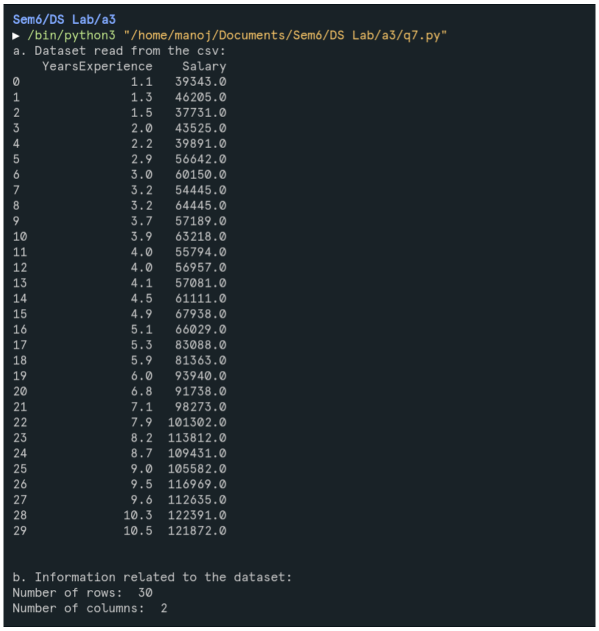
c. Display the first 5 rows

d. Display the summary statistics for each numeric column

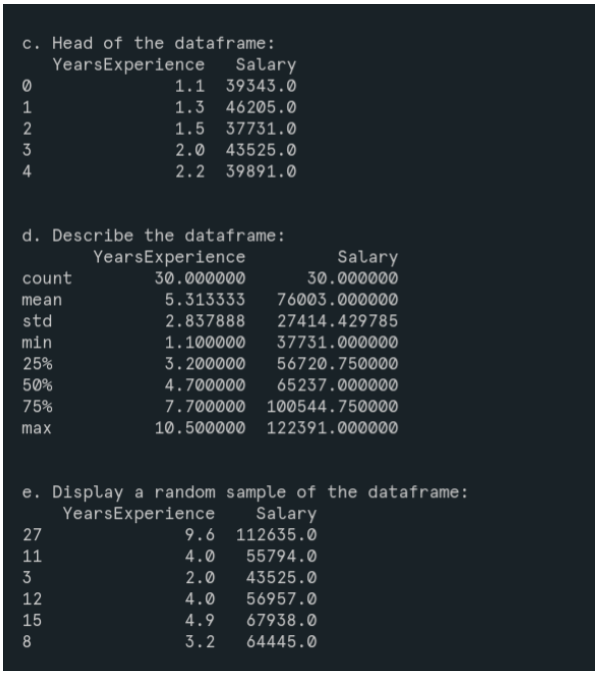
e. Display a random subset (at least 5)

```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt

# reading the dataset from the csv file
df = pd.read_csv("./Salary_Data.csv")
print("a. Dataset read from the csv: ")
print(df);

# Display the information related to the dataset such as the number of rows and columns
print("\n\nb. Information related to the dataset: ")
print("Number of rows: ", df.shape[0])
print("Number of columns: ", df.shape[1])

# display the head of dataframe
print("\n\nc. Head of the dataframe: ")
print(df.head())
```

```
# describe the dataframe
print("\n\nd. Describe the dataframe: ")
print(df.describe())

# Display a random sample of the dataframe
print("\n\ne. Display a random sample of the dataframe: ")
print(df.sample(6))
```



```
Sem6/DS Lab/a3
▶ /bin/python3 "/home/manoj/Documents/Sem6/DS Lab/a3/q7.py"
a. Dataset read from the csv:
    YearsExperience    Salary
0               1.1   39343.0
1               1.3   46205.0
2               1.5   37731.0
3               2.0   43525.0
4               2.2   39891.0
5               2.9   56642.0
6               3.0   60150.0
7               3.2   54445.0
8               3.2   64445.0
9               3.7   57189.0
10              3.9   63218.0
11              4.0   55794.0
12              4.0   56957.0
13              4.1   57081.0
14              4.5   61111.0
15              4.9   67938.0
16              5.1   66029.0
17              5.3   83088.0
18              5.9   81363.0
19              6.0   93940.0
20              6.8   91738.0
21              7.1   98273.0
22              7.9  101302.0
23              8.2  113812.0
24              8.7  109431.0
25              9.0  105582.0
26              9.5  116969.0
27              9.6  112635.0
28             10.3  122391.0
29             10.5  121872.0


b. Information related to the dataset:
Number of rows:  30
Number of columns:  2
```



```
c. Head of the dataframe:
   YearsExperience    Salary
0              1.1   39343.0
1              1.3   46205.0
2              1.5   37731.0
3              2.0   43525.0
4              2.2   39891.0


d. Describe the dataframe:
       YearsExperience         Salary
count        30.000000      30.000000
mean          5.313333   76003.000000
std           2.837888   27414.429785
min           1.100000   37731.000000
25%           3.200000   56720.750000
50%           4.700000   65237.000000
75%           7.700000  100544.750000
max          10.500000  122391.000000


e. Display a random sample of the dataframe:
    YearsExperience    Salary
27              9.6  112635.0
11              4.0   55794.0
3               2.0   43525.0
12              4.0   56957.0
15              4.9   67938.0
8               3.2   64445.0
```