P Manohar Rao

Roll no:197158

Data Science Lab Assignment-3

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

```
import math
print(2,end=" ")
for num in range(3,1000):
··c=0
··for·i·in·range(2,int(math.sqrt(num))+1):
    if(num%i)==0:
      c=c+1
  if c==0:
    print(num,end=" ")
```

```
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 1
```

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]

```
def bsort(arr):
  n=len(arr)
  c=1
  for i in range(n-1):
    for j in range(n-i-1):
      if arr[j]>arr[j+1]:
        arr[j],arr[j+1]=arr[j+1],arr[j]
        c+=1
        print(c,"-",">",end=" ")
        print(*arr)
a=[]
n=int(input("no of no's"))
for i in range(n):
  a.append(int(input()))
print(a)
bsort(a)
print(a)
```

```
no of no's5
5
4
3
```

```
2
1
[5, 4, 3, 2, 1]
2 - > 4 5 3 2 1
3 - > 4 3 5 2 1
4 - > 4 3 2 5 1
5 - > 4 3 2 1 5
6 - > 3 4 2 1 5
7 - > 3 2 4 1 5
8 - > 3 2 1 4 5
9 - > 2 3 1 4 5
10 - > 2 1 3 4 5
11 - > 1 2 3 4 5
[1, 2, 3, 4, 5]
```

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

```
m1=[]
m2=[]
r=int(input("enter no of rows"))
c=int(input("enter no of cols"))

for i in range(r):
  t=[]
  for k in range(c):
    n=int(input("enter elements of matrix 1 :"))
    t.append(n)
  m1.append(t)

for i in range(r):
  t=[]
  for k in range(c):
    n=int(input("enter elements of matrix 2 :"))
    t.append(n)
  m2.append(t)
res=[]
for i in range(r):
  l=[]
  for k in range(c):
    n=m1[i][k]+m2[i][k]
    l.append(n)
  res.append(l)
```

```
enter no of rows5
enter no of cols6
enter elements of matrix 1 :4
enter elements of matrix 1 :6
enter elements of matrix 1 :8
enter elements of matrix 1 :1
enter elements of matrix 1 :0
enter elements of matrix 1 :6
enter elements of matrix 1 :8
```

```
enter elements of matrix 1 :1
enter elements of matrix 1 :3
enter elements of matrix 1 :5
enter elements of matrix 1 :3
enter elements of matrix 1 :8
enter elements of matrix 1 :7
enter elements of matrix 1 :9
enter elements of matrix 1 :2
enter elements of matrix 1 :6
enter elements of matrix 1 :5
enter elements of matrix 1 :2
enter elements of matrix 1 :3
enter elements of matrix 1 :2
enter elements of matrix 1 :3
enter elements of matrix 1 :4
enter elements of matrix 1 :6
enter elements of matrix 1 :5
enter elements of matrix 1 :5
enter elements of matrix 1 :6
enter elements of matrix 1 :8
enter elements of matrix 1 :9
enter elements of matrix 1 :6
enter elements of matrix 1 :5
enter elements of matrix 2 :6
enter elements of matrix 2 :2
enter elements of matrix 2 :3
enter elements of matrix 2 :5
enter elements of matrix 2 :8
enter elements of matrix 2 :6
enter elements of matrix 2 :9
enter elements of matrix 2 :4
enter elements of matrix 2 :5
enter elements of matrix 2 :6
enter elements of matrix 2 :2
enter elements of matrix 2 :6
enter elements of matrix 2 :8
enter elements of matrix 2 :6
enter elements of matrix 2 :5
enter elements of matrix 2 :1
enter elements of matrix 2 :2
enter elements of matrix 2 :3
enter elements of matrix 2 :4
enter elements of matrix 2 :2
enter elements of matrix 2 :5
enter elements of matrix 2 :0
enter elements of matrix 2 :0
enter elements of matrix 2 :6
enter elements of matrix 2 :5
enter elements of matrix 2 :8
```

```
m1
```

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

```
m2
```

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

```
res
```

```
[[10, 8, 11, 6, 8, 12],
 [17, 5, 8, 11, 5, 14],
 [15, 15, 7, 7, 7, 5],
 [7, 4, 8, 4, 6, 11],
 [10, 14, 14, 11, 7, 8]]
```

4) Use python programming to implement the binary search by using the methods[take the input from the user]:

a) Iterative method

```
def binSearch(arr,x):
  n=len(arr)
  l,h=0,n-1

  while(l<=h):
    mid=(l+h)//2
    if x==arr[mid]:
      return mid
    elif x<arr[mid]:
      h=mid-1
    else:
      l=mid+1
  return -1
a=[]
n=int(input("enter no of elements in an array "))
t=int(input("enter the target to be searched "))
for i in range(n):
  a.append(int(input()))
a.sort()
res=binSearch(a,t)
if res!=-1:
  print("element found at",res)
else:
  print("element not found!!")
```

```
enter no of elements in an array 9
enter the target to be searched 6
5
3
2
6
8
3
```

```
7
1
0
element found at 6
```

a) Recursive method

```
def binSearch(arr,x,l,h):
    mid=(l+h)//2
    if l>h:
      return -1
    elif x==arr[mid]:
      return mid
    elif x<arr[mid]:
      return binSearch(arr,x,l,mid-1)
    else:
      return binSearch(arr,x,mid+1,h)

a=[]
n=int(input("enter no of elements in an array "))
t=int(input("enter the target to be searched "))
for i in range(n):
  a.append(int(input()))
a.sort()
res=binSearch(a,t,0,len(a)-1)
if res!=-1:
  print("element found at",res)
else:
  print("element not found!!")
```

```
enter no of elements in an array 9
enter the target to be searched 5
3
6
8
11
6
2
5
10
6
element found at 2
```

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:

```
import numpy as np
a=np.array([6,7,9,10])
b=np.array([5,6,8,11])
x=np.where(a>=b)
x
```

```
(array([0, 1, 2]),)
```

i) Replace all even numbers in the array with 0

```
a=np.array([2,5,6,7,8,9,1,4,11,13])
y=np.where((a%2)==0,0,a)
y
```

```
array([ 0,  5,  0,  7,  0,  9,  1,  0, 11, 13])
```

ii) Extract the prime numbers from the array

```
import math
def isprime(n):
  if n<=1:
    return 0
  if n==2:
    return 1
  if n%2==0 and n>2:
    return 0
  div=int(math.sqrt(n))
  for i in range(3,div+1,2):
    if n%i==0:
      return 0
  return 1

b=[]
for ele in a:
  if isprime(ele)==1:
    b.append(ele)
print(b)
```

```
[2, 5, 7, 11, 13]
```

iii) Convert the 1D array to a 2D array in 2 rows Input

```
a.reshape(2,-1)
```

```
array([[ 2,  5,  6,  7,  8],
       [ 9,  1,  4, 11, 13]])
```

iv) Display the array element indices such that array elements are sorted in ascending order [without the changing the position of elements]

```
print(np.argsort(a))
print(a)
```

```
[6 0 7 1 2 3 4 5 8 9]
[ 2  5  6  7  8  9  1  4 11 13]
```

v) Convert a binary NumPy array (holding only 0s and 1s) to a Boolean NumPy array.

```
barray=np.array([0,1,1,0,1,0,1,0,0])
barray=map(lambda x:True if x==1 else False,barray)
print(*barray)
```

```
False True True False True False True False False
```

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.

```
a1=[]
for i in range(10):
  a1.append(int(input()))
a1=np.array(a1)
a11=a1[0:2]
a12=a1[2:4]
a13=a1[4:]
print("main array",a1)
print("array 1",a11)
print("array 2",a12)
print("array 3",a13)
```

```
25
65
20
74
82
19
27
31
18
82
main array [25 65 20 74 82 19 27 31 18 82]
array 1 [25 65]
array 2 [20 74]
array 3 [82 19 27 31 18 82]
```

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.

```
import pandas as pd
data={
    "day":["Mon","Tues","Wed","Thurs","Fri","Sat","Sun"],
    "attendance":[80,74,97,54,34,30,52]

}
dataset=pd.DataFrame(data)
```

a) Display the dataset

```
print(dataset)
```

```
     day  attendance
0    Mon          80
1   Tues          74
2    Wed          97
3  Thurs          54
4    Fri          34
5    Sat          30
6    Sun          52
```

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

```
print(dataset.sort_values(by='attendance'))
```

```
     day  attendance
5    Sat          30
4    Fri          34
6    Sun          52
3  Thurs          54
1   Tues          74
0    Mon          80
2    Wed          97
```

c) Show the day with maximum number of attendees

```
print("Maximum number of students came on :\n",dataset[dataset.attendance==dataset.attenda
```

```
Maximum number of students came on :
    day  attendance
2  Wed          97
```

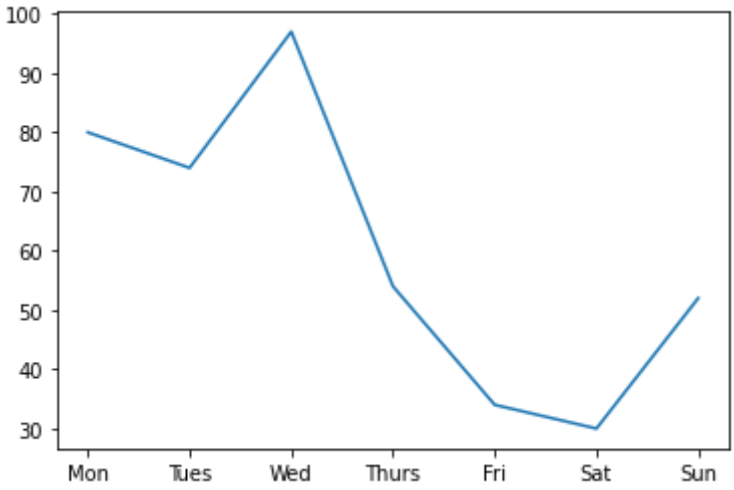
d) Display the 1st two days of the week and the number of attendees

```
print("1st two days of the week attendance are :\n",dataset.head(2))
```

```
1st two days of the week attendance are :
     day  attendance
0   Mon          80
1  Tues          74
```

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

```
import matplotlib.pyplot as plt
plt.plot(dataset.day,dataset.attendance)
plt.show()
```



7. Consider the data set: https://www.kaggle.com/karthickveerakumar/salary-data-simple-linear-regression and perform the following:

a) Read the dataset

```
from google.colab import drive
drive.mount('/content/drive')
```

```
Mounted at /content/drive
```

```
import pandas as pd
path="/content/drive/MyDrive/Salary_Data.csv"
df=pd.read_csv(path)
```

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

```
print(df.info())
```

```
<class 'pandas.core.frame.DataFrame'>
```

```
RangeIndex: 30 entries, 0 to 29
Data columns (total 2 columns):
 #   Column           Non-Null Count  Dtype  
---  ------           --------------  -----  
 0   YearsExperience  30 non-null     float64
 1   Salary           30 non-null     float64
dtypes: float64(2)
memory usage: 608.0 bytes
None
```

c) Display the first 5 rows

```
print(df.head(5))
```

```
   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) Display the summary statistics for each numeric column

```
print(df.describe())
```

```
       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 subset (at least 5)

```
print(df.sample(n=5))
```

```
    YearsExperience    Salary
24              8.7  109431.0
29             10.5  121872.0
13              4.1   57081.0
3               2.0   43525.0
0               1.1   39343.0
```

✓ 0s completed at 1:47 PM