DS Lab Assignment 3

Kadali Sai Vivek

197139

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

In [5]:
```
primes=[] #storing all primes in a list

for i in range(1,1001): #iterate over the numbers between 1 to 1000
  if i>1:#As 1 is not a Prime
    f=1 #flag to check if loop break in middle
    for j in range(2,i): #loop to check each number whether it has any divisor between
      if i%j==0 : # if so ,break in middle
        f=0
        break
    if f==1:#if no Divisor then it is a Prime number
      primes.append(i)

print(primes)
```

```
[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, 27
7, 281, 283, 293, 307, 311, 313, 317, 331, 337, 347, 349, 353, 359, 367, 373, 379, 383,
389, 397, 401, 409, 419, 421, 431, 433, 439, 443, 449, 457, 461, 463, 467, 479, 487, 49
1, 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, 72
7, 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, 97
1, 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]

In [ ]:
```
def bubblesort(arr): #Function to do bubble sort
  n=len(arr) #calculating length of array
  passno=1   #initializing pass number with 1

  for i in range(n-1): #iterate each number in array
    print("Pass number :",passno)
    for j in range(i+1,n):
      if arr[i] > arr[j]: #compair if element i is greater than element j
        temp=arr[i]
        arr[i]=arr[j]
        arr[j]=temp
```

```
    print(arr)
    passno=passno+1
  return arr



arr=[]
n=int(input("Enter number of elements in the Array :"))
for i in range(n):
  k=int(input())
  arr.append(k)

print("Initial Array :",arr)
bubblesort(arr)
print("Final sorted Array :",arr)
```

```
Enter number of elements in the Array :5
6
7
9
1
0
Initial Array : [6, 7, 9, 1, 0]
Pass number  1
[0, 7, 9, 6, 1]
Pass number  2
[0, 1, 9, 7, 6]
Pass number  3
[0, 1, 6, 9, 7]
Pass number  4
[0, 1, 6, 7, 9]
Final sorted Array : [0, 1, 6, 7, 9]
```

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

In [12]:

```
def createMatrix():
  n=int(input("Enter number of Rows:"))
  m=int(input("Enter number of Columns:"))
  Matrix=[]
  print("Enter Elements of matrix row wise :")
  for i in range(n):
    tmp=[] #creating a temporary row
    for j in range(m):
      k=int(input())
      tmp.append(k)
    Matrix.append(tmp) #Adding the row to matrix

  return Matrix


matrix1=createMatrix() #Creating Matrix 1 from user
matrix2=createMatrix() #Creating Matrix 2 from user

print("Matrix 1 :",matrix1)
print("Matrix 2 :",matrix2)
```

```
a=len(matrix1)
b=len(matrix1[0])
c=len(matrix2)
d=len(matrix2[0])

if a!=c and b!=d :
  print("Matrices are not of same size")
else :
  for i in range(a):
    for j in range(b):
      matrix1[i][j]=matrix1[i][j]+matrix2[i][j]


  print("Addition of 2 matrix is :")
  print(matrix1)
```

```
Enter number of Rows:2
Enter number of Columns:3
Enter Elements of matrix row wise :
5
5
6
4
1
2
Enter number of Rows:2
Enter number of Columns:3
Enter Elements of matrix row wise :
7
0
4
5
6
1
Matrix 1 : [[5, 5, 6], [4, 1, 2]]
Matrix 2 : [[7, 0, 4], [5, 6, 1]]
Addition of 2 matrix is :
[[12, 5, 10], [9, 7, 3]]
```

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

a. Recursive method

In [ ]:

```
def binarysearch(arr,x): #function for binary serach
  l=0
  r=len(arr)-1
  while(l<=r): #loop till the left pointer crosses the right pointer
    m=int(l+(r-l)/2) #find middle element
    if(arr[m]==x):
      print("Present at index :",m) #if middle element is the one print it and end the
      return
    elif(arr[m]>x): #if middle element is greater serach left side
      r=m-1
    else: #if middle element is less serach right side
      l=m+1
  print(-1)
```

```
n=int(input("Enter length of the array :"))
arr=[]
for i in range(n):
  k=int(input())
  arr.append(k)
x=int(input("Enter the value to be searched :")) #value to be searched

binarysearch(arr,x) #Apply Binary Serach
```

```
Enter length of the array :6
1
2
3
4
5
6
Enter the value to be searched :2
Present at index : 1
```

b. Iterative method

In [3]:

```
def binarysearch(l,r,arr,x):# Iterative function for binary serach
  if(l>r): #Termination condition
    print(-1)
    return
  m=int(l+(r-l)/2)#find middle element
  if(arr[m]==x):
    print("Present at index :",m)#if middle element is the one print it and end the fun
    return
  elif(arr[m]>x):#if middle element is greater serach left side
    binarysearch(l,m-1,arr,x)
  else:#if middle element is less serach right side
    binarysearch(m+1,r,arr,x)

n=int(input("Enter length of the array :"))
arr=[]
for i in range(n):
  k=int(input())
  arr.append(k)
arr.sort()
x=int(input("Enter the value to be searched :"))#value to be searched

binarysearch(0,len(arr)-1,arr,x)#Apply Binary Serach
```

```
Enter length of the array :5
1
2
3
4
9
Enter the value to be searched :4
Present at index : 3
```

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.

In [4]:
```
import numpy as np


#initializing two 1-D arrays
arr1=np.array([2,3,7,9,1,0,4,1,3,7,2])
arr2=np.array([4,3,2,1,8,3,1,5,0,6,8])

n=len(arr1)

for i in range(n):
    if arr1[i]>=arr2[i] : # compairing 2 arrays to find whether 1st array element is la
        print(i) #print the correspoding index
```

```
1
2
3
6
8
9
```

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

- ## i. Replace all even numbers in the array with 0

In [3]:
```
import numpy as np

arr=np.array([5,4,3,1,6,8,2,7,9,12]) #array creastion

n=len(arr) #finding length of the array
for i in range(n):
    if arr[i]%2==0 :# if it is a divisor of 2 replace with 0
        arr[i]=0

print(arr)
```

```
[5 0 3 1 0 0 0 7 9 0]
```

- ## ii. Extract the prime numbers from the array

In [9]:
```
def isprime(n): #function to check if a number is a prime or not
    if n==1:
        return False
    for i in range(2,n-1): #loop to check each number whether it has any divisor betwee
      if n%i==0 : # if so ,break in middle
        return False
    return True

arr=np.array([5,4,3,1,6,8,2,7,9,12])
```

```
for i in arr: #iterate over the array to find prime
    if isprime(i)==True:
        print(i)
```

```
5
3
2
7
```

- ## iii. Convert the 1D array to a 2D array in 2 rows Input

In [11]:
```
arr=np.array([5,4,3,1,6,8,2,7,9,12])

new_2darr=np.reshape(arr,(2,5)) #reshaping the array into a 2D array of 2 rows 5 column
print(new_2darr)
```

```
[[ 5  4  3  1  6]
 [ 8  2  7  9 12]]
```

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

In [4]:
```
arr1=np.array([6,7,2,1,4,0,9,5,3,8])# initilizing an array

arr2=np.argsort(arr1) # Returns the indices that would sort an array.

print("Original Array :",arr)
print("Indexes of the sorted Array :",arr2)

#if we place the indexes of arr2 from arr1 and display we get sorted array
sortedArray=[]
for i in arr2:
    sortedArray.append(arr[i])
print("Sorted array if we want :",sortedArray)
```

```
Original Array : [5 0 3 1 0 0 0 7 9 0]
Indexes of the sorted Array : [5 3 2 8 4 7 0 1 9 6]
Sorted array if we want : [0, 1, 3, 9, 0, 7, 5, 0, 0, 0]
```

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

In [30]:
```
binary_array=np.array([0,1,1,0,0,0,1,0,1,1,1,0])#creating an Binary array

print("Binary Array  :",binary_array)

boolean_array=np.array(binary_array,dtype=bool) #converting binary array into boolean a

print("Boolean Array :",boolean_array)
```

```
Binary Array  : [0 1 1 0 0 0 1 0 1 1 1 0]
Boolean Array : [False  True  True False False False  True False  True  True  True Fals
e]
```

- ## 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.

In [40]:
```
List=[]
print("Enter 10 elements:")
for i in range(10):
    List.append(int(input())) #take input from the user

array=np.array(List)

print("Complete Array :",array)

arr1=array[:2] #diving the first 2 elements
arr2=array[2:4] #divinding the next 2 elements
arr3=array[4:] #remaining last elements

print("1st Part :",arr1)
print("2nd Part :",arr2)
print("3rd Part :",arr3)
```

```
Enter 10 elements:
5
2
0
8
7
4
1
9
6
3
Complete Array : [5 2 0 8 7 4 1 9 6 3]
1st Part : [5 2]
2nd Part : [0 8]
3rd Part : [7 4 1 9 6 3]
```

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.

In [6]:
```
import pandas as pa

#Data of number of students present on the corresponding day of that week
Data = {
    "Day" : ["Monday","Tuesday","Wednesday","Thursday","Friday","Saturday","Sunday"],
    "Number_of_Attendees":[80,101,75,87,65,55,91]
}
```

- ## a. Display the dataset

In [65]:
```
dataset=pa.DataFrame(Data) #converting the dataframe into dataset

dataset
```

Out[65]:

	Day	Number_of_Attendees
0	Monday	80
1	Tuesday	101
2	Wednesday	75
3	Thursday	87
4	Friday	65
5	Saturday	55
6	Sunday	91

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

In [12]:
```
# Displaying the sorted data based on the "Number_of_Attendees" Column

dataset.sort_values("Number_of_Attendees")
```

Out[12]:

	Day	Number_of_Attendees
5	Saturday	55
4	Friday	65
2	Wednesday	75
0	Monday	80
3	Thursday	87
6	Sunday	91
1	Tuesday	101

- ## c. Show the day with maximum number of attendees

In [30]:
```
column=dataset["Number_of_Attendees"] #storing the Attendence data in a coloumn

max_index=column.idxmax() #finding the maximum index in the column

day_with_max_attendenc=dataset["Day"][max_index] #finding the corresponding Day with ma

print(day_with_max_attendenc) #printing the final result

# (OR)
# Everything in one step
print(dataset["Day"][dataset["Number_of_Attendees"].idxmax()])
```

```
Tuesday
Tuesday
```

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

In [32]:
```
#displaying only the 1st two days using head() command
dataset.head(2)
```

Out[32]:

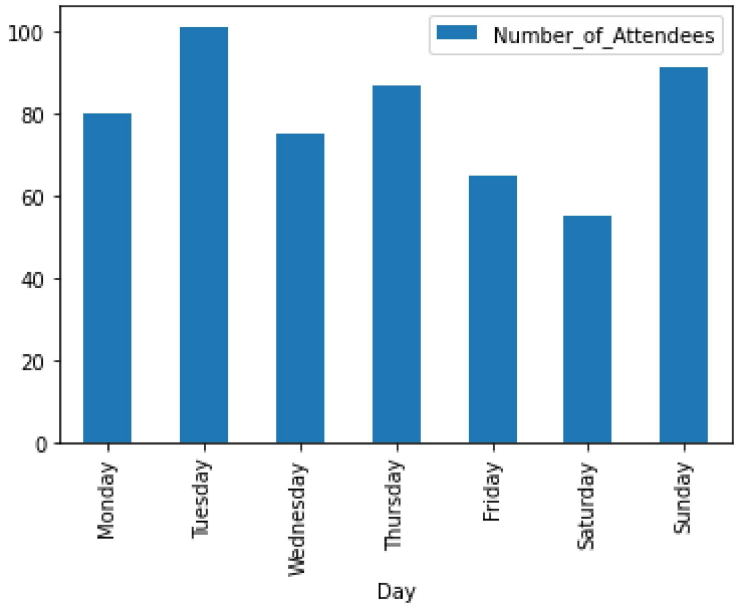
	Day	Number_of_Attendees
0	Monday	80
1	Tuesday	101

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

In [58]:
```
# using Plot method with kind as "bar" to display the dataset

dataset.plot(x='Day', y='Number_of_Attendees',kind="bar")
```

Out[58]:
```
<AxesSubplot:xlabel='Day'>
```



7. Consider the given data set and perform the following:

- ## a. Read the dataset

In [5]:
```
import pandas as pa

dataset=pa.read_csv("./Salary_Data.csv") # reading dataset from the Salary_data using r

dataset # printing whole dataset
```

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

In [79]:

```
rows = len(dataset.axes[0]) # calculating number of rows in dataset
```

```
cols = len(dataset.axes[1]) # calculating number of columns in dataset

print("Total number of rows :",rows)
print("Total number of columns :",cols)
```

```
Total number of rows : 30
Total number of columns : 2
```

- ## c. Display the first 5 rows

In [72]:
```
#displaying the first 5 rows using head method

dataset.head(5)
```

Out[72]:

	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

In [81]:
```
#displaying the summary statistics for each numeric column ,which includes mean,min ele
dataset.describe()
```

Out[81]:

	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)

In [6]:
```
#Printing a random subset form the dataset of 7 elements
dataset.sample(7)
```

Out[6]:

	YearsExperience	Salary

	YearsExperience	Salary
8	3.2	64445.0
26	9.5	116969.0
22	7.9	101302.0
29	10.5	121872.0
28	10.3	122391.0
23	8.2	113812.0
24	8.7	109431.0