```
#@title
# Write a python program to print all the prime numbers between 1 to 1000 using loop
for i in range(1, 1001):
    print(i)


''' Use python programming to implement bubble sort. [define a function to perform the sor
take the input from the user; for each passes display pass number and the respective sorte
'''
nums = list(map(int, input("Enter numbers separated by space: ").split()))
passNum = 1
for i in range(len(nums)):
    for j in range(len(nums)):
        if nums[i] < nums[j]:
            nums[i], nums[j] = nums[j], nums[i]
            print(f"Pass number: {passNum}")
            passNum+=1
            print(f"list: {nums}")
print("\nFinal sorted list\n")
print(nums)
```

```
 Enter numbers separated by space: 23 25 26 2 11 796 12 1
 Pass number: 1
 list: [25, 23, 26, 2, 11, 796, 12, 1]
 Pass number: 2
 list: [26, 23, 25, 2, 11, 796, 12, 1]
 Pass number: 3
 list: [796, 23, 25, 2, 11, 26, 12, 1]
 Pass number: 4
 list: [23, 796, 25, 2, 11, 26, 12, 1]
 Pass number: 5
 list: [23, 25, 796, 2, 11, 26, 12, 1]
 Pass number: 6
 list: [2, 25, 796, 23, 11, 26, 12, 1]
 Pass number: 7
 list: [2, 23, 796, 25, 11, 26, 12, 1]
 Pass number: 8
 list: [2, 23, 25, 796, 11, 26, 12, 1]
 Pass number: 9
 list: [2, 11, 25, 796, 23, 26, 12, 1]
 Pass number: 10
 list: [2, 11, 23, 796, 25, 26, 12, 1]
 Pass number: 11
 list: [2, 11, 23, 25, 796, 26, 12, 1]
 Pass number: 12
 list: [2, 11, 23, 25, 26, 796, 12, 1]
 Pass number: 13
 list: [2, 11, 12, 25, 26, 796, 23, 1]
 Pass number: 14
 list: [2, 11, 12, 23, 26, 796, 25, 1]
 Pass number: 15
 list: [2, 11, 12, 23, 25, 796, 26, 1]
 Pass number: 16
 list: [2, 11, 12, 23, 25, 26, 796, 1]
 Pass number: 17
 list: [1, 11, 12, 23, 25, 26, 796, 2]
 Pass number: 18
```

```
list: [1, 2, 12, 23, 25, 26, 796, 11]
Pass number: 19
list: [1, 2, 11, 23, 25, 26, 796, 12]
Pass number: 20
list: [1, 2, 11, 12, 25, 26, 796, 23]
Pass number: 21
list: [1, 2, 11, 12, 23, 26, 796, 25]
Pass number: 22
list: [1, 2, 11, 12, 23, 25, 796, 26]
Pass number: 23
list: [1, 2, 11, 12, 23, 25, 26, 796]

Final sorted list

[1, 2, 11, 12, 23, 25, 26, 796]
```

```
# Write a python program to compute the sum of two matrices and display the result. [take
mat1S = list(map(int, input("Matrix 1 size: ").split()))
mat2S = list(map(int, input("Matrix 2 size: ").split()))

mat1, mat2 = [], []
print("Enter mat1 elements")
for _ in range(mat1S[0]):
    a = []
    for __ in range(mat1S[1]):
        a.append(int(input()))
    mat1.append(a)
print("Enter mat2 elements")
for _ in range(mat2S[0]):
    a = []
    for __ in range(mat2S[1]):
        a.append(int(input()))
    mat2.append(a)

if mat1S[0] != mat2S[0] or mat1S[1] != mat2S[1]:
    print("Matrix addition is not possible")
else:
    result = []
    print("Result matrix")
    for i in range(mat1S[0]):
        a = []
        for j in range(mat1S[1]):
            a.append(mat1[i][j] + mat2[i][j])
        print(*a)
        result.append(a)
```

```
Matrix 1 size: 2 2
Matrix 2 size: 2 2
Enter mat1 elements
2
4
6
8
Enter mat2 elements
1
3
```

```
5
7
Result matrix
3 7
11 15


'''
Use python programming to implement the binary search by using the methods[take the input
a. Recursive method
b. Iterative method
'''
def recursive(nums, target, start, end):
    if end >= start:
        m = (end+start)//2
        if nums[m] == target:
            return True
        elif nums[m] > target:
            return recursive(nums, target, start, m-1)
        else:
            return recursive(nums, target, m+1, end)
    else:
        return False

nums = list(map(int, input("Enter sorted list of numbers separated by space: ").split()))
target = int(input("Enter number to find in list: "))

# recursive method
if recursive(nums, target, 0, len(nums)-1):
    print("Found recursively !!!!")
else:
    print("Not Found Recursively ---")

start, flag, end = 0, 1, len(nums)-1
while start <= end:
    m = (start+end)//2
    if nums[m] == target:
        print("Found num using iterative method!!")
        flag = 0
        break
    elif nums[m] > target:
        end = m-1
    elif nums[m] < target:
        start = m+1
    else:
      break
if flag:
    print("Not Found target...")

 Enter sorted list of numbers separated by space: 2 4 6 8 10 13 133
 Enter number to find in list: 10
 Found recursively !!!!
 Found num using iterative method!!


'''
```

```
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 t
equal to its corresponding element in 2nd array.

'''

import numpy as np

# a

randArray1 = np.random.randint(low=50, high=101, size=(10))
print(randArray1)
randArray2 = np.random.randint(low=50, high=101, size=(10))
print(randArray2)
print(randArray1 > randArray2)
for i in range(10):
    if randArray1[i] >= randArray2[i]:
        print(i)
```

```
 [50 75 97 94 64 76 98 69 89 90]
 [ 93  71  95  81 100  87  63  73  80  60]
 [False  True  True  True False False  True False  True  True]
 1
 2
 3
 6
 8
 9
```

```
# b. Create a 1-D array and perform the following:
primaryArray = np.arange(20, 30)
print(primaryArray)
```

```
 [20 21 22 23 24 25 26 27 28 29]
```

```
import copy

# i. Replace all even numbers in the array with 0
array = copy.deepcopy(primaryArray)
for i in range(len(array)):
    if array[i]%2 == 0:
        array[i] = 0
print(array)
```

```
 [ 0 21  0 23  0 25  0 27  0 29]
```

```
# ii. Extract the prime numbers from the array
def isPrime(num):
    if num%2 == 0:
        return False
    temp = int(num**0.5)
    for i in range(3, temp, 2):
        if num%i == 0:
```

```
            return False
    return True

array = copy.deepcopy(primaryArray)
for i in array:
    if isPrime(i):
        print(f"{i} is prime")
```

```
 23 is prime
 25 is prime
 29 is prime
```

```
# iii. Convert the 1D array to a 2D array in 2 rows Input
array = copy.deepcopy(primaryArray).reshape((2,5))
print(array)
```

```
 [[20 21 22 23 24]
  [25 26 27 28 29]]
```

```
# iv. Display the array element indices such that array elements are sorted in ascending o
array = np.random.randint(low=50, high=101, size=(10))
print(array)
array2 = np.argsort(array)
print(array2)
```

```
 [75 82 65 79 62 89 84 72 75 55]
 [9 4 2 7 0 8 3 1 6 5]
```

```
# v. Convert a binary NumPy array (holding only 0s and 1s) to a Boolean NumPy array.
bArray = np.array([0,1,0,0,0,1,1])
boolArray = np.array(bArray, dtype=bool)
print(boolArray)
```

```
 [False  True False False False  True  True]
```

```
# vi. Take an input of 10 elements and split the array into 3 arrays, where 1st two arrays
# Display the arrays.
inputArray = list(map(int, input("Enter 10 elements separated by space: ").split()))
array1 = inputArray[:2]
array2 = inputArray[2:4]
array3 = inputArray[4:]
print(array1)
print(array2)
print(array3)
```

```
 Enter 10 elements separated by space: 1 2 3 4 5 6 7 8 9 0
 [1, 2]
 [3, 4]
 [5, 6, 7, 8, 9, 0]
```

```
'''
There are 190 students in a class of Data Science Theory. The subject is taught every day
Sunday) in a week for an hour. Create and display a series of data as a count of attendanc
```

```
number of students attending the subject every day in a week. [Hint: Use pandas to create
dataset, create the dataset for a week i.e. for all 7 days in a week, for each respective
number of attendees.] Perform the following with the series dataset created.
'''

import pandas as pd
# import numpy as np

# data_ = np.array([['Monday', 100], ['Tuesday', 120], ['Wednesday', 110], ['Thrusday', 12
# columnNames = ['Weekday', 'attendees']

data = {
    'weekday': ['monday', 'tuesday', 'wednesday', 'thrusday', 'friday', 'saturday', 'sunda
    'attendees': [100, 120, 110, 120, 120, 100, 50]
}

Data = pd.DataFrame(data = data, columns = [i for i in data.keys()])
print(Data)
```

```
     weekday  attendees
0     monday        100
1    tuesday        120
2  wednesday        110
3   thrusday        120
4     friday        120
5   saturday        100
6     sunday         50
```

```
# a. Display the dataset
print(Data)
```

```
     weekday  attendees
0     monday        100
1    tuesday        120
2  wednesday        110
3   thrusday        120
4     friday        120
5   saturday        100
6     sunday         50
```

```
# b. Display the sorted dataset with least number of attendees at first
Data.sort_values(by=['attendees'], ascending=False, inplace=True)
print(Data)
```

```
     weekday  attendees
1    tuesday        120
3   thrusday        120
4     friday        120
2  wednesday        110
0     monday        100
5   saturday        100
6     sunday         50
```

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

```
print(Data.loc[Data['attendees'] == Data['attendees'].max()])
```

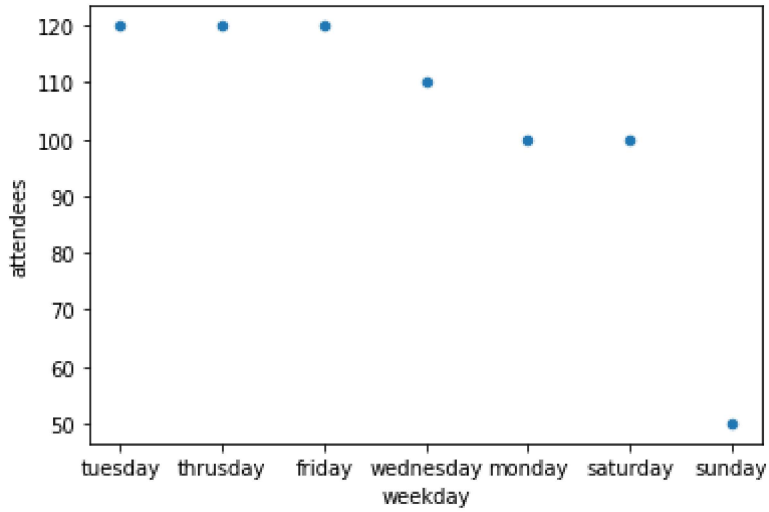
```
     weekday  attendees
1    tuesday        120
3   thrusday        120
4     friday        120
```

```
# d. Display the 1st two days of the week and the number of attendees
Data_ = pd.DataFrame(data = data, columns = [i for i in data.keys()])
print(Data_.head(2), '\n')
```

```
    weekday  attendees
0    monday        100
1   tuesday        120
```

```
# e. Plot the dataset for each day in the week.
Data.plot(x='weekday', y='attendees', kind='scatter')
```

```
<matplotlib.axes._subplots.AxesSubplot at 0x7f50a21691d0>
```



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

```
dataSet = pd.read_csv('/content/sample_data/salaryData.csv')
# i downloaded the csv file from the website given and uploaded it here in the folders sec
# data is read from the local csv file
```

```
# a. Read the dataset
print(dataSet)
```

```
    YearsExperience  Salary
0               1.1   39343
1               1.3   46205
2               1.5   37731
3               2.0   43525
4               2.2   39891
5               2.9   56642
6               3.0   60150
```

```
7               3.2   54445
8               3.2   64445
9               3.7   57189
10              3.9   63218
11              4.0   55794
12              4.0   56957
13              4.1   57081
14              4.5   61111
15              4.9   67938
16              5.1   66029
17              5.3   83088
18              5.9   81363
19              6.0   93940
20              6.8   91738
21              7.1   98273
22              7.9  101302
23              8.2  113812
24              8.7  109431
25              9.0  105582
26              9.5  116969
27              9.6  112635
28             10.3  122391
29             10.5  121872
```

```
# b. Display the information related to the dataset such as the number of rows and columns
dataSet.info()
# print(f"\ncolumns: {len(dataSet.columns)}")
# print(f"Rows: {len(dataSet)}")
```

```
<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     int64  
dtypes: float64(1), int64(1)
memory usage: 608.0 bytes
```

```
# c. Display the first 5 rows
dataSet.head(5)
```

	YearsExperience	Salary
0	1.1	39343
1	1.3	46205
2	1.5	37731
3	2.0	43525
4	2.2	39891

```
# d. Display the summary statistics for each numeric column
dataSet.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)
dataSet.sample(5)
```

	YearsExperience	Salary
21	7.1	98273
26	9.5	116969
3	2.0	43525
14	4.5	61111
22	7.9	101302