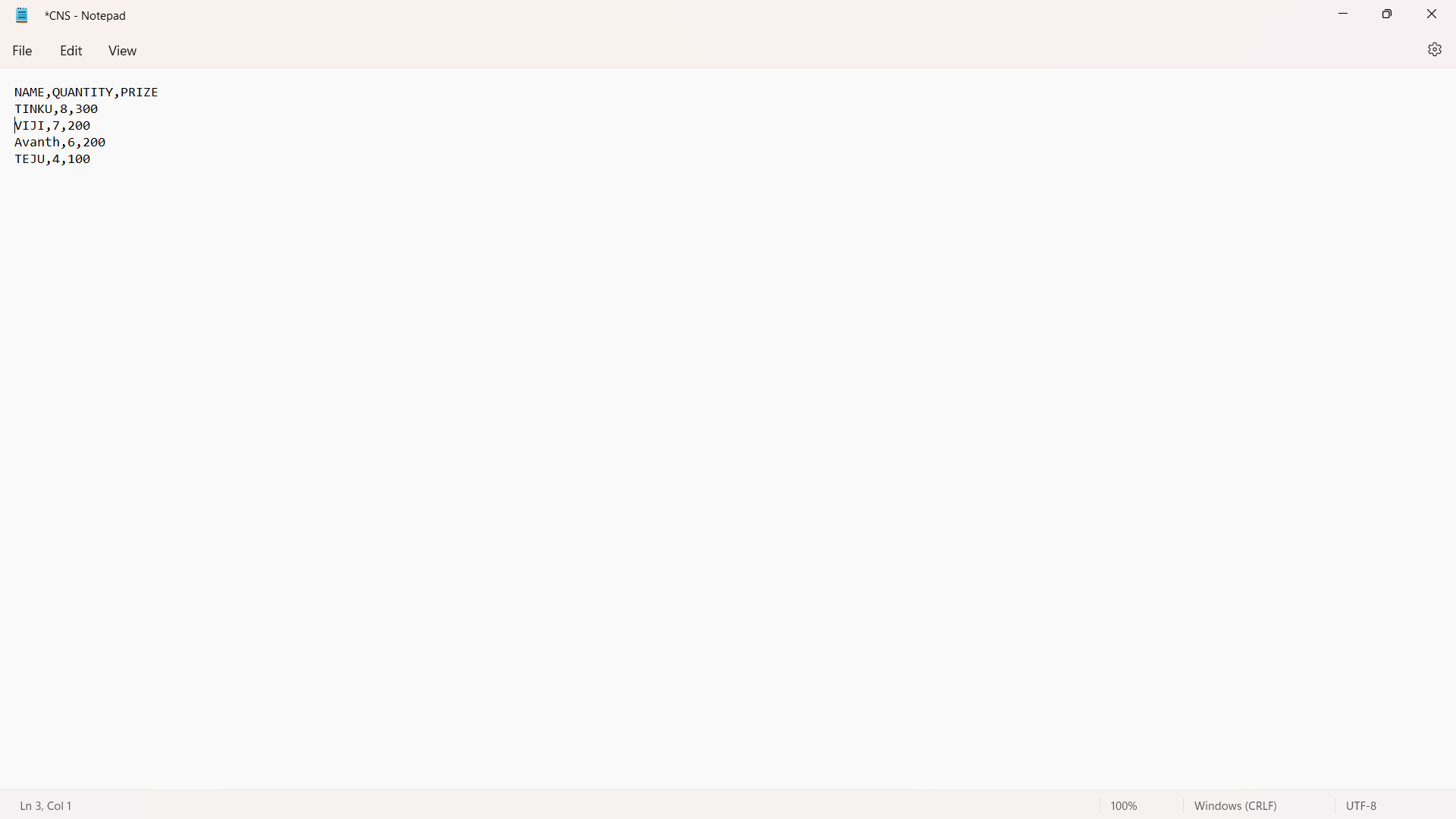
1.CSV FILE

A CSV (Comma Separated Values) format is one of the most simple and common ways to store tabular data. To represent a CSV file, it must be saved with the **.csv** file extension.

EXAMPLE:

FIRST we have to create a date using notepad or anyother text editor.



As you can see, the elements of a CSV file are separated by commas. Here,  is a delimiter.While we could use the built-in open() function to work with CSV files in Python, there is a dedicated csv module that makes working with CSV files much easier.

We need to import the module first:

import csv

**Reading CSV files Using csv.reader()**

import csv,operator

sortedlist=[]

csv.register\_dialect("mydialect",delimiter=',',skipinitialspace=True)

file=open("CNS.csv",'r')

filereader=csv.reader(file,dialect="mydialect")

next(filereader)

sortedlist=sorted(filereader,key=operator.itemgetter(0))

for row in sortedlist:

print(row)

file.close()

**Output:**



## Writing CSV files Using csv.writer()

To write to a CSV file in Python, we can use the csv.writer() function.

The csv.writer() function returns a writer object that converts the user's data into a delimited string. This string can later be used to write into CSV files using the writerow() function. Let's take an example.

import csv

info=[['SNO','NAME','AGE'],[1,'TINKU','20'],[2,'THARUN',20],[3,'TEJU',20],[4,'AVANTH',60]]

csv.register\_dialect("mydialect",quoting=csv.QUOTE\_ALL,delimiter=',')

file=open('age.csv','w')

filewriter=csv.writer(file,dialect="mydialect")

for row in info:

filewriter.writerow(row)

file.close()

**Output:**

"SNO","NAME","AGE"

"1","TINKU","20"

"2","THARUN","20"

"3","TEJU","20"

"4","AVANTH","60"

In the above program, we have opened the file in writing mode.

Then, we have passed each row as a list. These lists are converted to a delimited string and written into the CSV file.

2.PLOT & TYPES

Plot: Plots also known as charts is the visual representation of data in a graphical format for the better understanding of the data.By representing the data in graphical form we can get a fair idea of what the data is trying to show. We can use the Matplotlib visualization library in Python to portray the graphs.

Types: The various types of plots are as follows,

• Scatter plot

• Bar plot

• Pie plot

• Line plot

• Histogram plot

LINE GRAPH:

A LINE graph is a type of graph which displays data as a series of data points called markers connected by straight lines

Example:

import matplotlib.pyplot as plt

year=[1920,11930,1940,1950,1960,1970,1980,1990,2000]

unem=[9.8,12,2.2,2,4,5,6,7,8]

plt.plot(year,unem,marker='o')

plt.xlabel("year")

plt.ylabel("unem")

plt.title("year vs unemply rate")

plt.show()



Scatter plot:

We use scatter() function to plot one dot for each observation.It requires both x and y arrays are of the same length.

Example:

from matplotlib import pyplot as plt

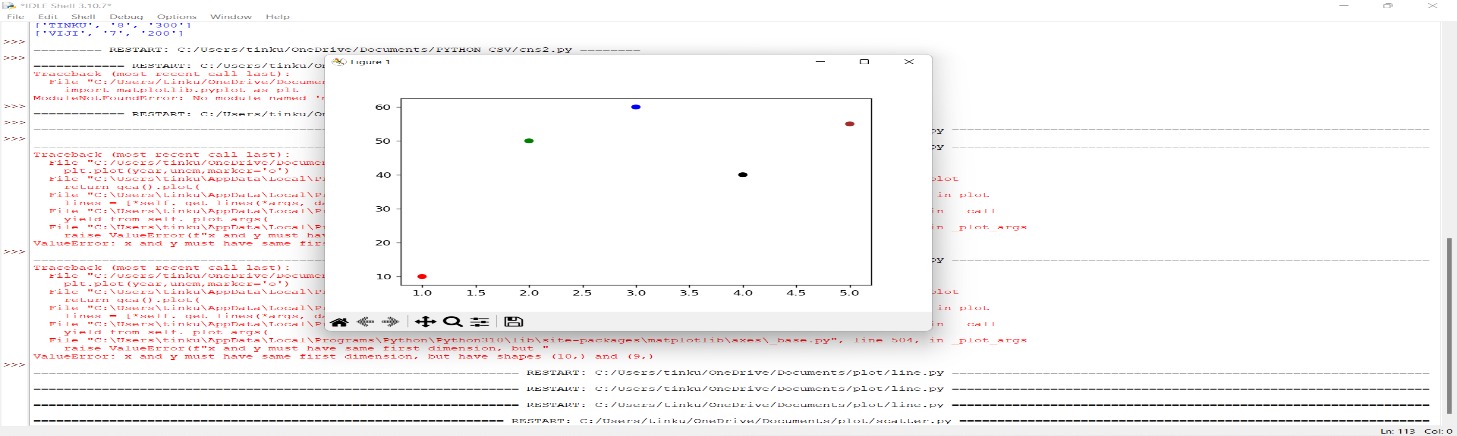
x=[1,2,3,4,5]

y=[10,50,60,40,55]

c=["red","green","blue","black","brown"]

plt.scatter(x,y,c=c)

plt.show()



Bar plot:

We use bar() function to represent data with rectangular bars with length and height that is proportional to the value which they represent.They can be plotted both horizontally and vertically.

Example:

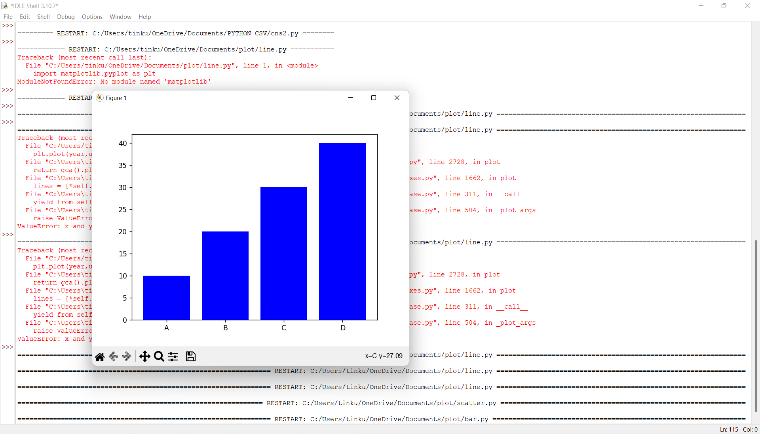
from matplotlib import pyplot as plt

x=['A','B','C','D']

y=[10,20,30,40]

plt.bar(x,y,color="blue")

plt.show()



Pie plot:

A Pie plot is a circular statistical plot that can display only one series of data. The area of the chart is the total percentage of the given data. The area of slices of the pie represents the percentage of the parts of the data. The slices of pie are called wedges.

Example:

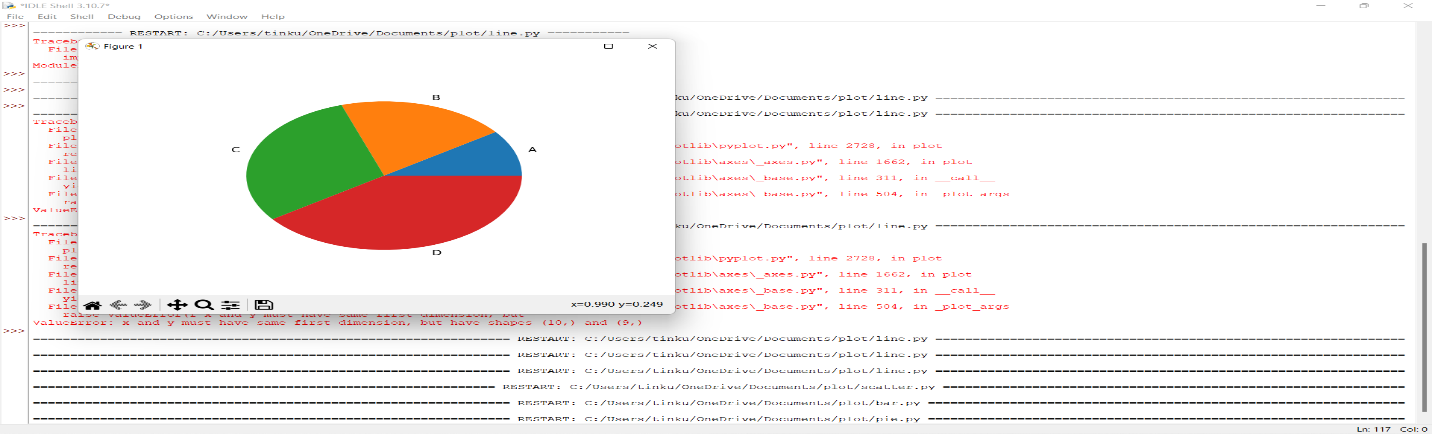
from matplotlib import pyplot as plt

x=['A','B','C','D']

y=[10,20,30,40]

plt.pie(y,labels=x)

plt.show()



Histogram plot:

A histogram is a graph showing frequency distributions.It is a graph showing the number of observations within each given interval. It is a type of bar plot where X-axis represents the bin ranges while Y-axis gives information about frequency

Example:

from matplotlib import pyplot as plt

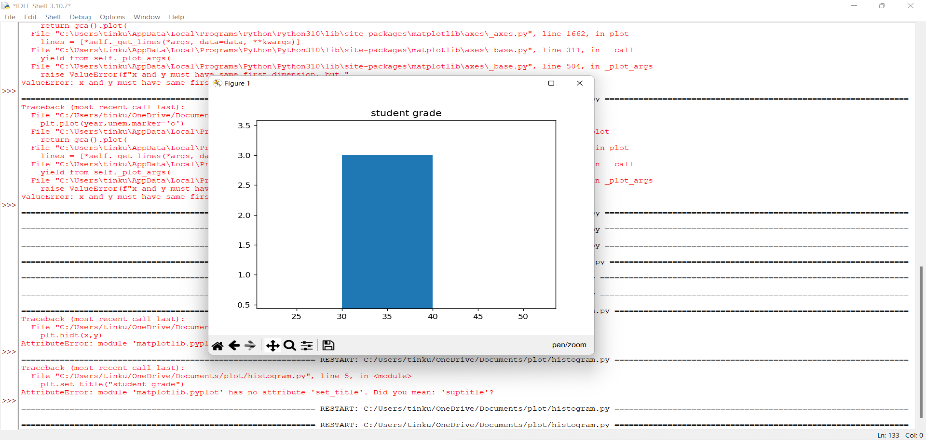
x=[90,30,50,60,34,35,60]

y=[10,20,30,40]

plt.hist(x,y)

plt.title("student grade")

plt.show()



HACKERRANK PROBLEM

1.ARRAY MANIPULATION

QUESTION:

Starting with a 1-indexed array of zeros and a list of operations, for each operation add a value to each the array element between two given indices, inclusive. Once all operations have been performed, return the maximum value in the array

SAMPLE INPUT:

5 3

1 2 100

2 5 100

3 4 100

SAMPLE OUTPUT:

200

SOLUTION:

arr=[0]\*(n+2)

    for a,b,k in queries:

        arr[a]+=k

        arr[b+1]-=k

    maxnum=temp=0

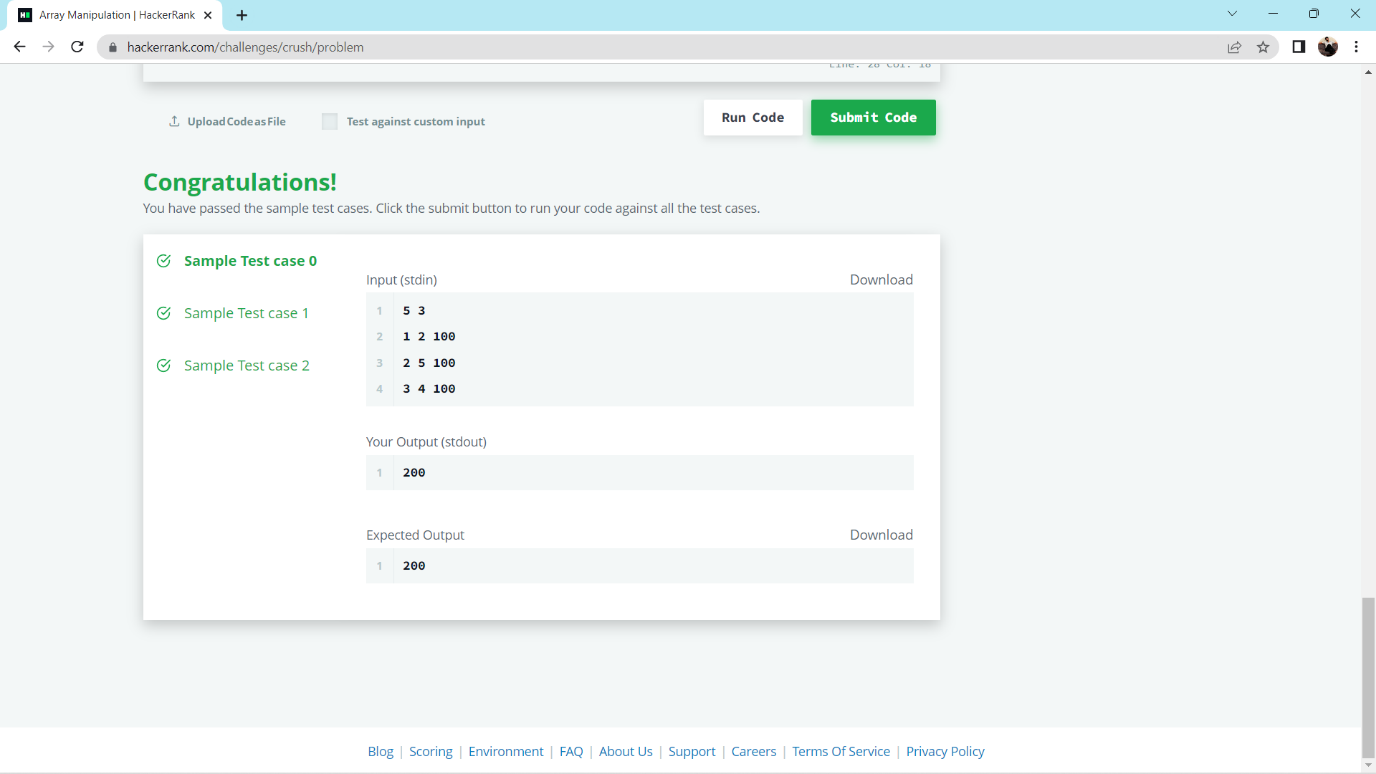
    for val in arr:

        temp+=val

        maxnum=max(maxnum,temp)

    return maxnum

OUTPUT:



2.

Watson gives Sherlock an array of integers. His challenge is to find an element of the array such that the sum of all elements to the left is equal to the sum of all elements to the right.

Sample input:

2

3

1 2 3

4

1 2 3 3

Output:

NO

YES

Solutiom:

right=sum(arr)

    left=0

    for num in arr:

        right-=num

        if right==left:

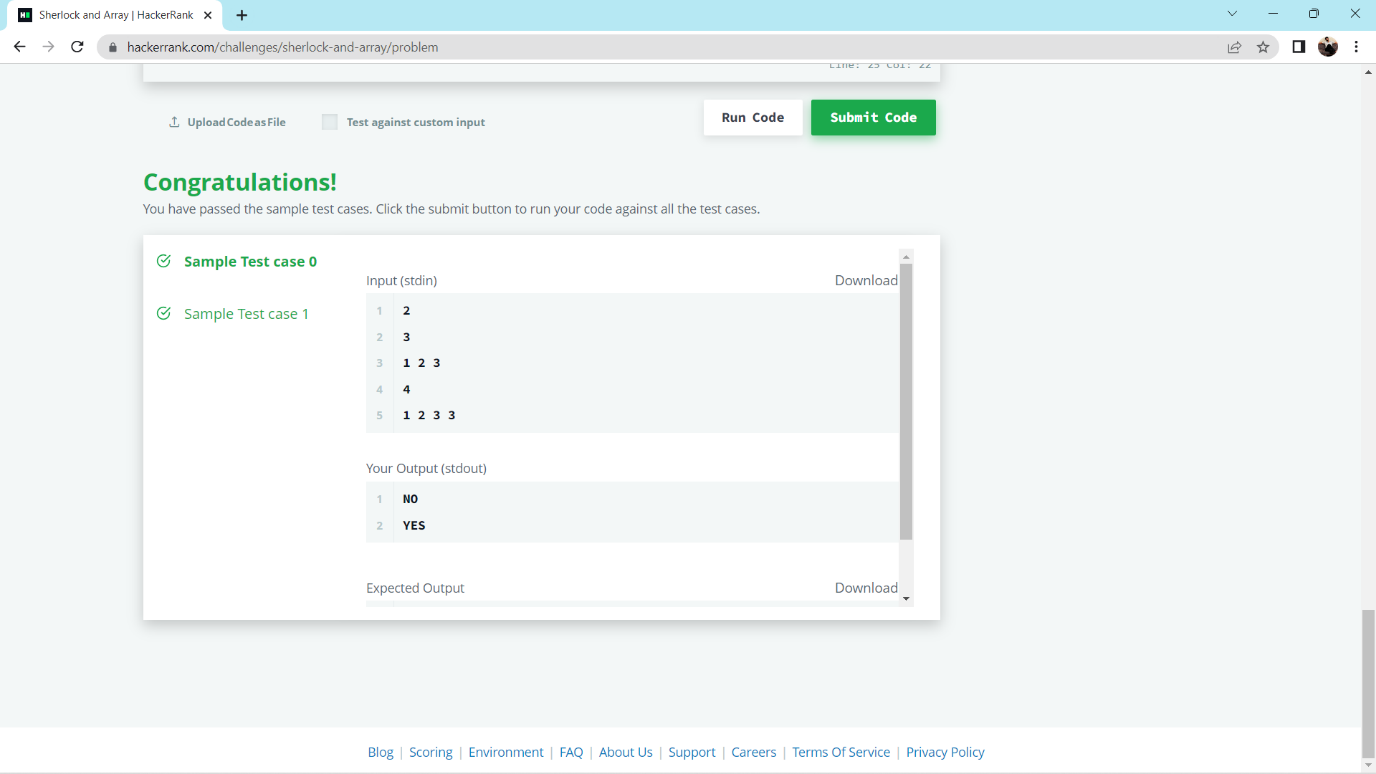
            return "YES"

        else:

            left+=num

    return "NO"

output:



**NUMPY**

1.It is a mathematical module in python which has various mathematical functions. 2.It is used to create multidimensional array. NumPy is a library for the Python programming language, adding support for large, multi-dimensional arrays and matrices, along with a large collection of highlevel mathematical functions to operate on these arrays.

The traditional python list serve the purpose of creating array but they are very much slower.So the great alternative for list is using Numpy. They aim to provide an array object that is more times faster than list. Numpy are faster than list because numpy arrays are stored in one continuous place in memory unlike lists.

Creating arrays using numpy:

We can create numpy ndarray using the array() function.

Eg: import numpy as np

a=np.array( [ [ 1,2,3,4 ],[ 5,6,7,8 ] ] )

print (a)

Numpy array indexing:

We can access the array elements by using the index number associated with it either it can be a one dimensional or two dimensional array.

Eg: For one dimensional array => a[0]

For two dimensional array => a[0,1]

For three dimensional array => a[0,1,2]

Negative indexing => a[1,-1]

To find shape of the array:

Using the shape attribute we can find the number of elements present in the array.

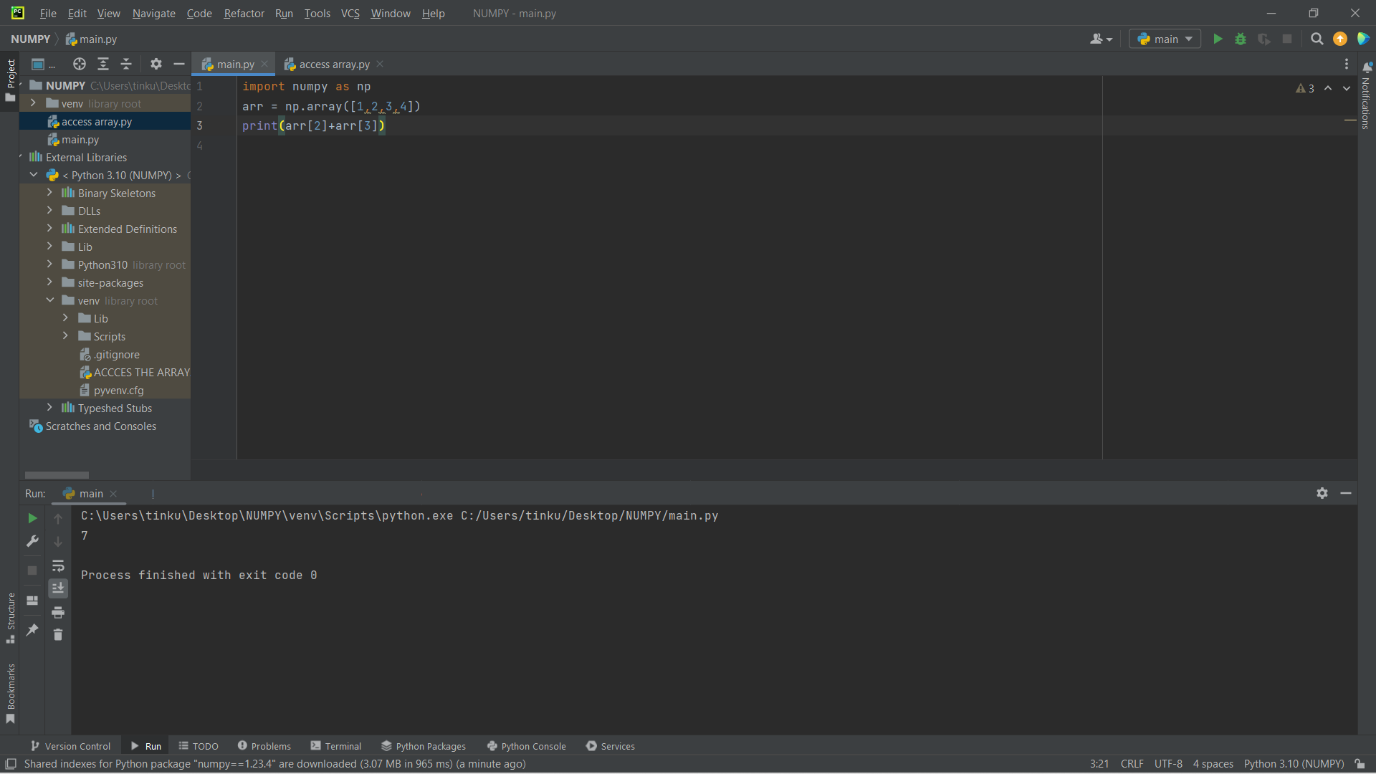
Eg: print (a.shape)

The example above returns (2,4) where the array has 2 dimensions and the first dimension has 2 elements and the second dimension has 4 elements

EXERCISES:

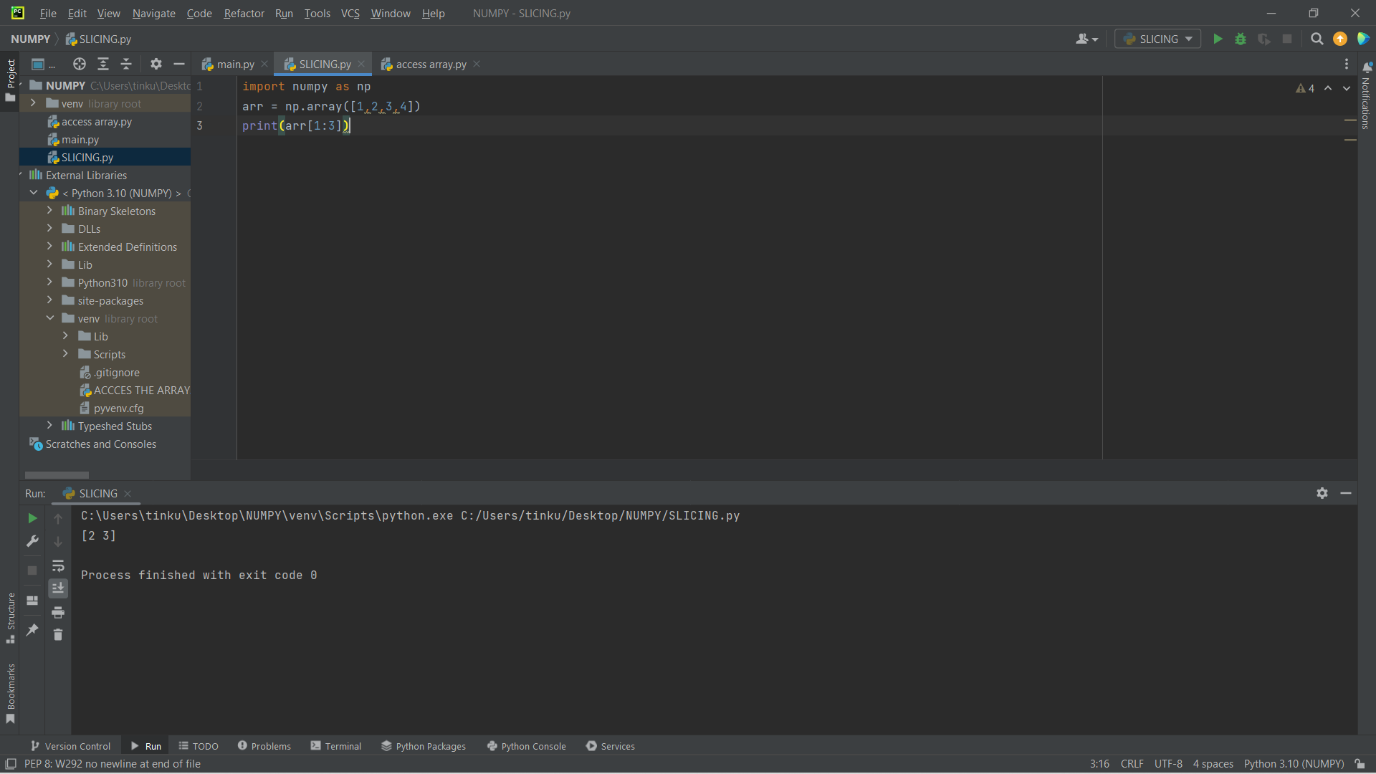
ACCESS THE ARRAYS:

PROGRAM



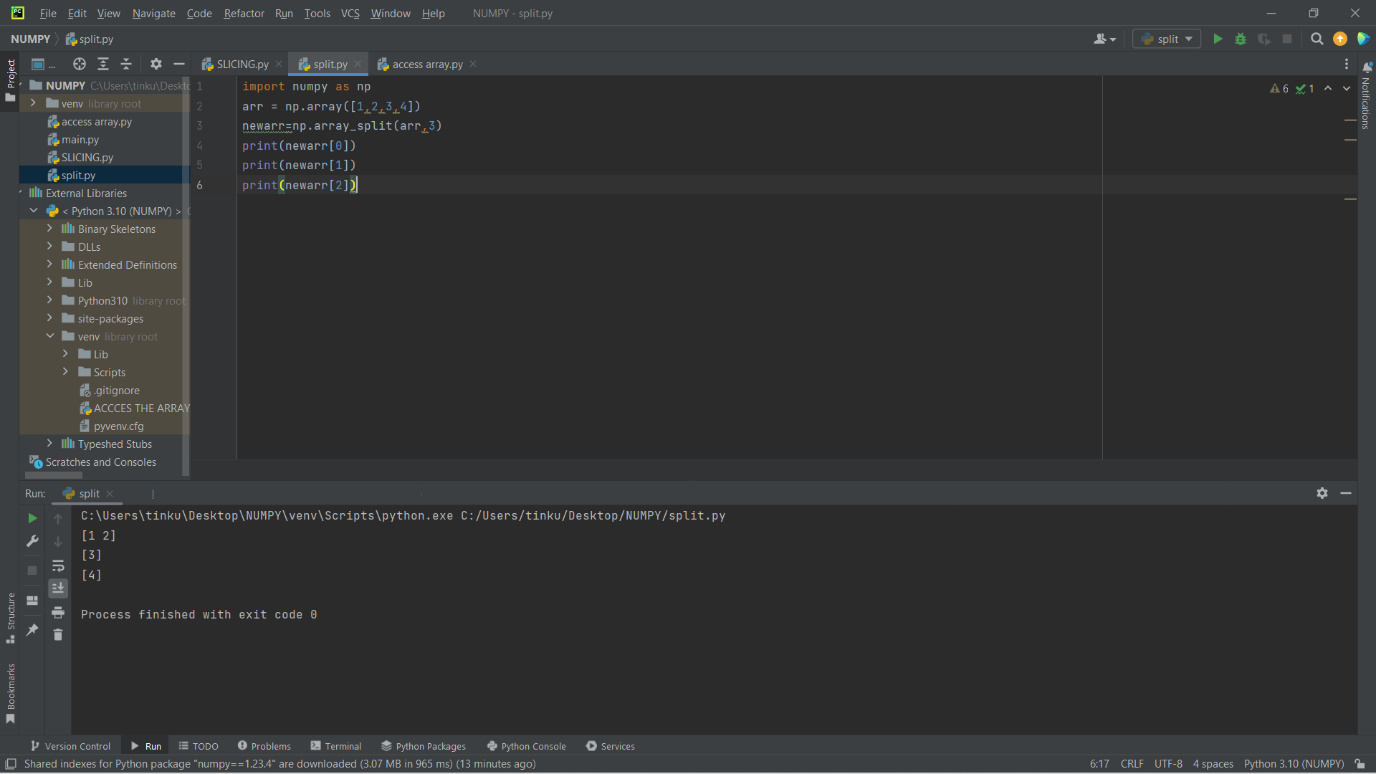
SLICING ARRAYS:

PROGRAM:



SPLITING THE ARRAYS

PROBLEM



1&2 DIMENSION

PROGRAM:

