डॉ. ए.पी.जे. अब्दुल कलाम प्रौद्योगिकी संस्थान, टनकपुर, उत्तराखंड

(संघटक संस्थान, उत्तराखंड तकनीकी विश्वविद्यालय, उत्तराखंड सरकार)

Dr. A.P.J. Abdul Kalam Institute of Technology, Tanakpur, Uttarakhand

(A Constituent Institute of Uttarakhand Technical University, Govt. of Uttarakhand)

Approved by AICTE, MHRD New Delhi

LAB FILE

Academic Session
(2022-23)

Programme	:	B. Tech in Computer Science and Engineering
Course	:	Computer Workshop (Python Programming)
NO. of credits	:	1
Semester	:	III
Session	:	July-Dec 2022
Batch	:	2021-2025
Faculty	:	Dr. Hitesh Kumar Sharma

Submitted By:

Roll No: 211620101022

Name: CHETAN PANDEY

Submitted to:

Dr. Hitesh Kumar Sharma

Dept. of Computer Science

LIST OF EXPERIMENTS

Exp No.	Experiment Name	Date of Experiment
1	How to declare and use variables and operators	
2	Programming using Basic Libraries (Numpy, Pandas, SK Learn etc)	
3	To write a Python program to print HELLO INDIA.	
4	To write a Python program that takes in command line arguments as input and print the number of arguments.	
5	To write a Python program find the division of student.	
6	To write a program implements Fibonacci series.	
7	To write a Python program for factorial.	
8	To write a Python program to use of functions.	
9	To write a Python program to implement list.	
10	To write a Python program to implement tuples.	
11	To write a Python program Insertion sort.	
12	To write a Python program merge sort.	
13	To write a Python program first n prime numbers.	
14	Implementation of Data Science concepts using Python	

LAB EXERCISE 1

Aim: How to declare and use variables and operators

Description:

In Python, we can declare a variable by simply assigning a value to it. For example:

```
x = 10
y = "Hello, World!"
```

Here, x is an integer variable and y is a string variable.

we can also use various operators in Python to perform operations on variables. Some common operators include:

+: Addition
-: Subtraction
*: Multiplication
/: Division
%: Modulus (remainder after division)
For example:

```
a = 10
b = 20

c = a + b     # c is now 30
d = a - b     # d is now -10
e = a * b    # e is now 200
f = b / a     # f is now 2.0
g = b % a   # g is now 0
```

we can also use comparison operators to compare the values of two variables. Some common comparison operators include:

==: Equal to

!=: Not equal to

<: Less than

>: Greater than

<=: Less than or equal to

>=: Greater than or equal to

For example:

```
h = 10
i =  20

if  h < i:
  print("h is less than i")
else:
  print("h is greater than or equal to i")
```

This would print "h is less than i" because the value of h is indeed less than the value of i

LAB EXERCISE 2

Aim: Programming using Basic Libraries (Numpy, Pandas, SK Learn etc)

Description:

Pandas

Pandas is a very popular library for working with data (its goal is to be the most powerful and flexible open-source tool, and in our opinion, it has reached that goal). DataFrames are at the center of pandas. A DataFrame is structured like a table or spreadsheet. The rows and the columns both have indexes, and you can perform operations on rows or columns separately.

A pandas DataFrame can be easily changed and manipulated. Pandas has helpful functions for handling missing data, performing operations on columns and rows, and transforming data. If that wasn't enough, a lot of SQL functions have counterparts in pandas, such as join, merge, filter by, and group by. With all of these powerful tools, it should come as no surprise that pandas is very popular among data scientists.

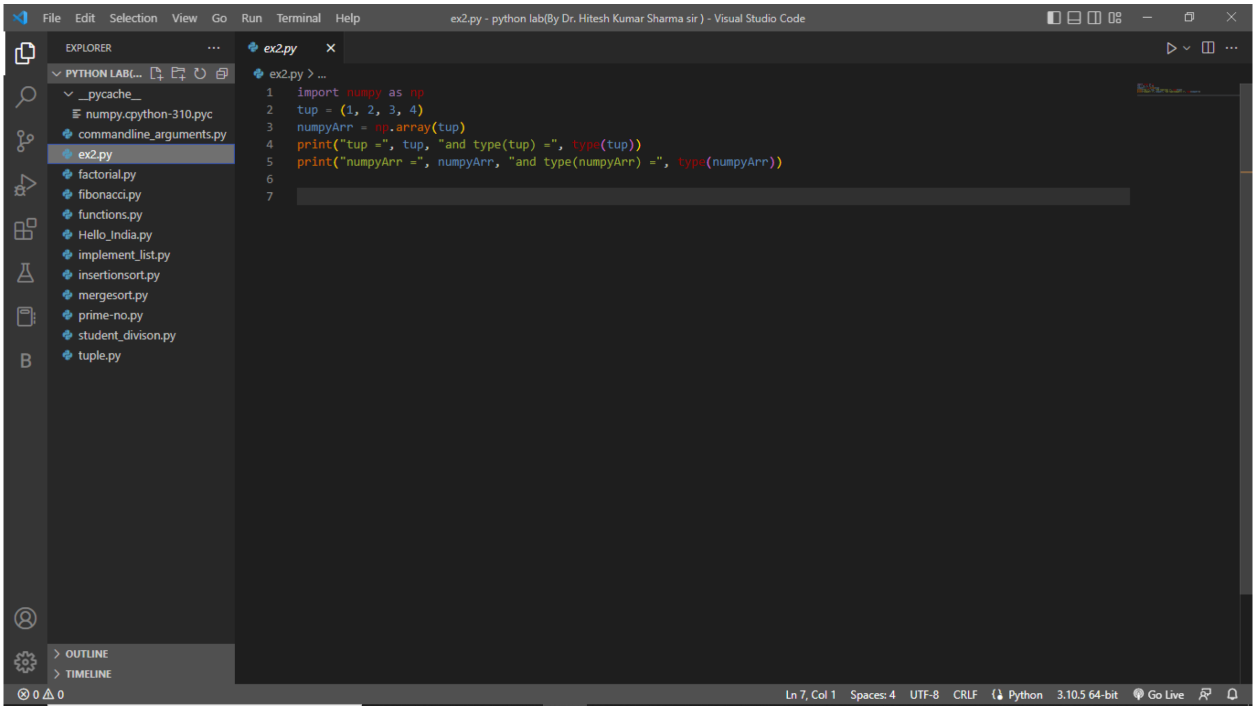
NumPy

NumPy is an open-source Python library that facilitates efficient numerical operations on large quantities of data. There are a few functions that exist in NumPy that we use on pandas DataFrames. For us, the most important part about NumPy is that pandas is built on top of it. So, NumPy is a dependenc

Command to install: pip install numpy

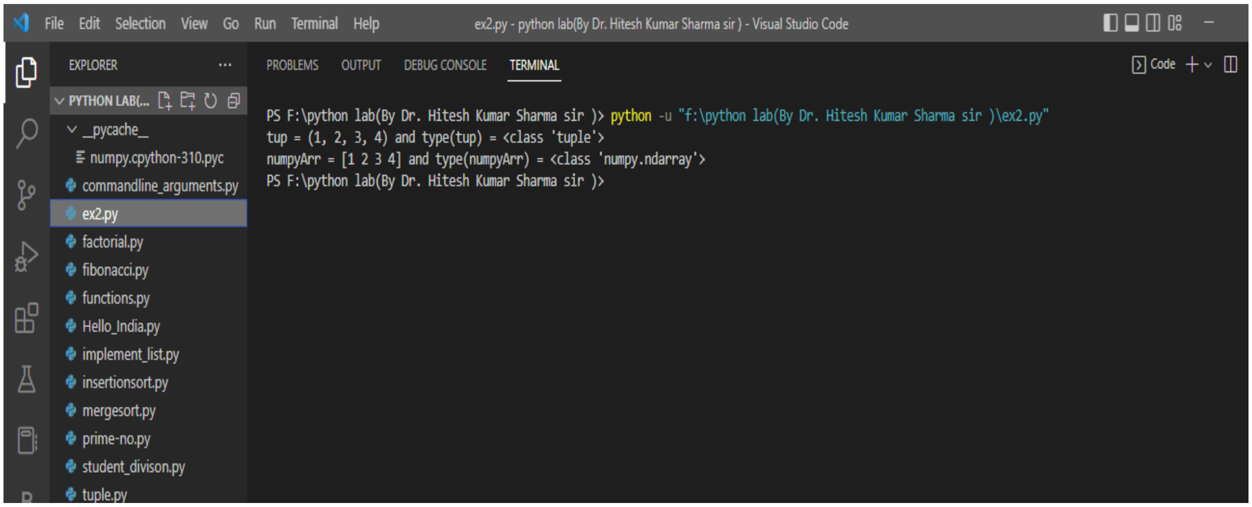
Solution:

Code:



```python
import numpy as np
tup = (1, 2, 3, 4)
numpyArr = np.array(tup)
print("tup =", tup, "and type(tup) =", type(tup))
print("numpyArr =", numpyArr, "and type(numpyArr) =", type(numpyArr))
```

Input/Output:



ex2.py - python lab(By Dr. Hitesh Kumar Sharma sir) - Visual Studio Code
PROBLEMS
OUTPUT
DEBUG CONSOLE
TERMINAL

```
PS F:\python lab(By Dr. Hitesh Kumar Sharma sir )> python -u "f:\python lab(By Dr. Hitesh Kumar Sharma sir )\ex2.py"
tup = (1, 2, 3, 4) and type(tup) = <class 'tuple'>
numpyArr = [1 2 3 4] and type(numpyArr) = <class 'numpy.ndarray'>
PS F:\python lab(By Dr. Hitesh Kumar Sharma sir )>
```

LAB EXERCISE 3

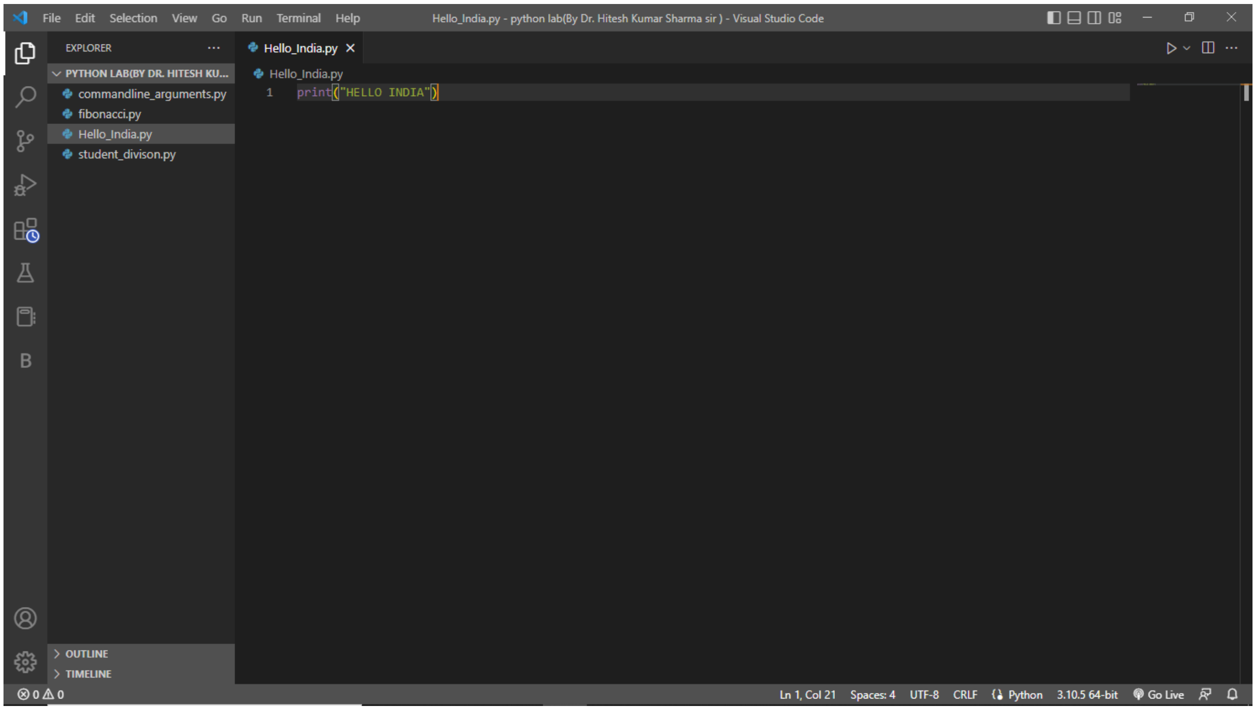
Aim: To write a Python program to print HELLO INDIA.

Description:

In this program, we have used the built-in function to print the string HELLO INDIA on our screen.

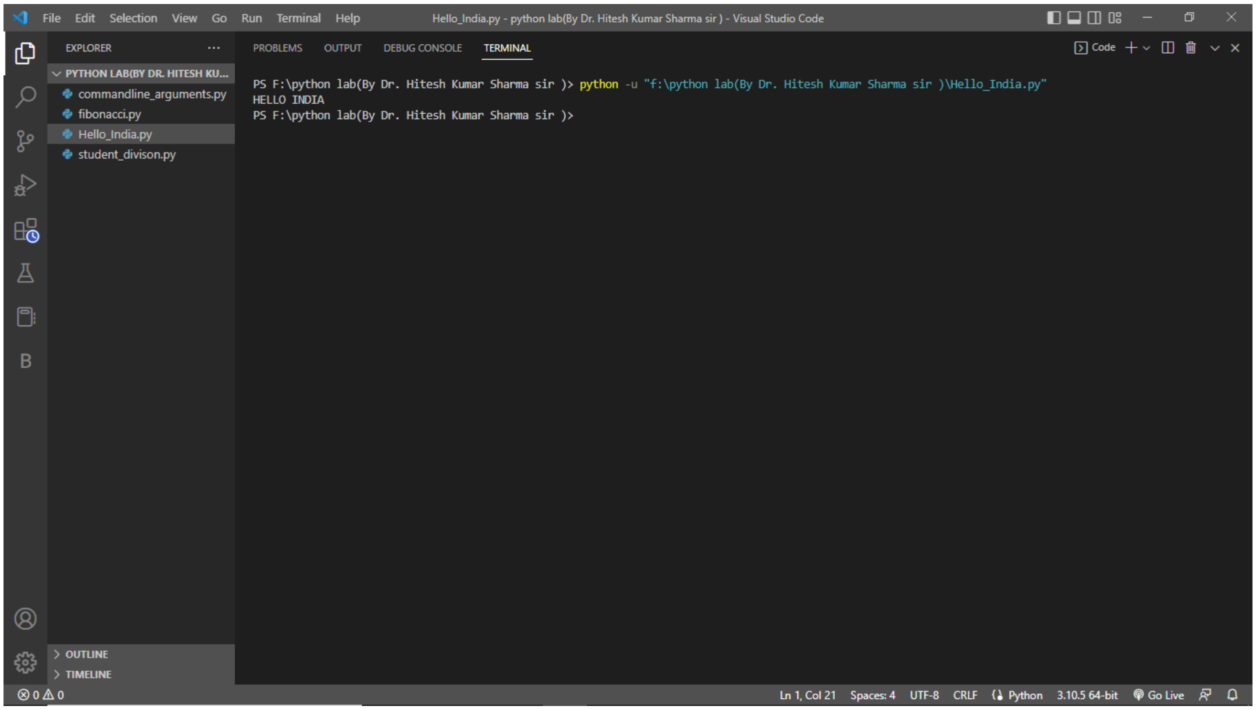
Solution:

Code:



```python
print("HELLO INDIA")
```

Input/Output:



```
PS F:\python lab(By Dr. Hitesh Kumar Sharma sir )> python -u "f:\python lab(By Dr. Hitesh Kumar Sharma sir )\Hello_India.py"
HELLO INDIA
PS F:\python lab(By Dr. Hitesh Kumar Sharma sir )>
```

LAB EXERCISE 4

Aim: To write a Python program that takes in command line arguments as input and print the number of arguments.

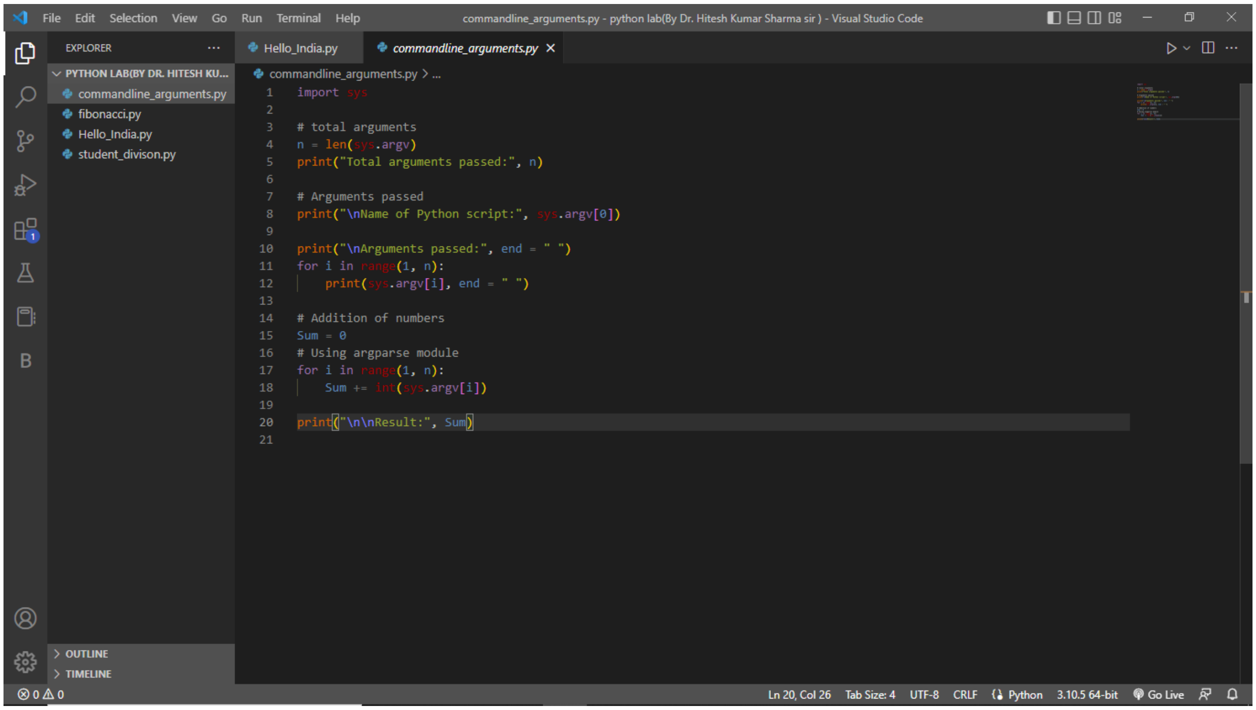
Description:

The argument that are given after the name of program in the command line shell of the operating system are known as Command Line Arguments.
One such variable is sys.argv which is a simple list structure. It’s main purpose are:

- It is a list of command line arguments.
- len(sys.argv) provides the number of command line arguments.
- sys.argv[0] is the name of the current Python script.

Solution:

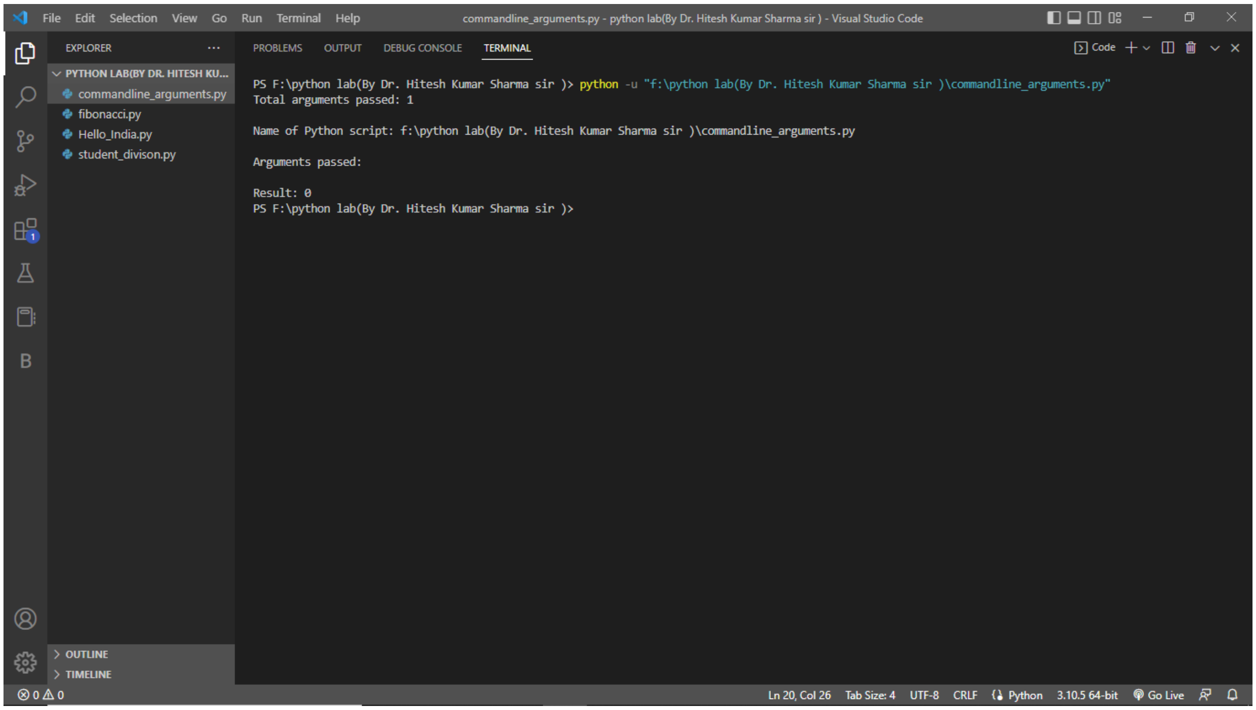
Code:



File Edit Selection View Go Run Terminal Help
commandline_arguments.py - python lab(By Dr. Hitesh Kumar Sharma sir) - Visual Studio Code
EXPLORER
PYTHON LAB(BY DR. HITESH KU...
commandline_arguments.py
fibonacci.py
Hello_India.py
student_divison.py
Hello_India.py
commandline_arguments.py
commandline_arguments.py > ...
OUTLINE
TIMELINE
Ln 20, Col 26 Tab Size: 4 UTF-8 CRLF Python 3.10.5 64-bit Go Live

```python
import sys

# total arguments
n = len(sys.argv)
print("Total arguments passed:", n)

# Arguments passed
print("\nName of Python script:", sys.argv[0])

print("\nArguments passed:", end = " ")
for i in range(1, n):
    print(sys.argv[i], end = " ")

# Addition of numbers
Sum = 0
# Using argparse module
for i in range(1, n):
    Sum += int(sys.argv[i])

print("\n\nResult:", Sum)

```

Input/Output:



```
PS F:\python lab(By Dr. Hitesh Kumar Sharma sir )> python -u "f:\python lab(By Dr. Hitesh Kumar Sharma sir )\commandline_arguments.py"
Total arguments passed: 1

Name of Python script: f:\python lab(By Dr. Hitesh Kumar Sharma sir )\commandline_arguments.py

Arguments passed:

Result: 0
PS F:\python lab(By Dr. Hitesh Kumar Sharma sir )>
```

LAB EXERCISE 5

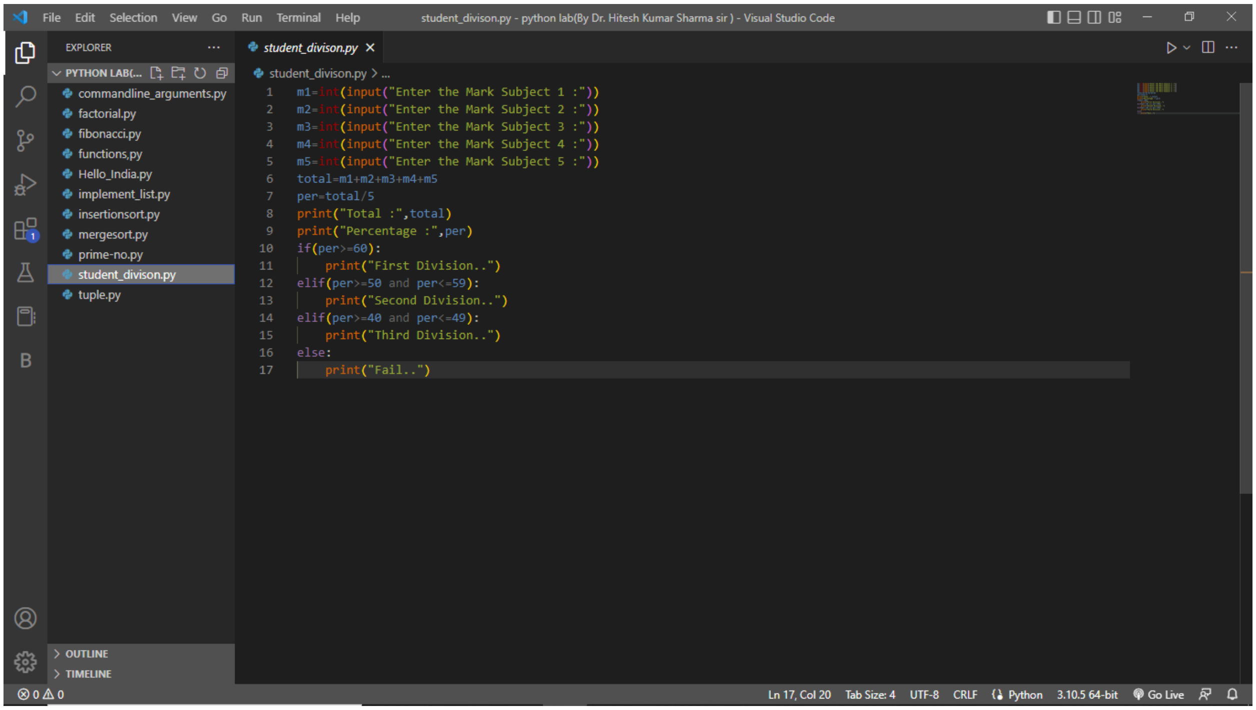
Aim: To write a Python program find the division of student.

Description:

This program finds and prints the grade of a student based on marks obtained in five subjects entered by the user at run-time.
To calculate the grade of students in Python, you have to ask the user to enter marks obtained in five subjects. Now calculate the sum of all the marks and then calculate the average marks to find the grade according to the average marks obtained by the students.

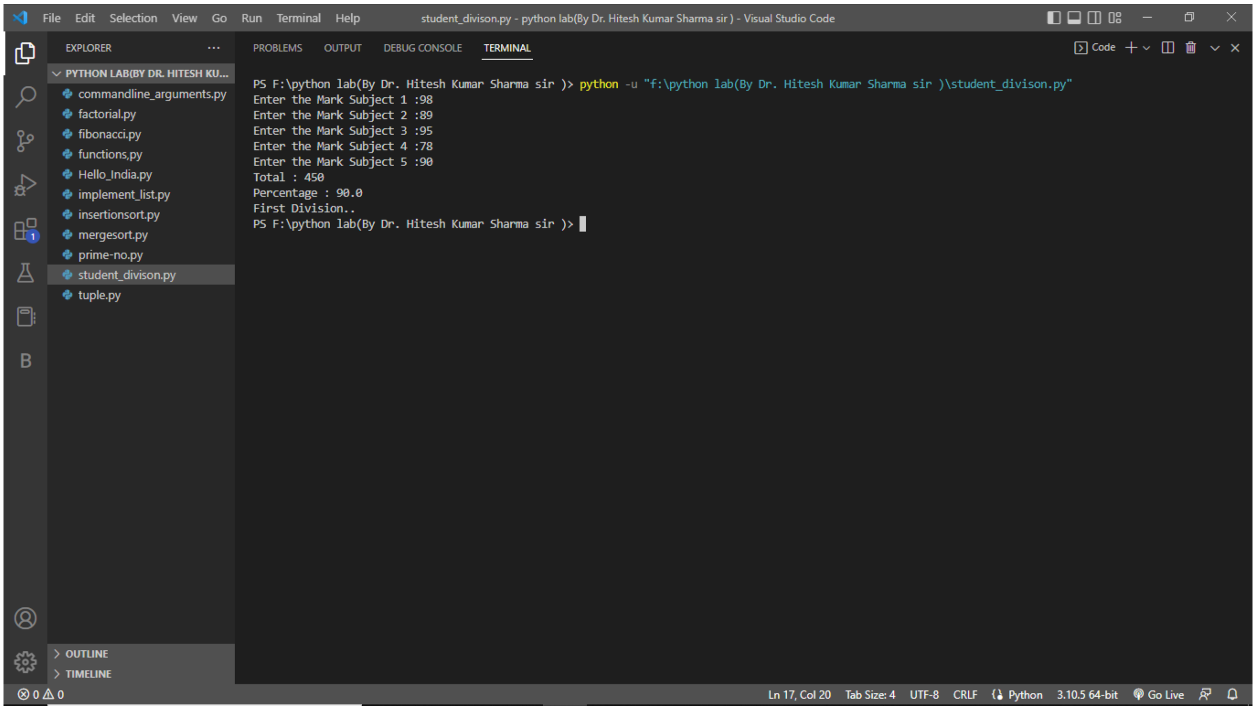
Solution:

Code:



```python
m1=int(input("Enter the Mark Subject 1 :"))
m2=int(input("Enter the Mark Subject 2 :"))
m3=int(input("Enter the Mark Subject 3 :"))
m4=int(input("Enter the Mark Subject 4 :"))
m5=int(input("Enter the Mark Subject 5 :"))
total=m1+m2+m3+m4+m5
per=total/5
print("Total :",total)
print("Percentage :",per)
if(per>=60):
    print("First Division..")
elif(per>=50 and per<=59):
    print("Second Division..")
elif(per>=40 and per<=49):
    print("Third Division..")
else:
    print("Fail..")
```

Input/Output:



```
PS F:\python lab(By Dr. Hitesh Kumar Sharma sir )> python -u "f:\python lab(By Dr. Hitesh Kumar Sharma sir )\student_divison.py"
Enter the Mark Subject 1 :98
Enter the Mark Subject 2 :89
Enter the Mark Subject 3 :95
Enter the Mark Subject 4 :78
Enter the Mark Subject 5 :90
Total : 450
Percentage : 90.0
First Division..
PS F:\python lab(By Dr. Hitesh Kumar Sharma sir )>
```

LAB EXERCISE 6

Aim: To write a program implements Fibonacci series.

Description:

The Fibonacci numbers are the numbers in the following integer sequence. 0, 1, 1, 2, 3, 5, 8, 13, 21, 34, 55, 89, 144, ……..

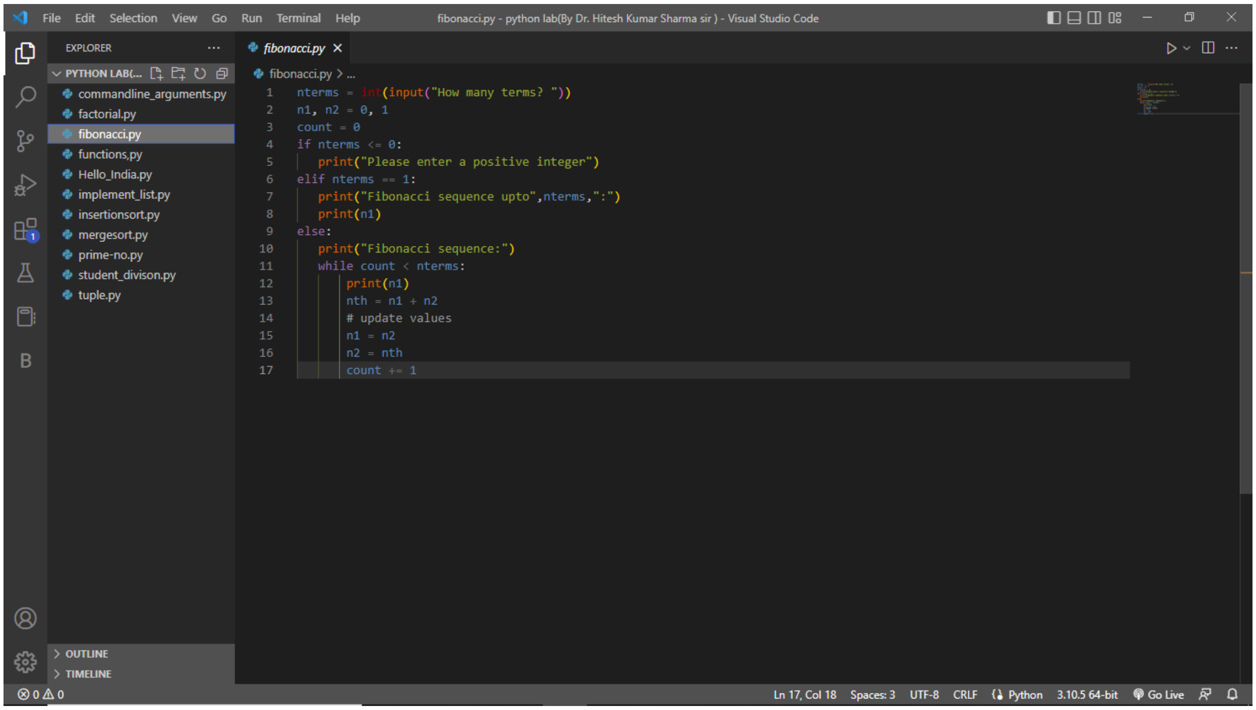
In mathematical terms, the sequence Fn of Fibonacci numbers is defined by the recurrence relation

$F_n = F_{n-1} + F_{n-2}$ with seed values

$F_0 = 0$ and $F_1 = 1$.

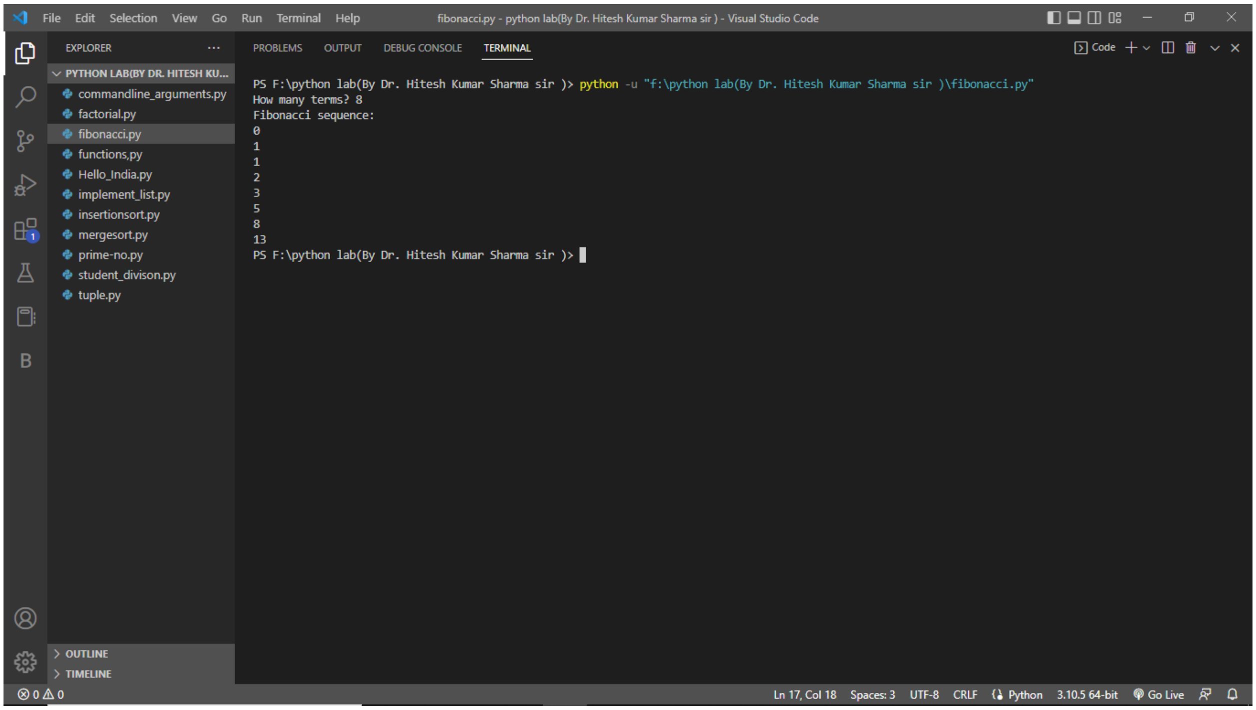
Solution:

Code:



```python
nterms = int(input("How many terms? "))
n1, n2 = 0, 1
count = 0
if nterms <= 0:
   print("Please enter a positive integer")
elif nterms == 1:
   print("Fibonacci sequence upto",nterms,":")
   print(n1)
else:
   print("Fibonacci sequence:")
   while count < nterms:
       print(n1)
       nth = n1 + n2
       # update values
       n1 = n2
       n2 = nth
       count += 1
```

Input/Output:



```
PS F:\python lab(By Dr. Hitesh Kumar Sharma sir )> python -u "f:\python lab(By Dr. Hitesh Kumar Sharma sir )\fibonacci.py"
How many terms? 8
Fibonacci sequence:
0
1
1
2
3
5
8
13
PS F:\python lab(By Dr. Hitesh Kumar Sharma sir )>
```

LAB EXERCISE 7

Aim: To write a Python program for factorial

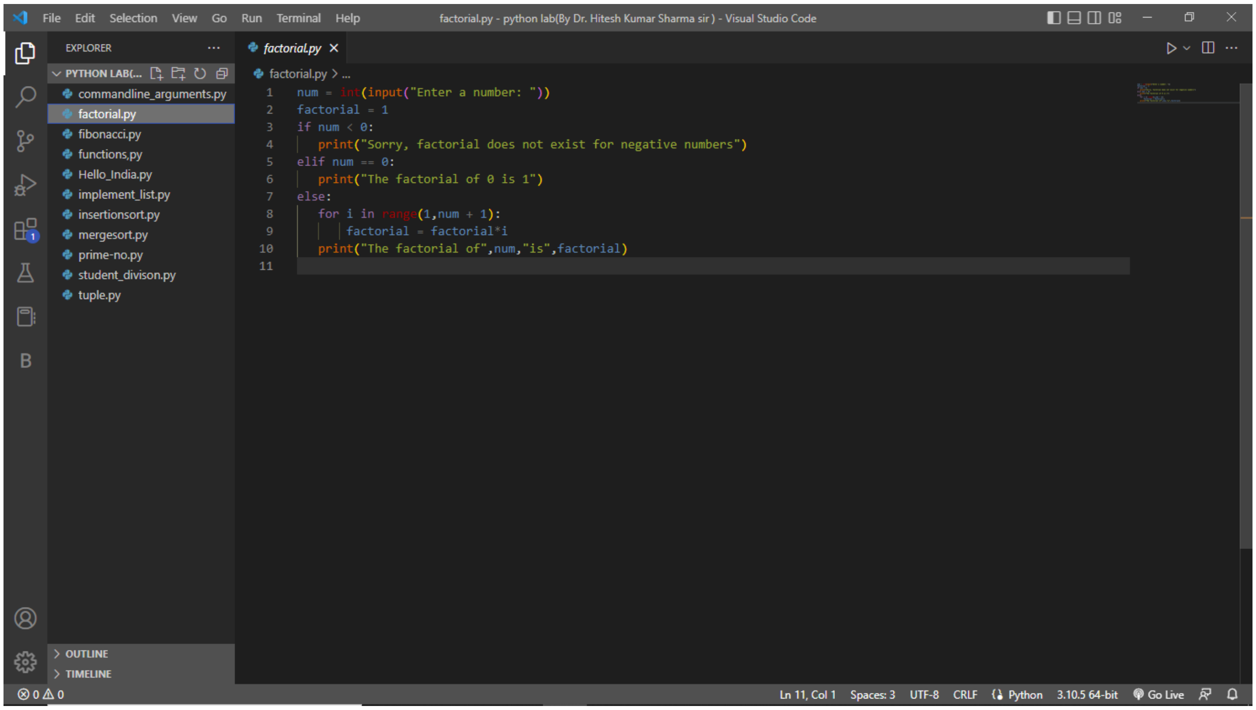
Description:

factorial is a simple thing. Factorials are just products. An exclamation mark indicates the factorial. Factorial is a multiplication operation of natural numbers with all the natural number that are less than it.

n! = n × (n-1) × (n-2) × (n-3) × ....× 3 × 2 × 1

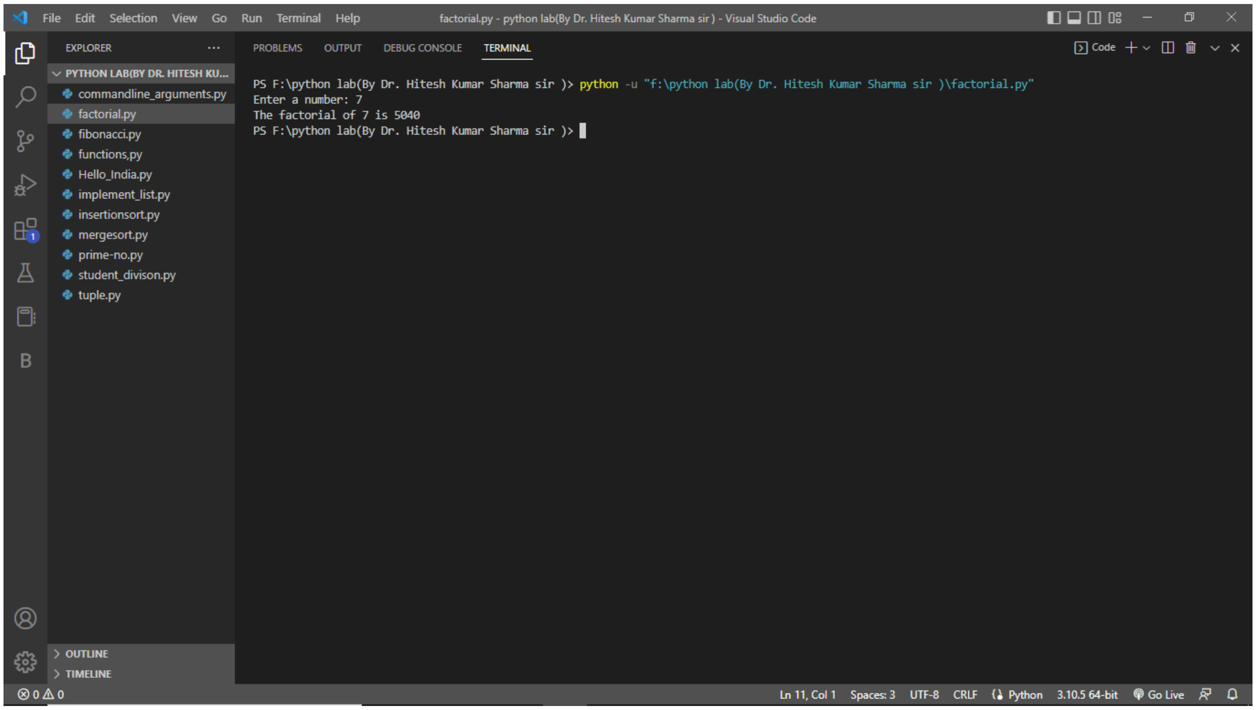
Solution:

Code:



```python
num = int(input("Enter a number: "))
factorial = 1
if num < 0:
   print("Sorry, factorial does not exist for negative numbers")
elif num == 0:
   print("The factorial of 0 is 1")
else:
   for i in range(1,num + 1):
      factorial = factorial*i
   print("The factorial of",num,"is",factorial)

```

Input/Output:



```
PS F:\python lab(By Dr. Hitesh Kumar Sharma sir )> python -u "f:\python lab(By Dr. Hitesh Kumar Sharma sir )\factorial.py"
Enter a number: 7
The factorial of 7 is 5040
PS F:\python lab(By Dr. Hitesh Kumar Sharma sir )> 
```

LAB EXERCISE 8

Aim: To write a Python program to use of functions

Description:

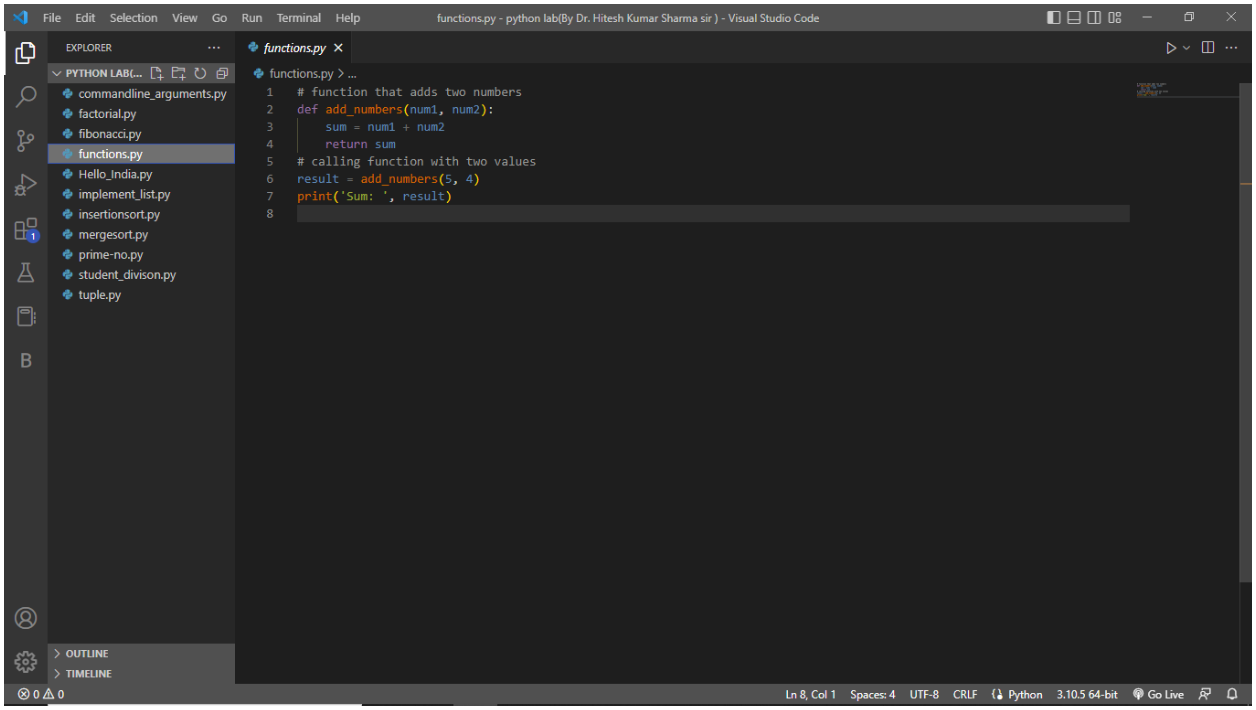
A function is a block of code which only runs when it is called.

You can pass data, known as parameters, into a function.

A function can return data as a result.

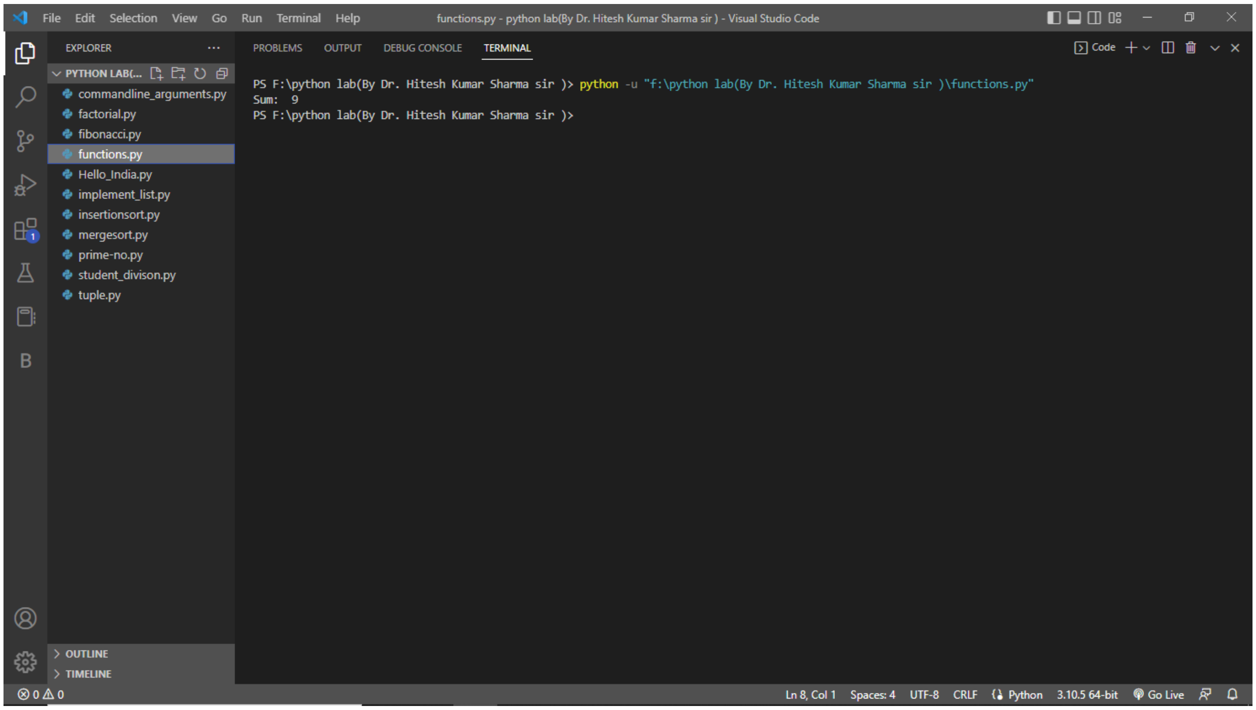
Solution:

Code:



```python
# function that adds two numbers
def add_numbers(num1, num2):
    sum = num1 + num2
    return sum
# calling function with two values
result = add_numbers(5, 4)
print('Sum: ', result)
```

Input/Output:



```
PS F:\python lab(By Dr. Hitesh Kumar Sharma sir )> python -u "f:\python lab(By Dr. Hitesh Kumar Sharma sir )\functions.py"
Sum:  9
PS F:\python lab(By Dr. Hitesh Kumar Sharma sir )>
```

LAB EXERCISE 9

Aim: To write a Python program to implement list

Description:

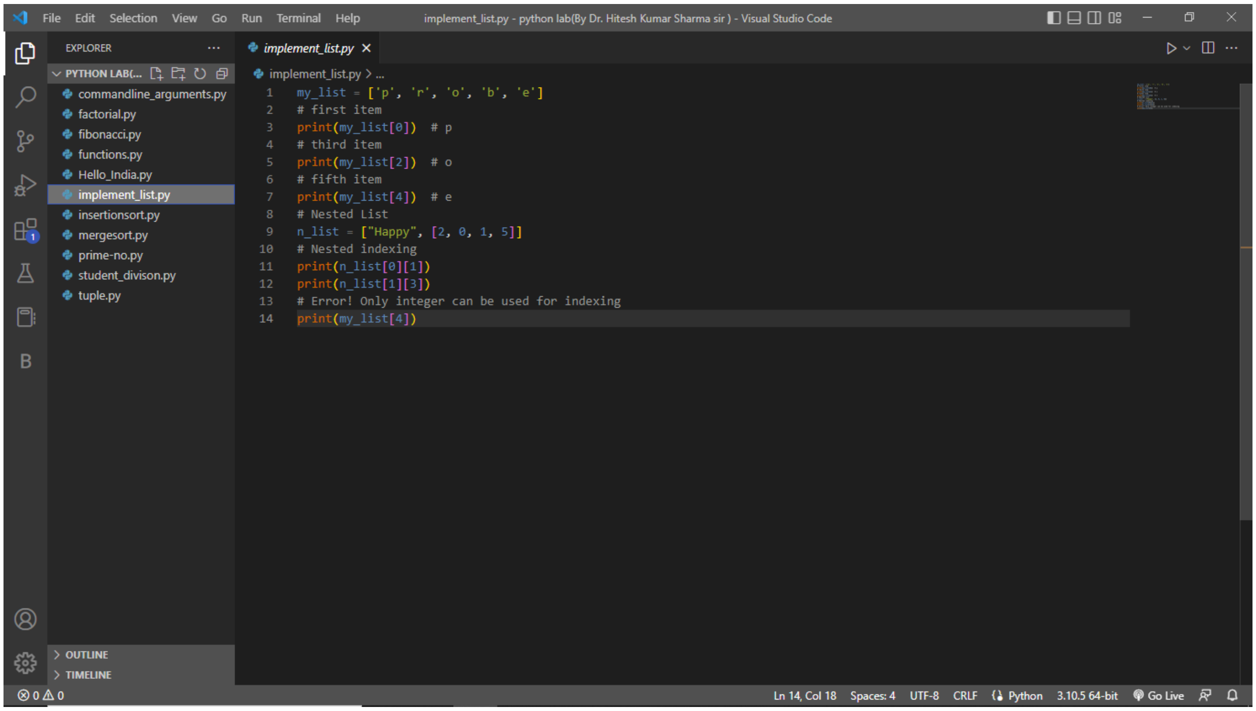
Lists are used to store multiple items in a single variable.

Lists are one of 4 built-in data types in Python used to store collections of data, the other 3 are Tuple, Set, and Dictionary, all with different qualities and usage.

Lists are created using square brackets:

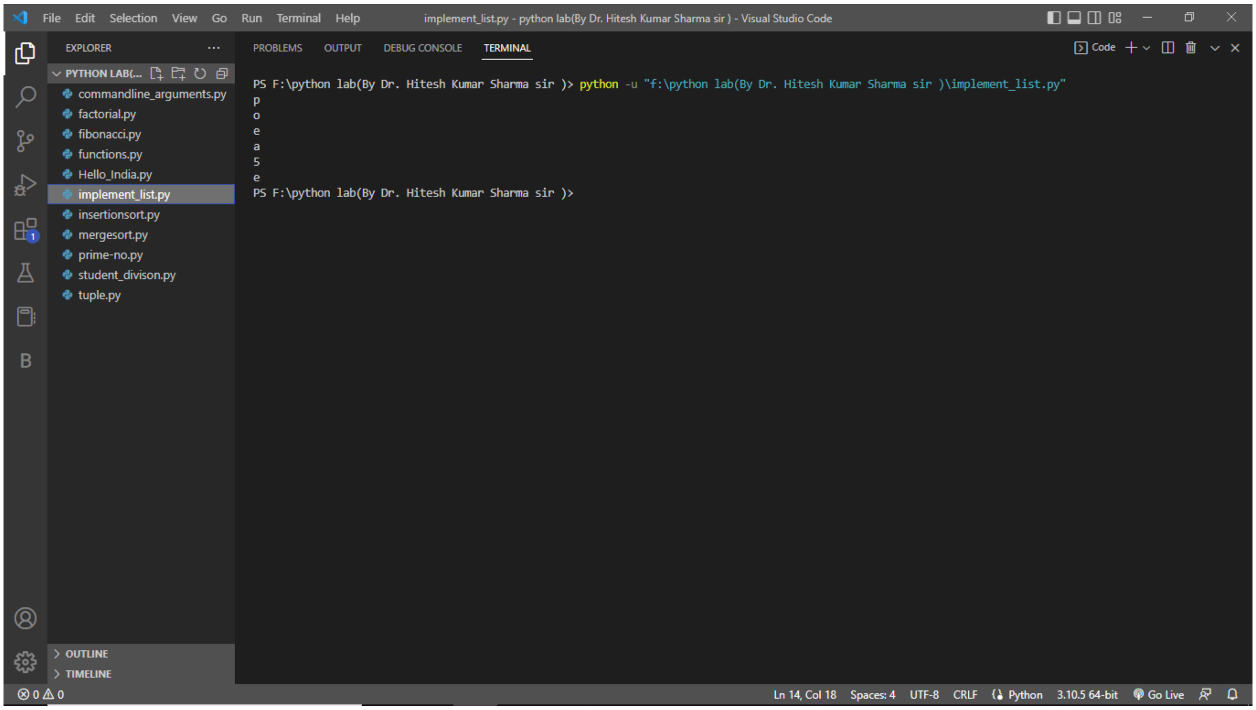
Solution:

Code:



```python
my_list = ['p', 'r', 'o', 'b', 'e']
# first item
print(my_list[0])  # p
# third item
print(my_list[2])  # o
# fifth item
print(my_list[4])  # e
# Nested List
n_list = ["Happy", [2, 0, 1, 5]]
# Nested indexing
print(n_list[0][1])
print(n_list[1][3])
# Error! Only integer can be used for indexing
print(my_list[4])
```

Input/Output:



```
PS F:\python lab(By Dr. Hitesh Kumar Sharma sir )> python -u "f:\python lab(By Dr. Hitesh Kumar Sharma sir )\implement_list.py"
p
o
e
a
5
e
PS F:\python lab(By Dr. Hitesh Kumar Sharma sir )>
```

LAB EXERCISE 10

Aim: To write a Python program to implement tuples.

Description:

Tuples are used to store multiple items in a single variable.

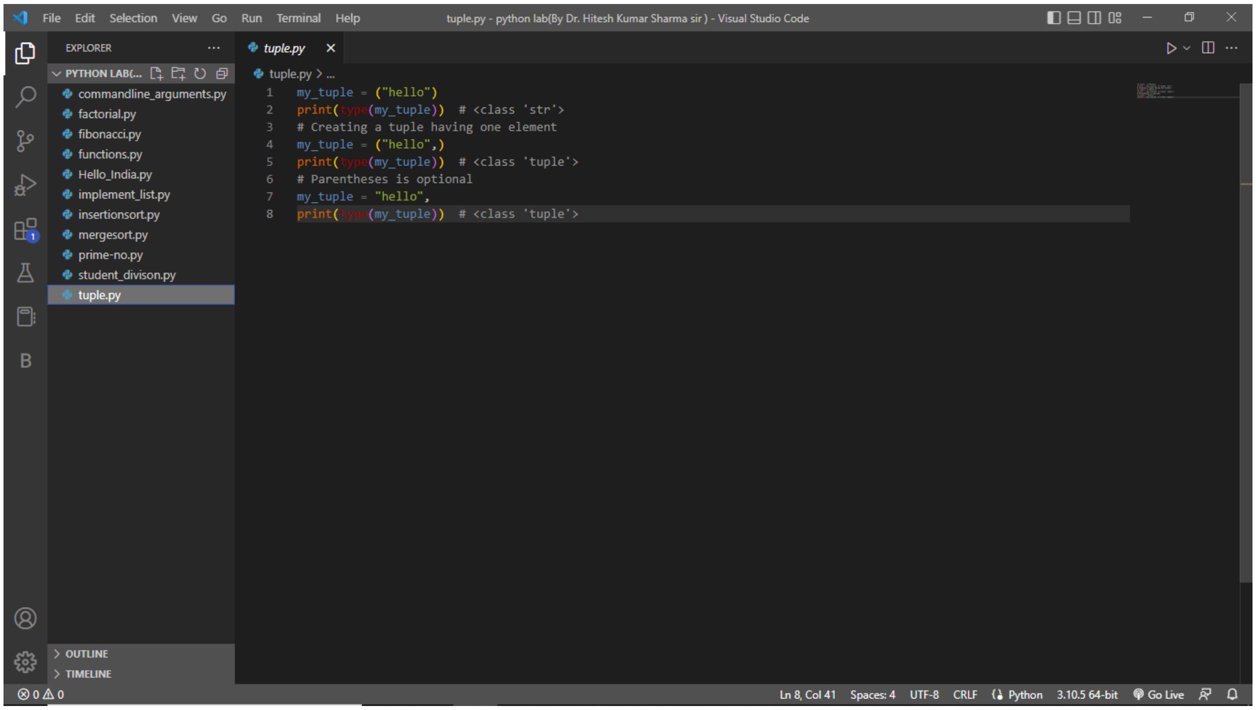
Tuple is one of 4 built-in data types in Python used to store collections of data, the other 3 are List, Set, and Dictionary, all with different qualities and usage.

A tuple is a collection which is ordered and unchangeable**.**

Tuples are written with round brackets.

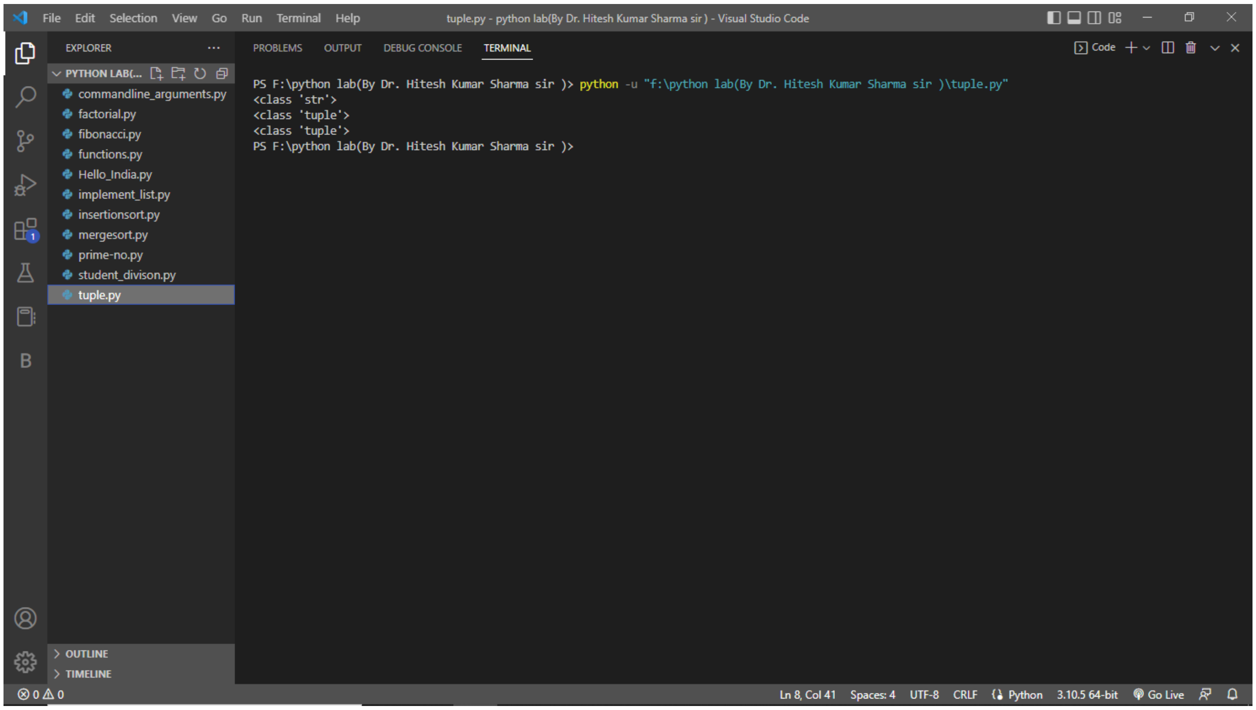
Solution:

Code:



```python
my_tuple = ("hello")
print(type(my_tuple))  # <class 'str'>
# Creating a tuple having one element
my_tuple = ("hello",)
print(type(my_tuple))  # <class 'tuple'>
# Parentheses is optional
my_tuple = "hello",
print(type(my_tuple))  # <class 'tuple'>
```

Input/Output:



File Edit Selection View Go Run Terminal Help
tuple.py - python lab(By Dr. Hitesh Kumar Sharma sir) - Visual Studio Code
EXPLORER
PYTHON LAB(...
commandline_arguments.py
factorial.py
fibonacci.py
functions.py
Hello_India.py
implement_list.py
insertionsort.py
mergesort.py
prime-no.py
student_divison.py
tuple.py
OUTLINE
TIMELINE
PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL
Code
Ln 8, Col 41 Spaces: 4 UTF-8 CRLF Python 3.10.5 64-bit Go Live

```
PS F:\python lab(By Dr. Hitesh Kumar Sharma sir )> python -u "f:\python lab(By Dr. Hitesh Kumar Sharma sir )\tuple.py"
<class 'str'>
<class 'tuple'>
<class 'tuple'>
PS F:\python lab(By Dr. Hitesh Kumar Sharma sir )>
```

LAB EXERCISE 11

Aim: To write a Python program Insertion sort.

Description:

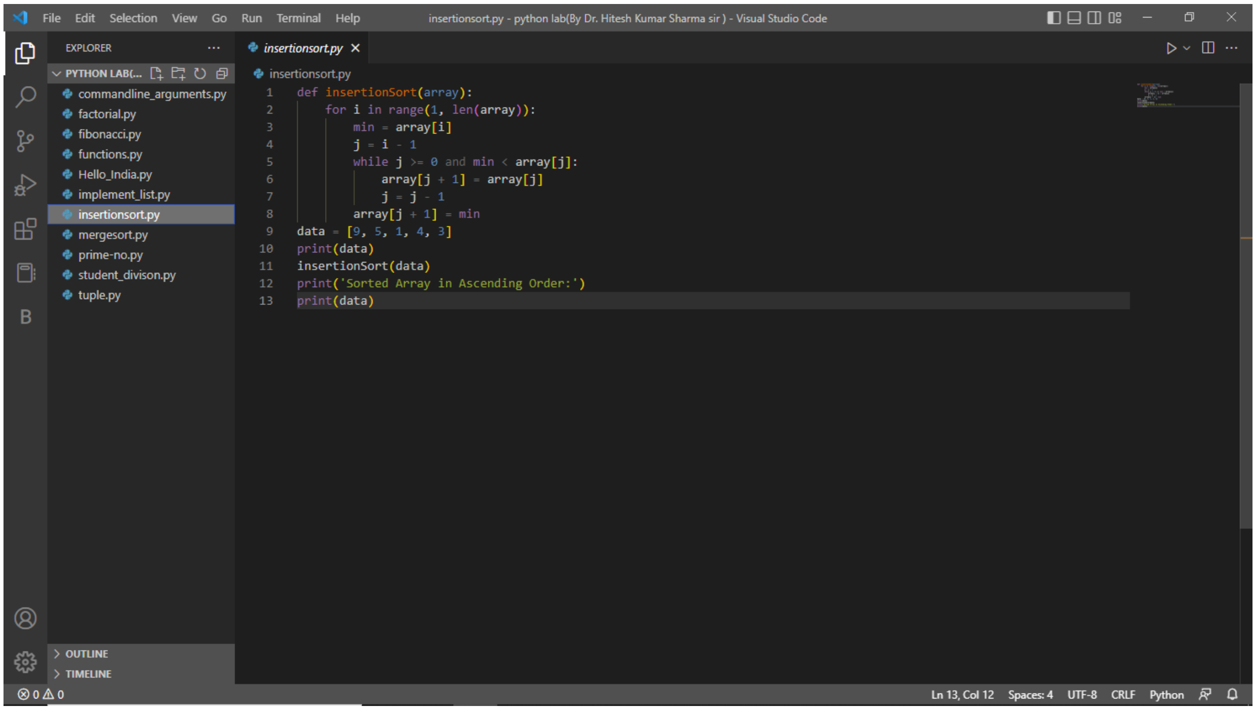
Insertion sort is the simple method of sorting an array. In this technique, the array is virtually split into the sorted and unsorted part. An element from unsorted part is picked and is placed at correct position in the sorted part.

The array elements are traversed from 1 to n.

If the array element at position i is greater than its predecessor, it does not need to be moved.

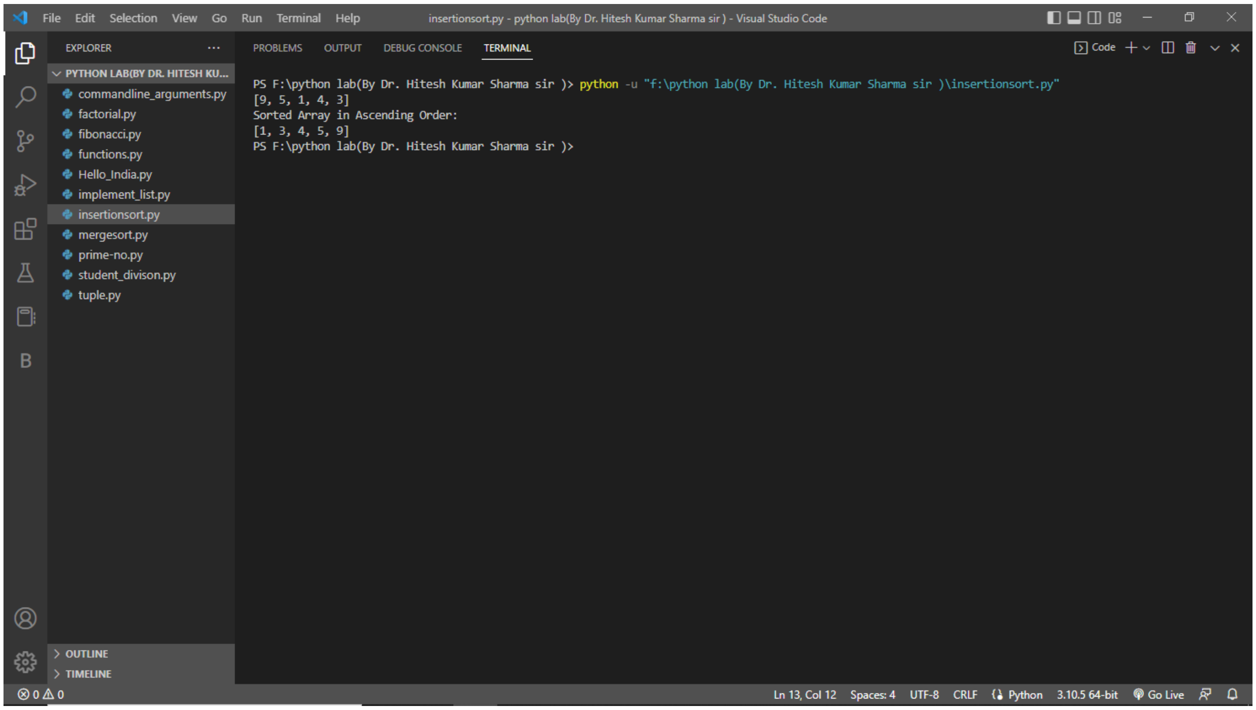
Solution:

Code:



```python
def insertionSort(array):
    for i in range(1, len(array)):
        min = array[i]
        j = i - 1
        while j >= 0 and min < array[j]:
            array[j + 1] = array[j]
            j = j - 1
        array[j + 1] = min
data = [9, 5, 1, 4, 3]
print(data)
insertionSort(data)
print('Sorted Array in Ascending Order:')
print(data)
```

Input/Output:



```
PS F:\python lab(By Dr. Hitesh Kumar Sharma sir )> python -u "f:\python lab(By Dr. Hitesh Kumar Sharma sir )\insertionsort.py"
[9, 5, 1, 4, 3]
Sorted Array in Ascending Order:
[1, 3, 4, 5, 9]
PS F:\python lab(By Dr. Hitesh Kumar Sharma sir )>
```

LAB EXERCISE 12

Aim: To write a Python program merge sort.

Description:

Merge Sort is a Divide and Conquer algorithm. It divides input array in two halves, calls itself for the two halves and then merges the two sorted halves. The merge() function is used for merging two halves.

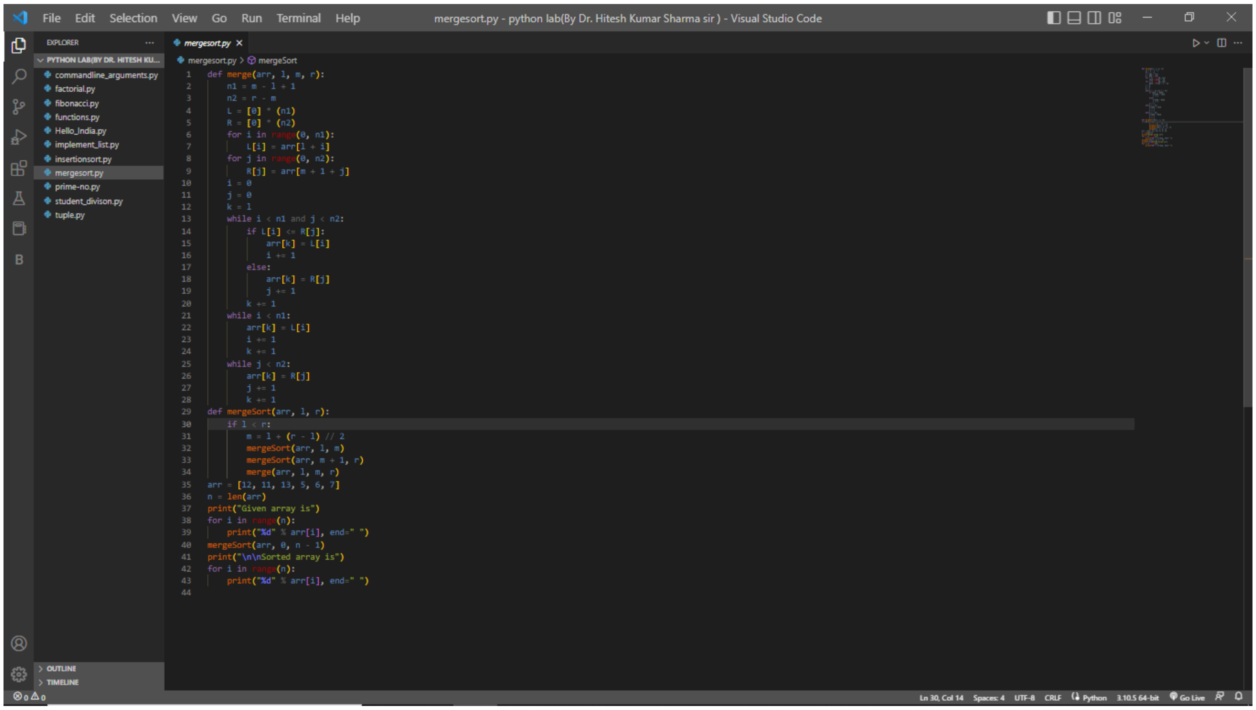
The merge(arr, l, m, r) is key process that assumes that arr[l..m] and arr[m+1..r] are sorted and merges the two sorted sub-arrays into one.

The sub lists are divided again and again into halves until we get the only one element each. Then we combine the pair of one element lists into two element lists, sorting them in the process.

The sorted two element pairs is merged into the four element lists, and so on until we get the sorted list.

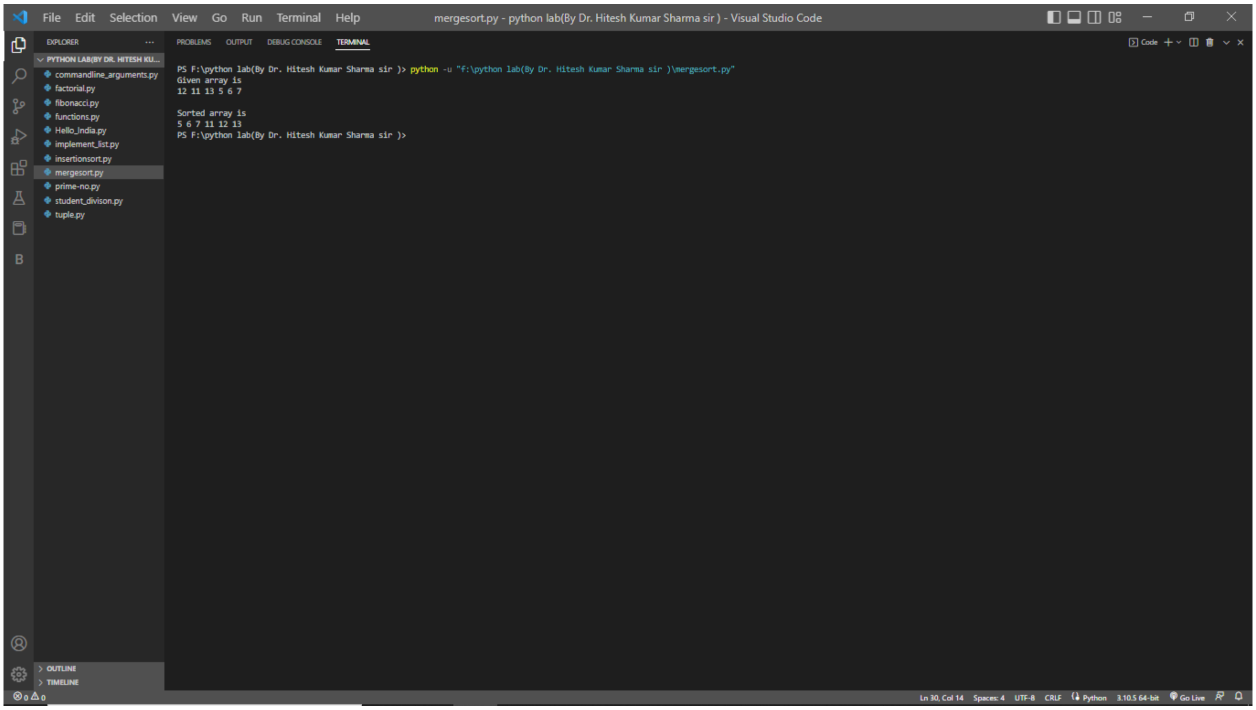
Solution:

Code:



```python
def merge(arr, l, m, r):
    n1 = m - l + 1
    n2 = r - m
    L = [0] * (n1)
    R = [0] * (n2)
    for i in range(0, n1):
        L[i] = arr[l + i]
    for j in range(0, n2):
        R[j] = arr[m + 1 + j]
    i = 0
    j = 0
    k = l
    while i < n1 and j < n2:
        if L[i] <= R[j]:
            arr[k] = L[i]
            i += 1
        else:
            arr[k] = R[j]
            j += 1
        k += 1
    while i < n1:
        arr[k] = L[i]
        i += 1
        k += 1
    while j < n2:
        arr[k] = R[j]
        j += 1
        k += 1
def mergeSort(arr, l, r):
    if l < r:
        m = l + (r - l) // 2
        mergeSort(arr, l, m)
        mergeSort(arr, m + 1, r)
        merge(arr, l, m, r)
arr = [12, 11, 13, 5, 6, 7]
n = len(arr)
print("Given array is")
for i in range(n):
    print("%d" % arr[i], end=" ")
mergeSort(arr, 0, n - 1)
print("\n\nSorted array is")
for i in range(n):
    print("%d" % arr[i], end=" ")
```

Input/Output:



```
PS F:\python lab(By Dr. Hitesh Kumar Sharma sir )> python -u "f:\python lab(By Dr. Hitesh Kumar Sharma sir )\mergesort.py"
Given array is
12 11 13 5 6 7

Sorted array is
5 6 7 11 12 13
PS F:\python lab(By Dr. Hitesh Kumar Sharma sir )>
```

LAB EXERCISE 13

Aim: To write a Python program first n prime numbers

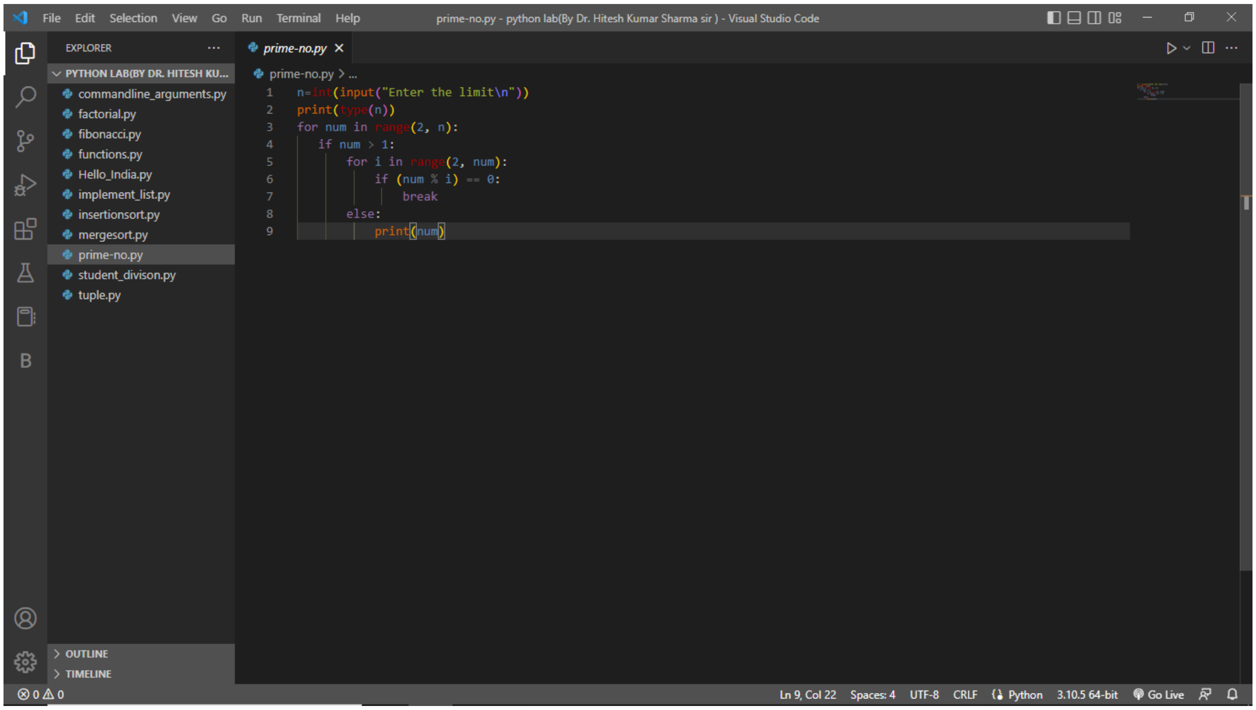
Description:

A prime number is a natural number which is greater than 1 and has no positive divisor other than 1 and itself, such as 2, 3, 5, 7, 11, 13, and so on.

The user is given two integer numbers, lower value, and upper value. The task is to write the Python program for printing all the prime numbers between the given interval (or range).

Solution:

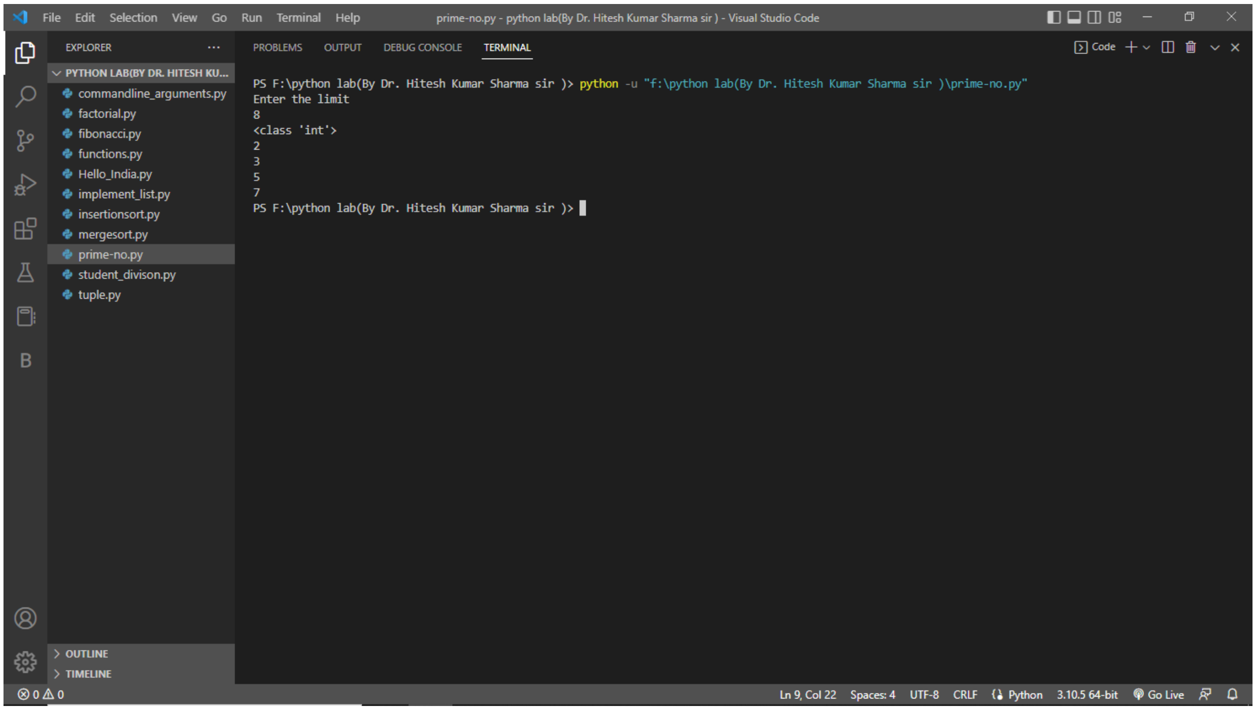
Code:



File Edit Selection View Go Run Terminal Help
prime-no.py - python lab(By Dr. Hitesh Kumar Sharma sir) - Visual Studio Code
EXPLORER
PYTHON LAB(BY DR. HITESH KU...
commandline_arguments.py
factorial.py
fibonacci.py
functions.py
Hello_India.py
implement_list.py
insertionsort.py
mergesort.py
prime-no.py
student_divison.py
tuple.py
OUTLINE
TIMELINE
prime-no.py
prime-no.py > ...
Ln 9, Col 22 Spaces: 4 UTF-8 CRLF Python 3.10.5 64-bit Go Live

```python
n=int(input("Enter the limit\n"))
print(type(n))
for num in range(2, n):
   if num > 1:
       for i in range(2, num):
           if (num % i) == 0:
               break
       else:
           print(num)
```

Input/Output:



File Edit Selection View Go Run Terminal Help
prime-no.py - python lab(By Dr. Hitesh Kumar Sharma sir) - Visual Studio Code
EXPLORER
PYTHON LAB(BY DR. HITESH KU...
commandline_arguments.py
factorial.py
fibonacci.py
functions.py
Hello_India.py
implement_list.py
insertionsort.py
mergesort.py
prime-no.py
student_divison.py
tuple.py
PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL
OUTLINE
TIMELINE
Ln 9, Col 22 Spaces: 4 UTF-8 CRLF Python 3.10.5 64-bit Go Live

```
PS F:\python lab(By Dr. Hitesh Kumar Sharma sir )> python -u "f:\python lab(By Dr. Hitesh Kumar Sharma sir )\prime-no.py"
Enter the limit
8
<class 'int'>
2
3
5
7
PS F:\python lab(By Dr. Hitesh Kumar Sharma sir )>
```

LAB EXERCISE 14

Aim: Implementation of Data Science concepts using Python

Description:

This Data Science with Python program provides learners with **a** complete understanding of data analytics tools & techniques. Getting started with Python can help you gain knowledge on data analysis, visualization, NumPy, SciPy, web scraping, and natural language processing**.**

Python offers a good number of libraries used in data science such as Pandas, Numpy, and Scikit-learn. Learning those libraries right away and skipping the basics isn't good though.

If you would like to learn Python for data science, you should master Python core concepts first. Having a solid foundation in Python will help you avoid common mistakes and bad practices. As a result, learning Python libraries used in data science will be much easier.

In this guide, we'll see some must-know Python concepts every data scientist should know. At the end of this article, you will find a Python for Data Science Cheat Sheet in PDF version (section 9 in the table of contents below

1. Python Attributes vs Methods

I can't tell how long I used the words "attribute" and "method" interchangeably when I was a beginner in Python.

When you learn libraries like Pandas, you'll frequently call attributes and methods, so it's good to know what's the difference between them.

- Attribute: An attribute is a variable stored in a class. That is, a value
- Method: A method is a function that is defined inside a class body.