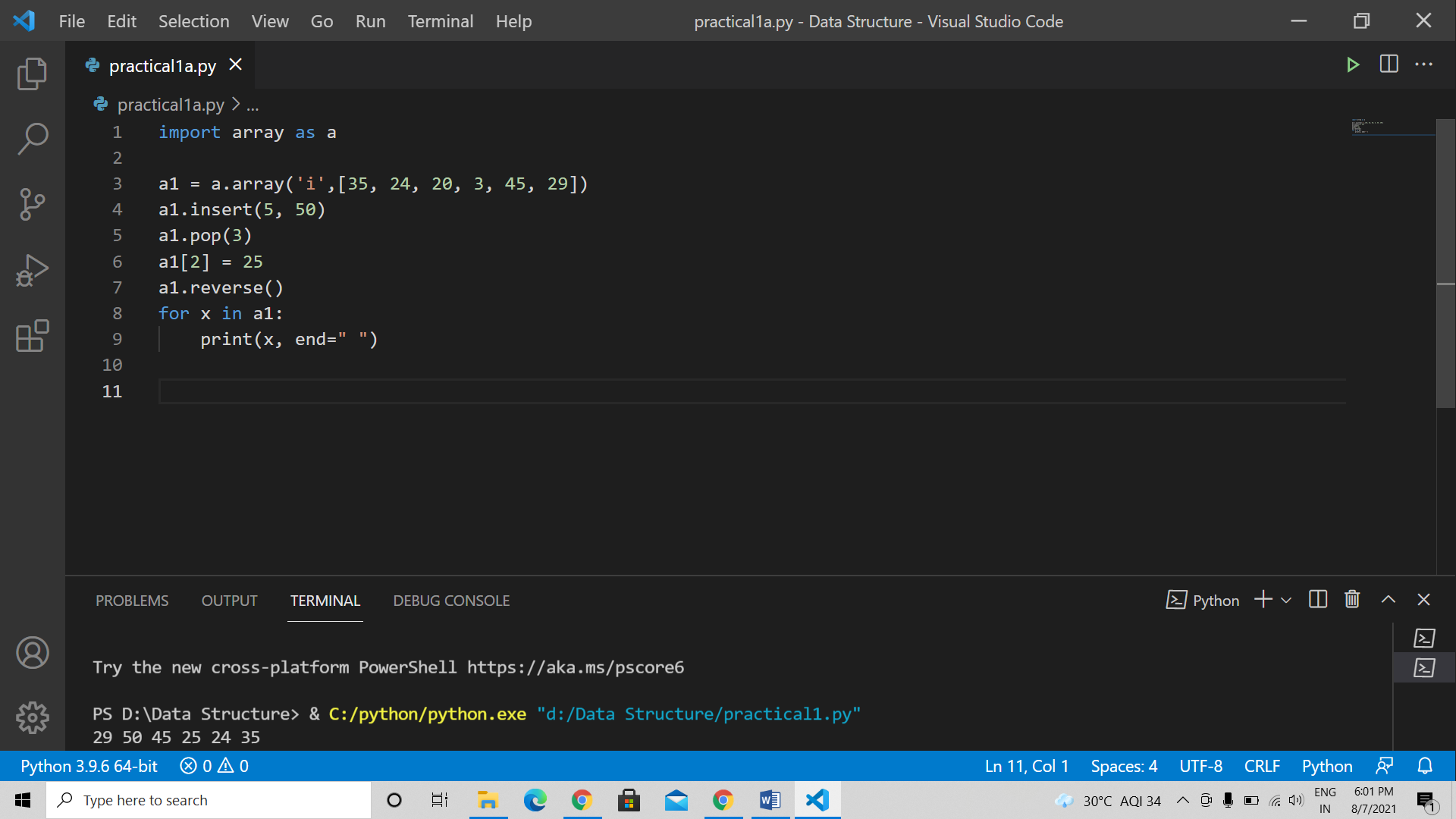
DS Practical

1. Implement the following for array.
2. Aim : write a program to store the elements in 1-D array and provide an option to perform the operations like searching, sorting, merging, reversing the elements.

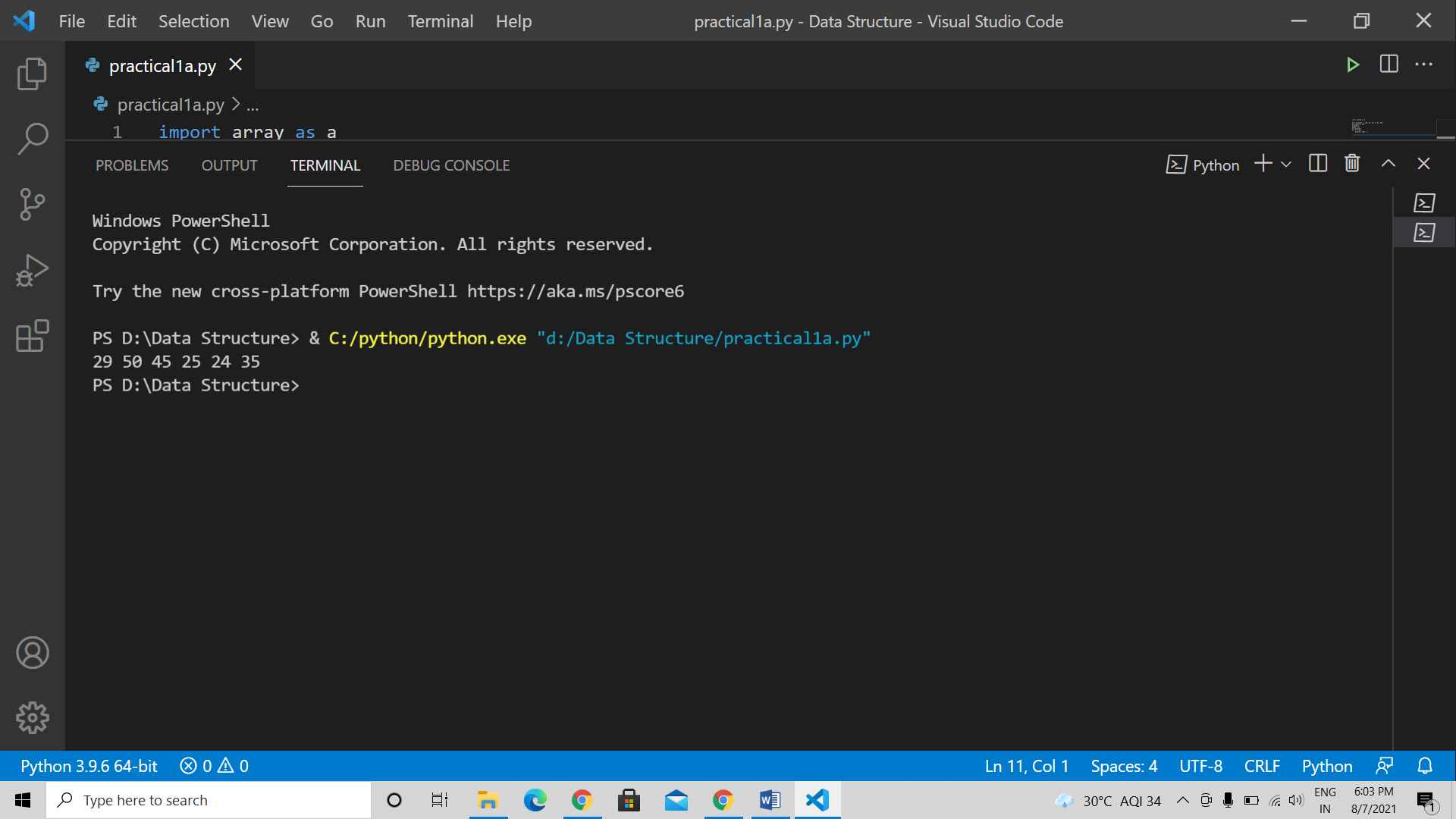
Theory :

1. Array : collection of similar type of data items stored at contiguous memory locations.
2. Element − Each item stored in an array is called an element.
3. Index − Each location of an element in an array has a numerical index, which is used to identify the element. Index starts with 0.
4. Insertion : Adds an element at the given index.
5. Pop or remove : Deletes an element at the given index.
6. Update : Updates an element at the given index.
7. Reverse : It reverses an array at its original location.

Code :



Output :

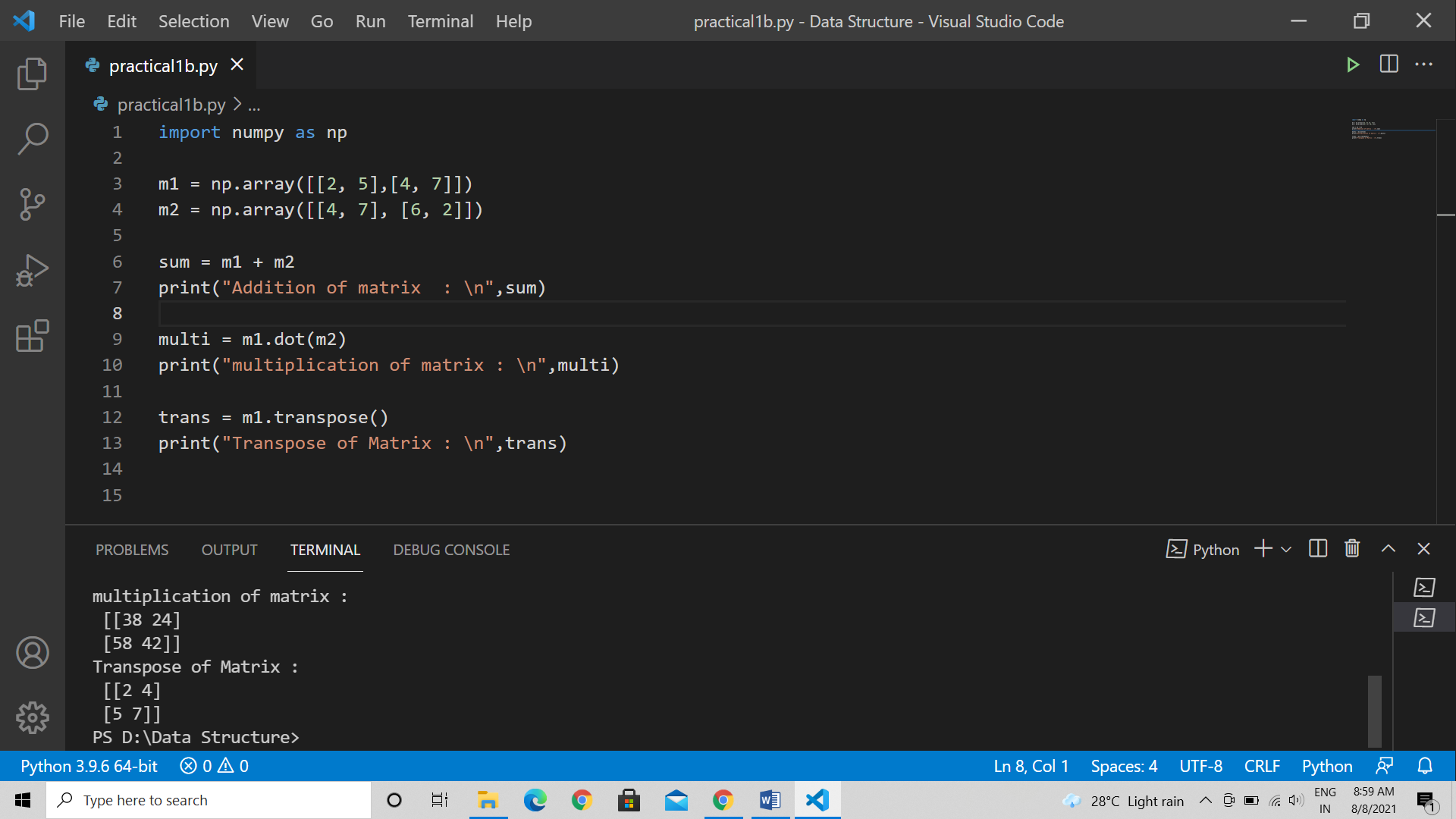


1. Aim : Write a program to the Matrix, addition, Multiplication and Transpose operations.

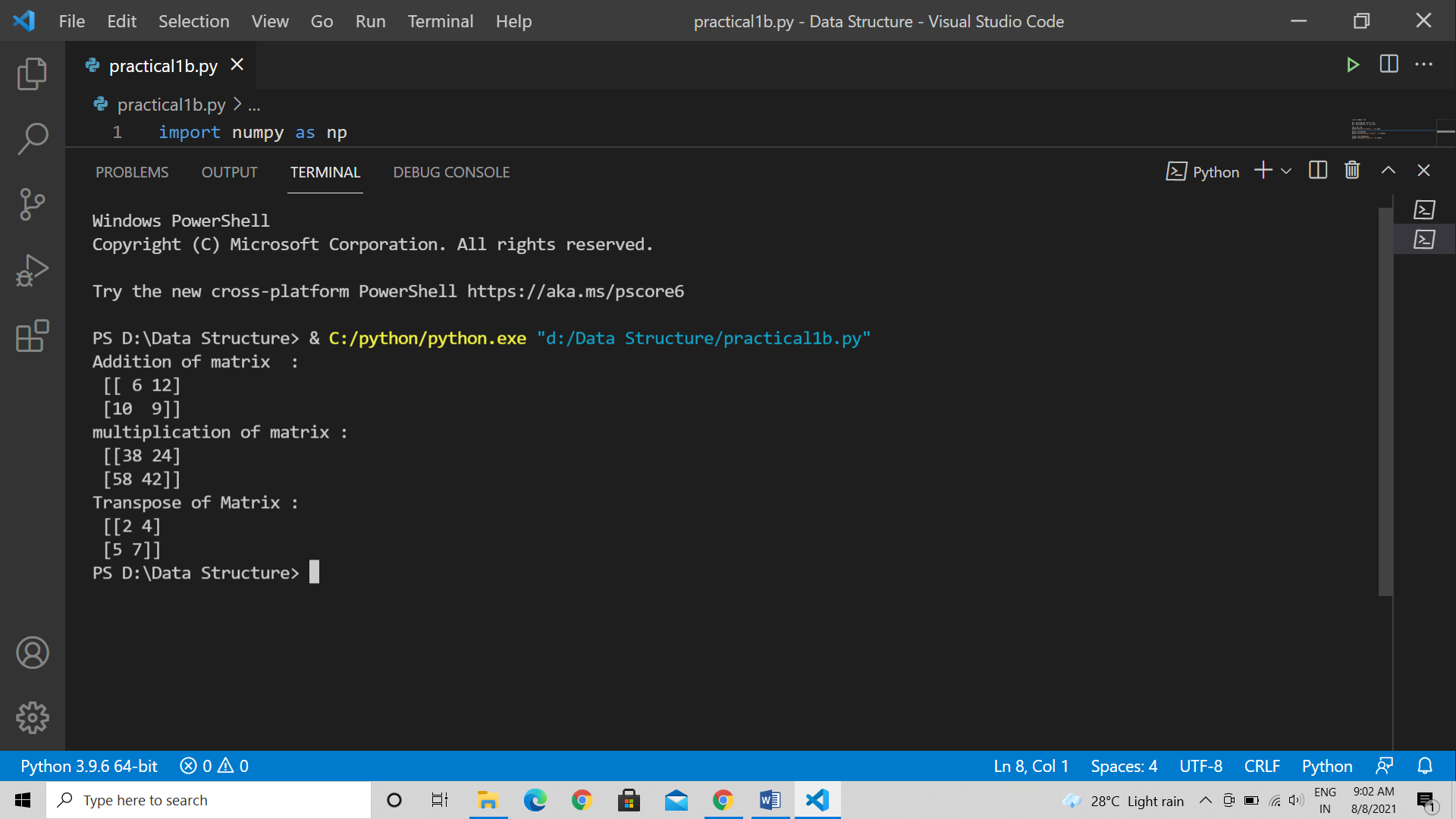
Theory :

1. Matrix Addition : To perform addition on the matrix, we will create two matrices using numpy.array() and add them using the (+) operator.
2. Matrix Multiplication : we can make use of numpy dot() method. Numpy.dot() is the dot product of matrix M1 and M2. Numpy.dot() handles the 2D arrays and perform matrix multiplications.
3. Matrix Transpose : The transpose of a matrix is calculated, by changing the rows as columns and columns as rows. The transpose() function from Numpy can be used to calculate the transpose of a matrix.

Code :



Output :

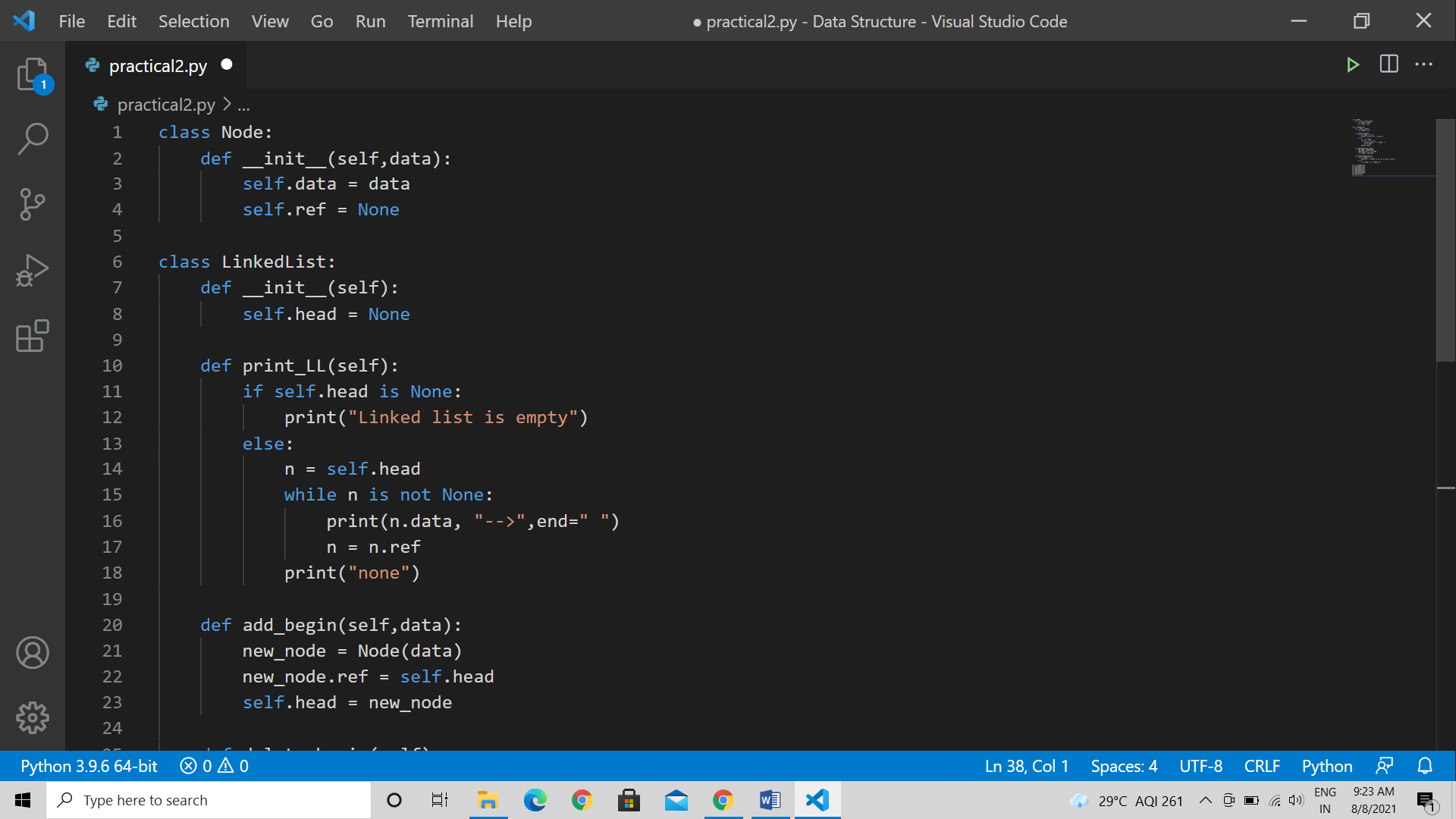


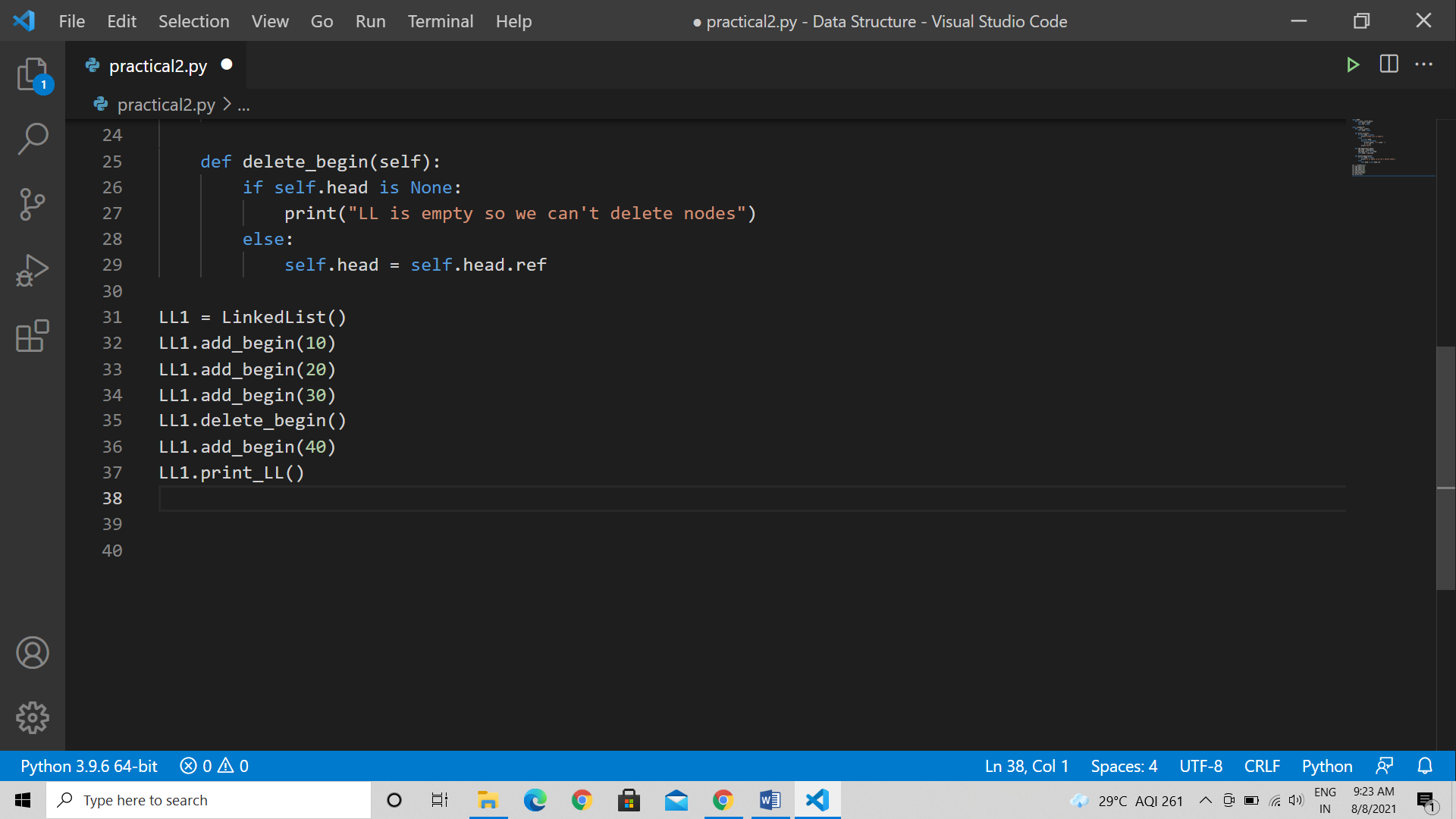
1. Aim : Implement Linked List. Include options for insertion, deletion and search of a number, reverse the list and concatenate two linked list.

Theory :

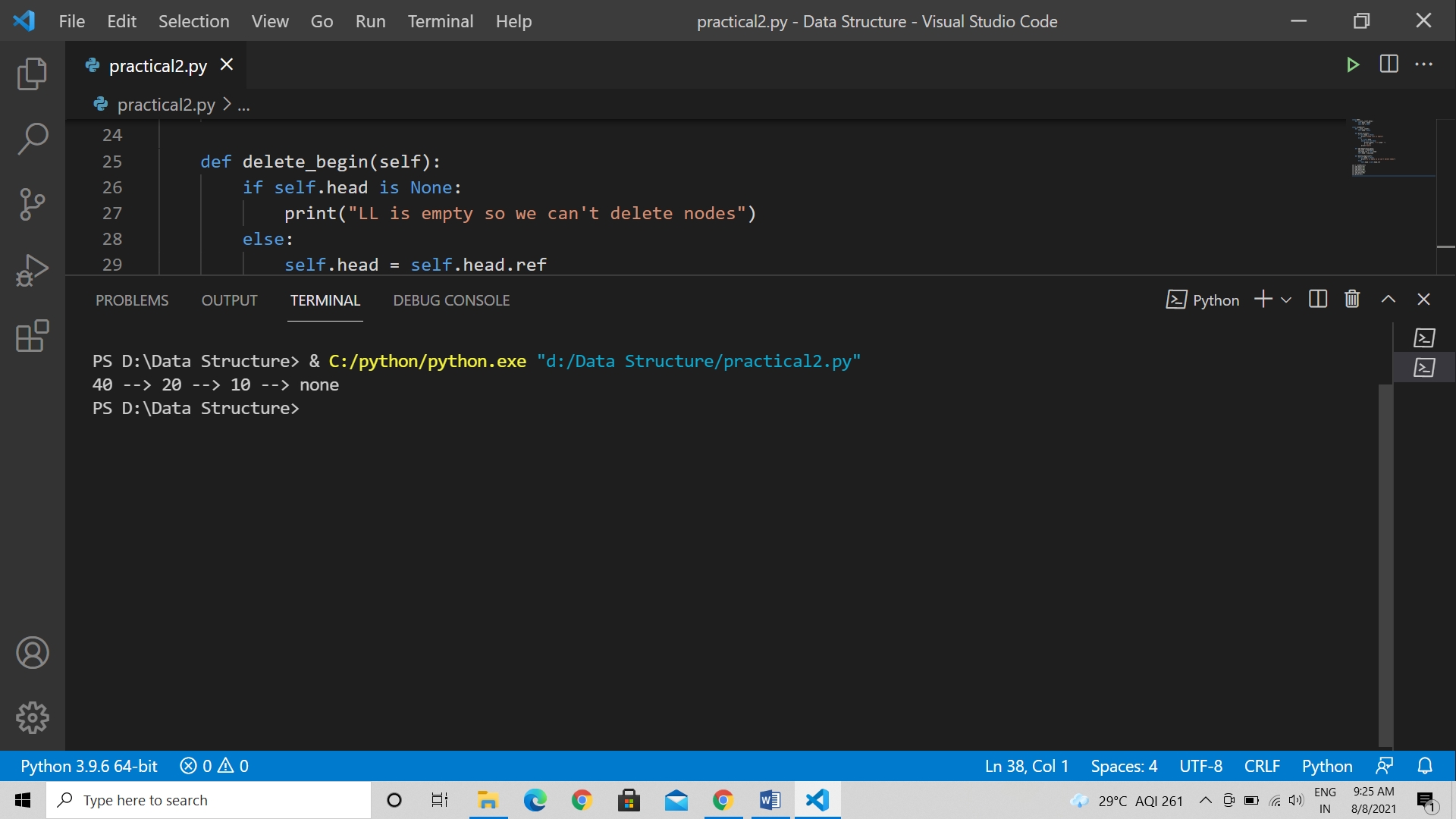
1. A linked list is a linear data structure that includes a series of connected nodes. Here, each node store the data and the address of the next node.
2. we give the address of the first node a special name called HEAD. Also, the last node in the linked list can be identified because its next portion points to NULL.
3. Insertion at the beginning : Since there is no need to find the end of the list. If the list is empty, we make the new node as the head of the list. Otherwise, we have to connect the newnode to the current head of the list and make the new node, the head of the list.
4. Deletion at the beginning : Find the previous node of the node to be deleted. Change the next of the previous node. Free memory for the node to be deleted.

Code :





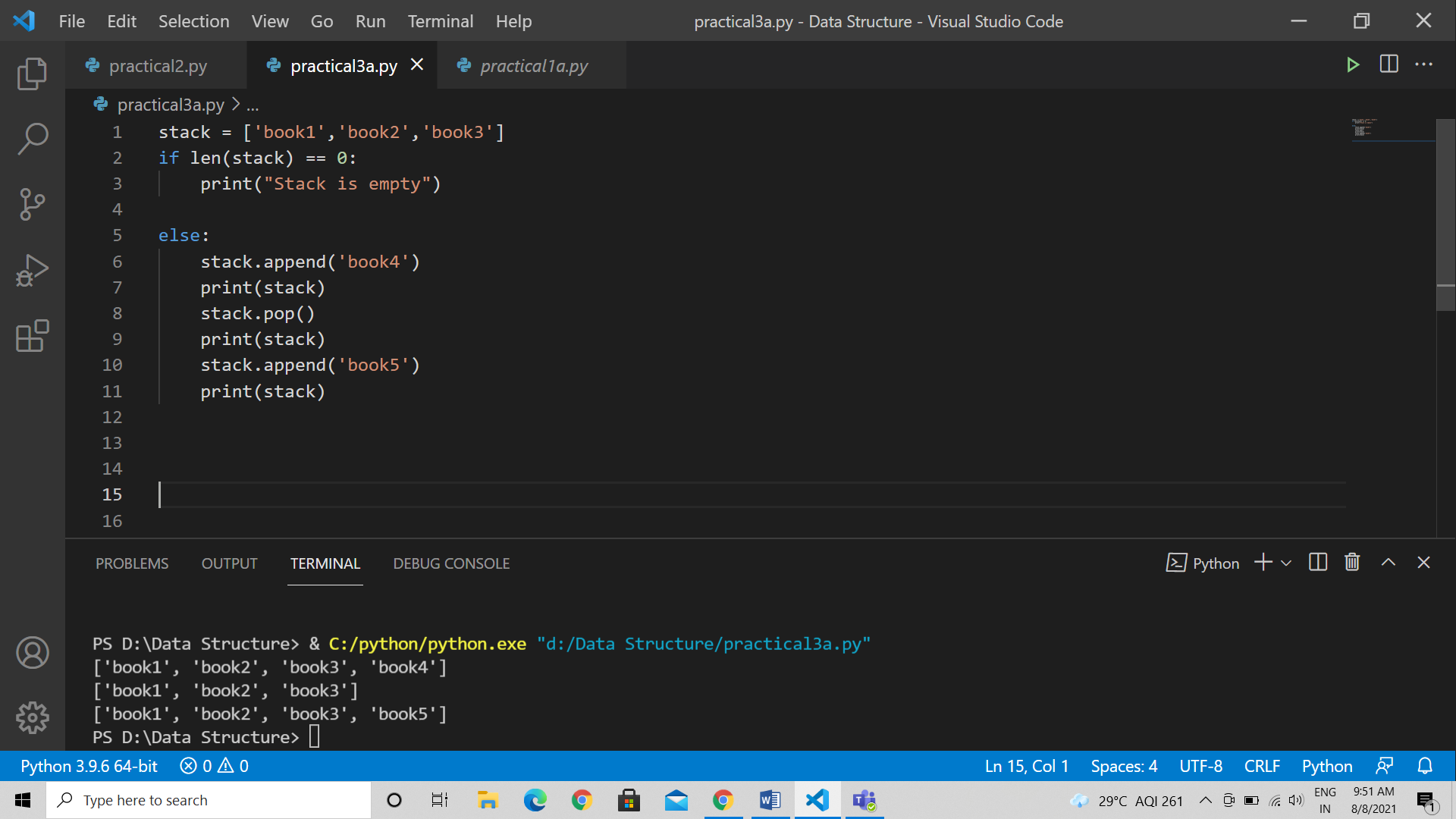
Output :



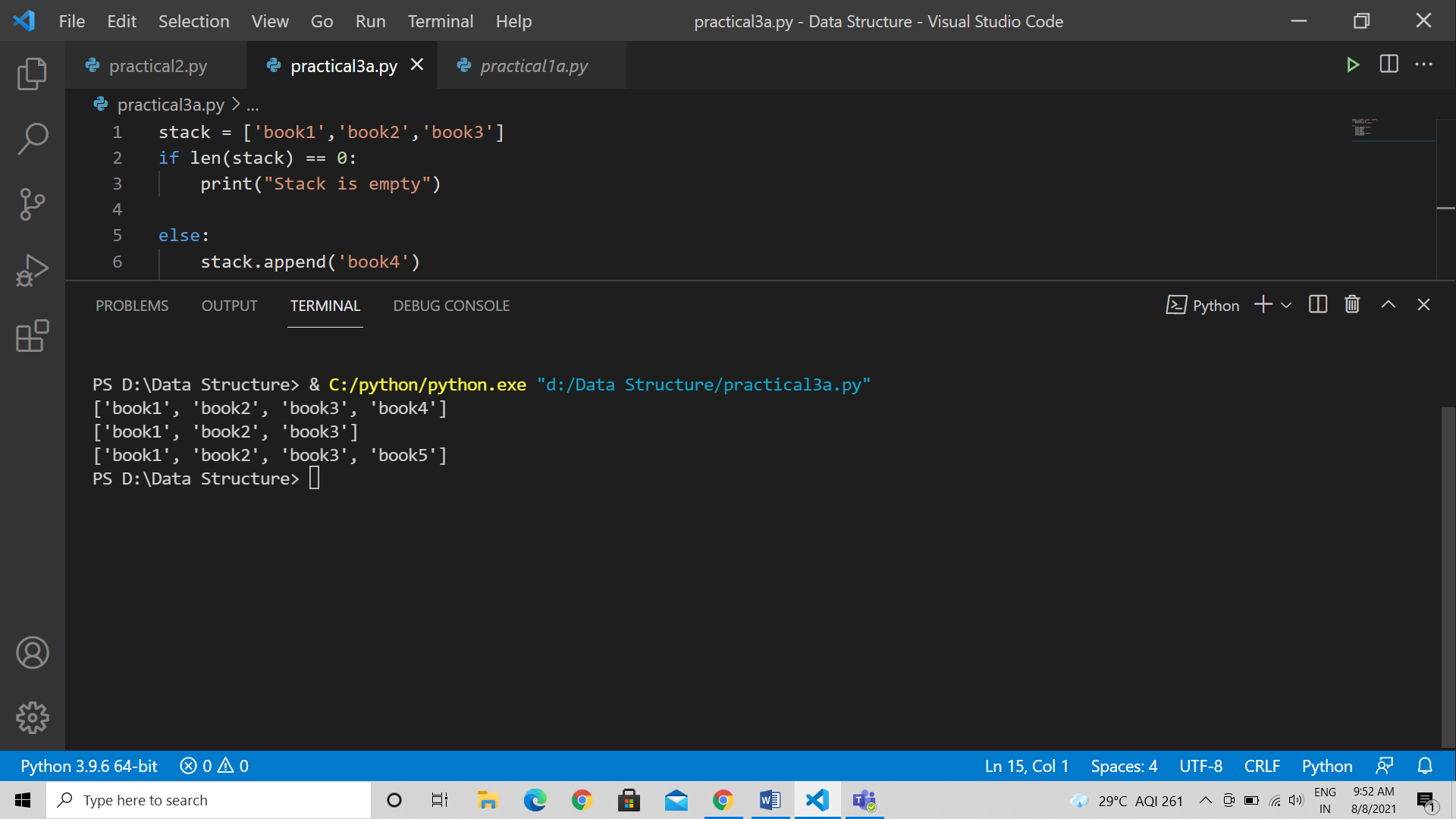
1. Implement the following for stack
2. Aim : Perform stack operation using array implementation.

Theory :

Code :



Output :

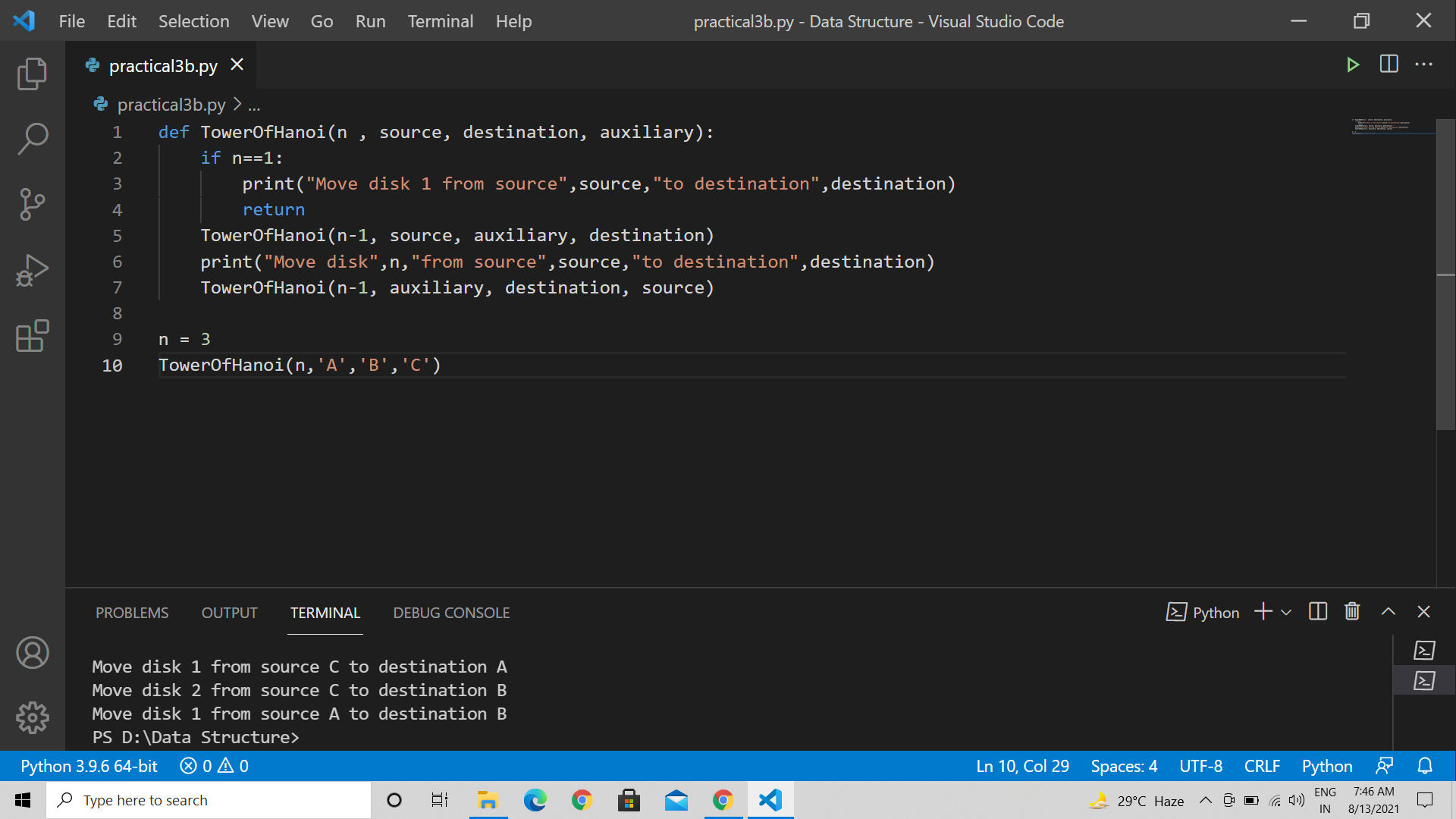


1. Aim : Implement Tower of Hanoi

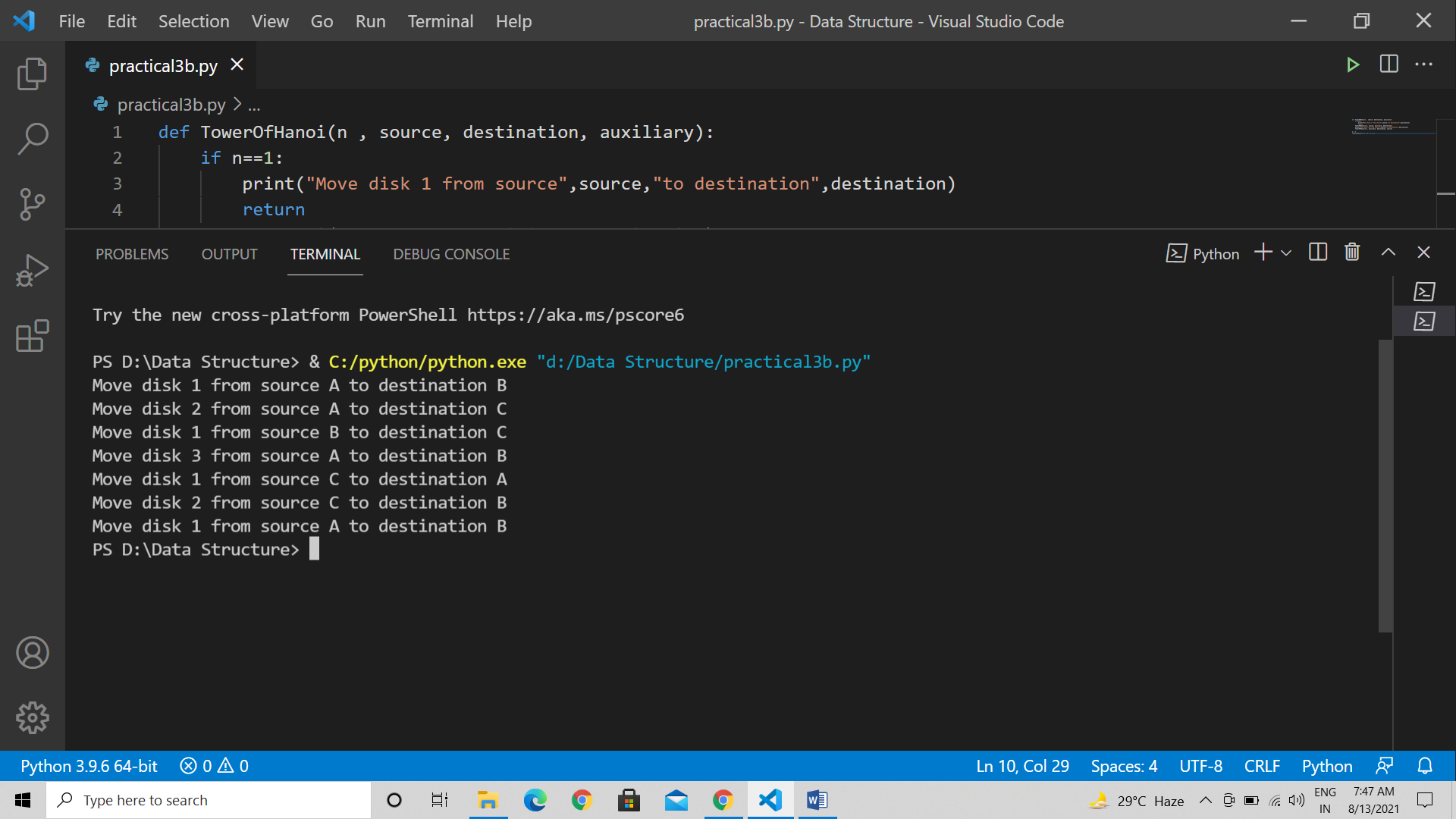
Theory :

1. Create a tower\_of\_hanoi recursive and pass two arguments : the number of disks n and the na,e of the rods such as source, aux, and target.
2. We can define the base case when the number of disks is 1. In this case simply move the one disk from the source to target and return.
3. Now, move remaining n-1 disks from source to auxiliary using the target as the auxiliary.
4. Then, the remaining 1 disk move on the source to target.
5. Move the n-1 disks on the auxiliary to the target using the source as the auxiliarly

Code :

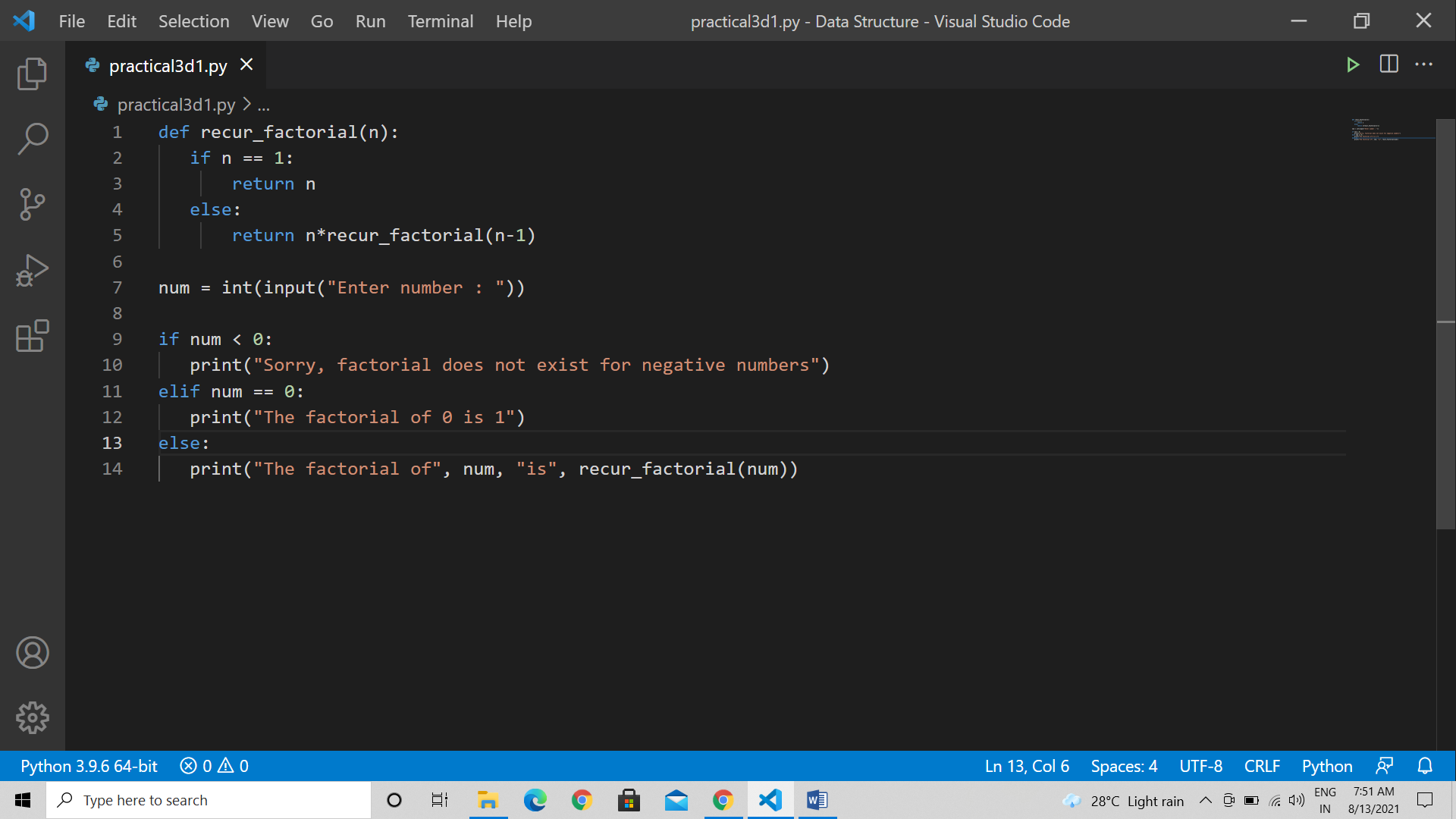


Output :

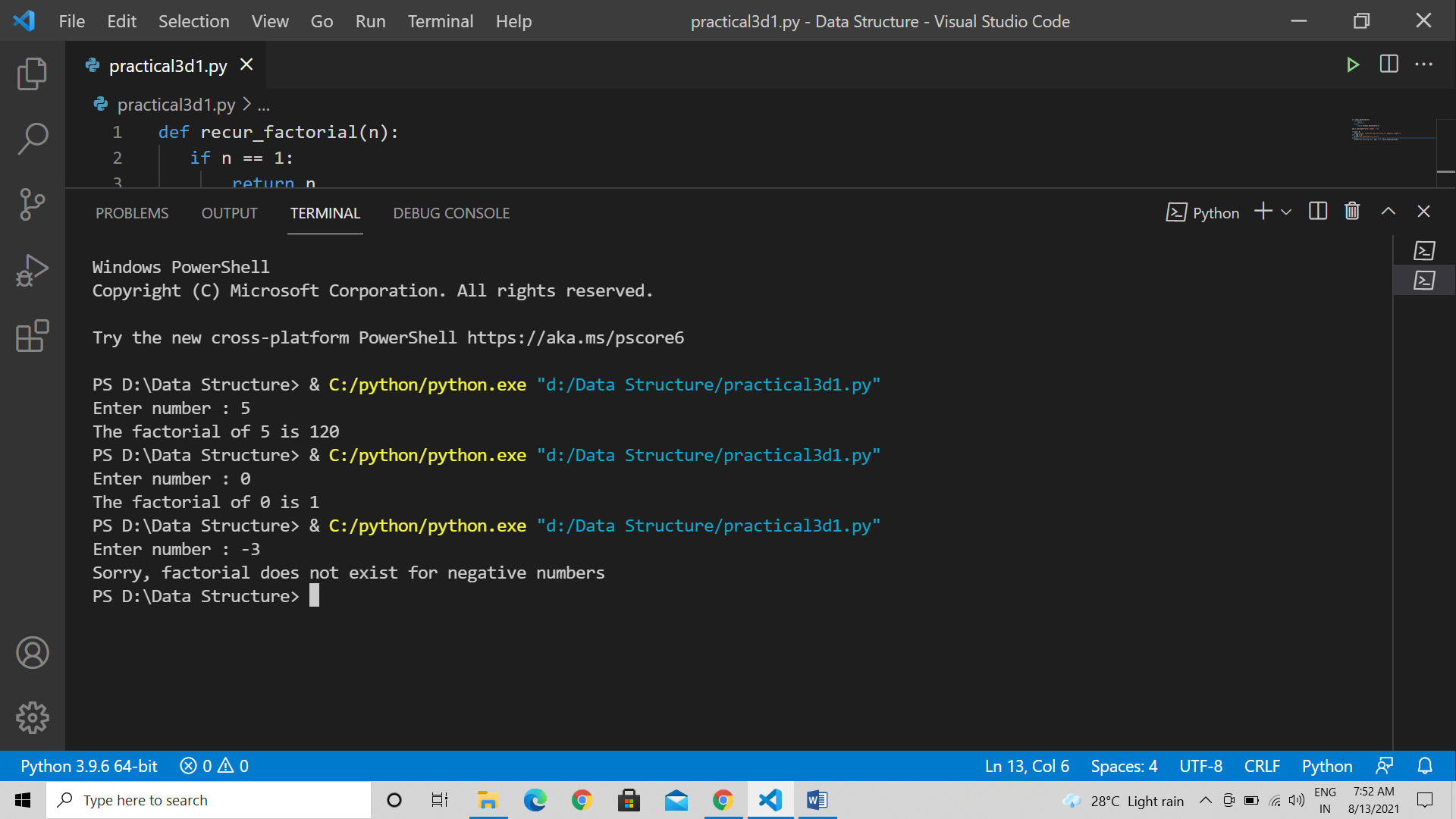


1. WAP to calculate factorial and to compute the factors of a given no.
2. using recursion

code :

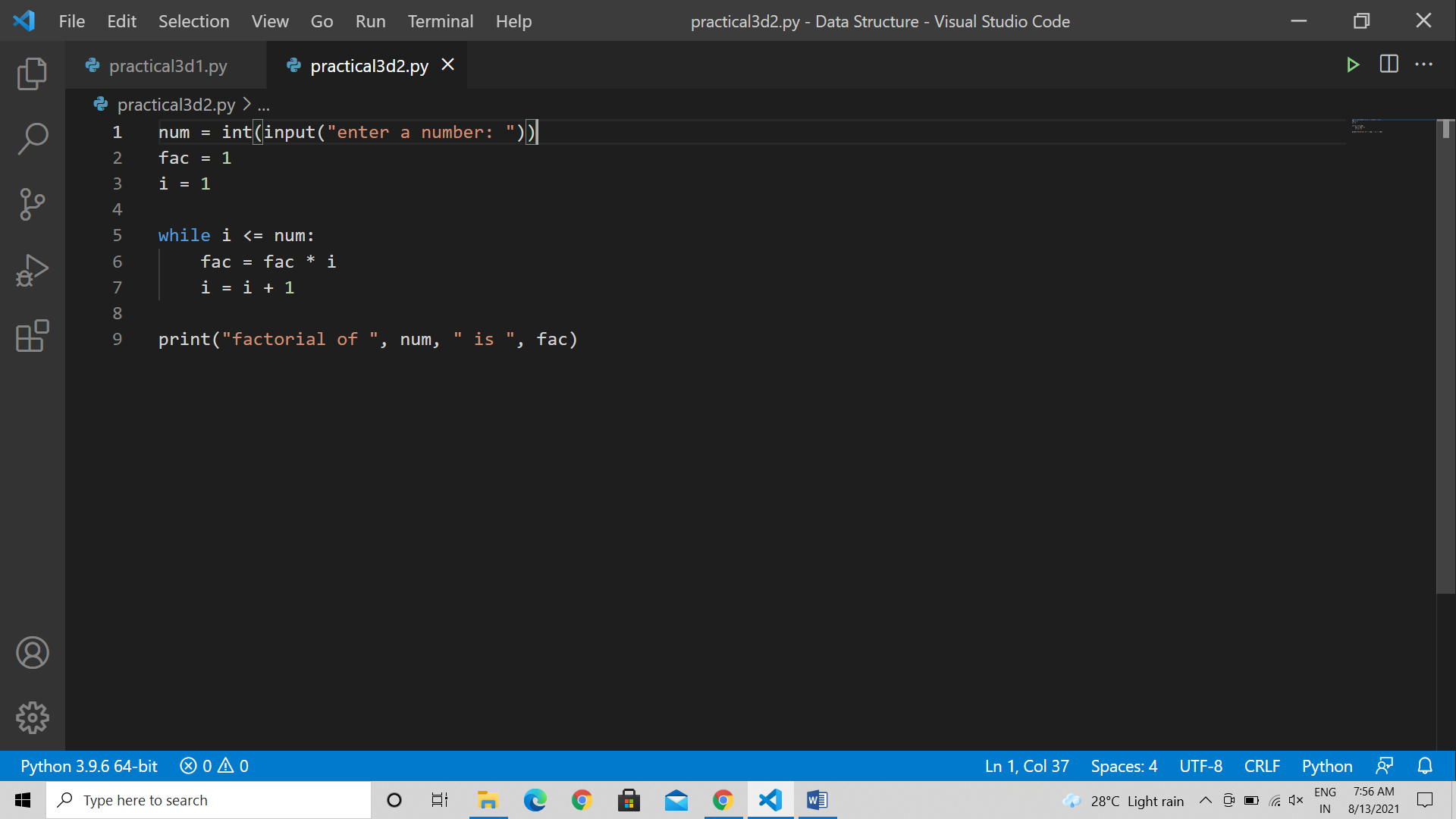


Output :



1. using iteration.

Code :



Output :

