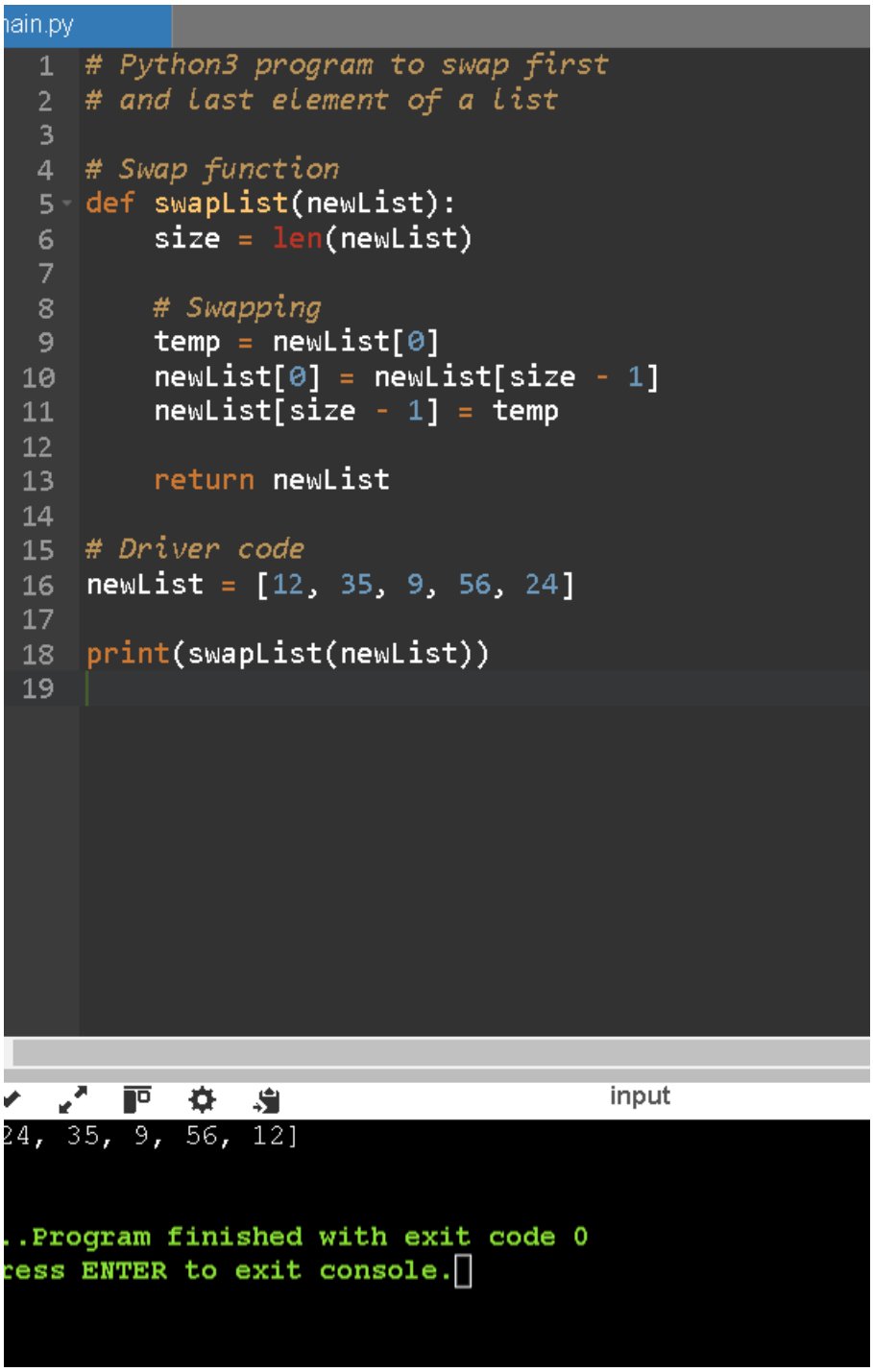
Assignment 1

Saveetha school of engineering

1.write a python program to interchange first and last elements in a list



2.Python program to swap 2 elements in a list

```
# at given positions

# Swap function
def swapPositions(list, pos1, pos2):

    list[pos1], list[pos2] = list[pos2], list[pos1]
    return list

# Driver function
List = [23, 65, 19, 90]
pos1, pos2  = 1, 3

print(swapPositions(List, pos1-1, pos2-1))
```

Output:

```
[19, 65, 23, 90]
```

3. Reverse words in a given string in python

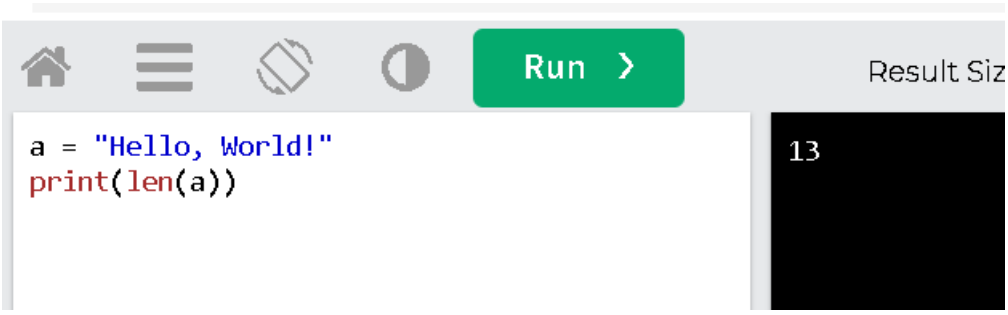
Python3

```
# Python code
# To reverse words in a given string

# input string
string = "geeks quiz practice code"
# reversing words in a given string
s = string.split()[::-1]
l = []
for i in s:
    # appending reversed words to l
    l.append(i)
# printing reverse words
print(" ".join(l))
```

Output

```
code practice quiz geeks
```

4. Find the length of a string in python



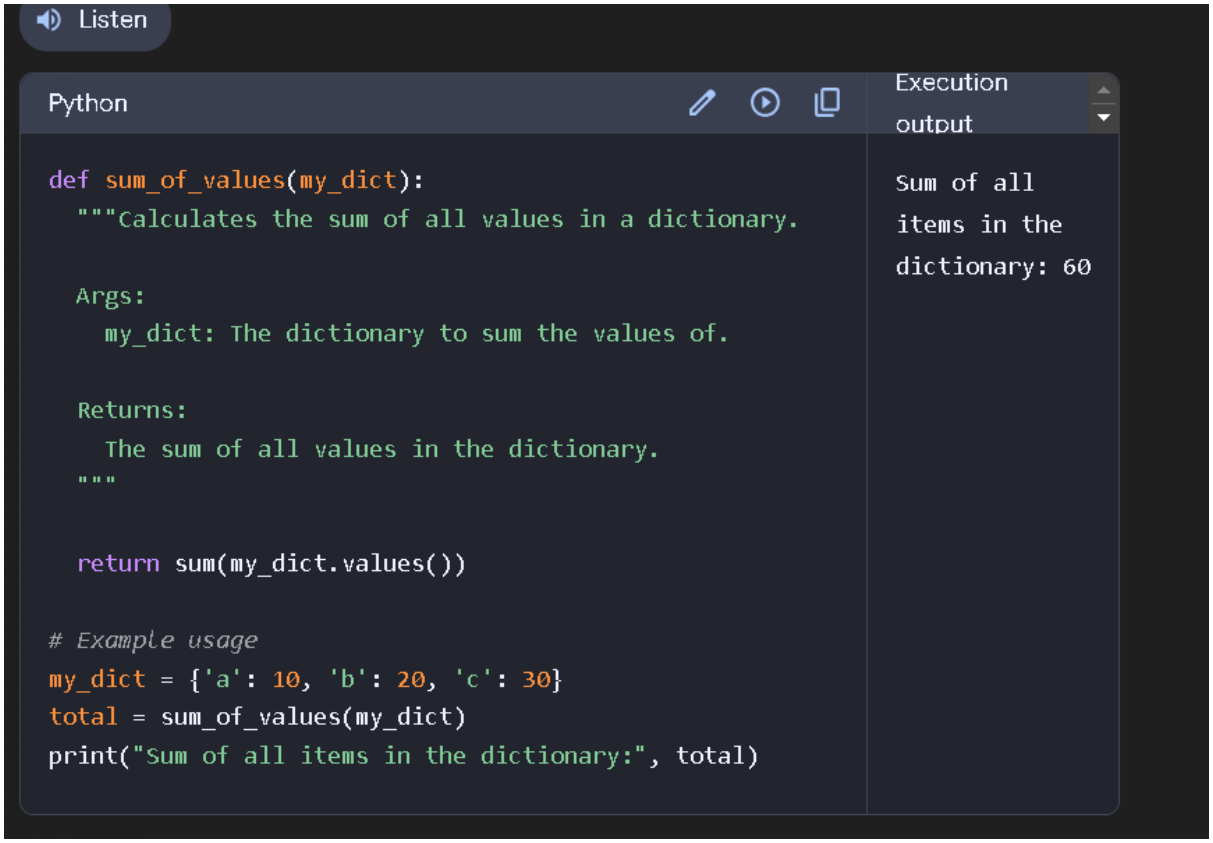
5. Python program to find the maximum and minimum k elements in a tuple

```python
# initializing tuple
test_tup = (5, 20, 3, 7, 6, 8)

# printing original tuple
print("The original tuple is : " + str(test_tup))

# initializing K
K = 2

# Maximum and Minimum K elements in Tuple
# Using sorted() + loop
res = []
test_tup = list(sorted(test_tup))

for idx, val in enumerate(test_tup):
    if idx < K or idx >= len(test_tup) - K:
        res.append(val)
res = tuple(res)

# printing result
print("The extracted values : " + str(res))
```

Output

```
The original tuple is : (5, 20, 3, 7, 6, 8)
The extracted values : (3, 5, 8, 20)
```

6. Python program to find sum of all item in a dictionary python



7. Python program to find sum of all item in a dictionary python

Python3

```
def Remove(initial_set):
    while initial_set:
        initial_set.pop()
        print(initial_set)


initial_set = set([12, 10, 13, 15, 8, 9])
Remove(initial_set)
```

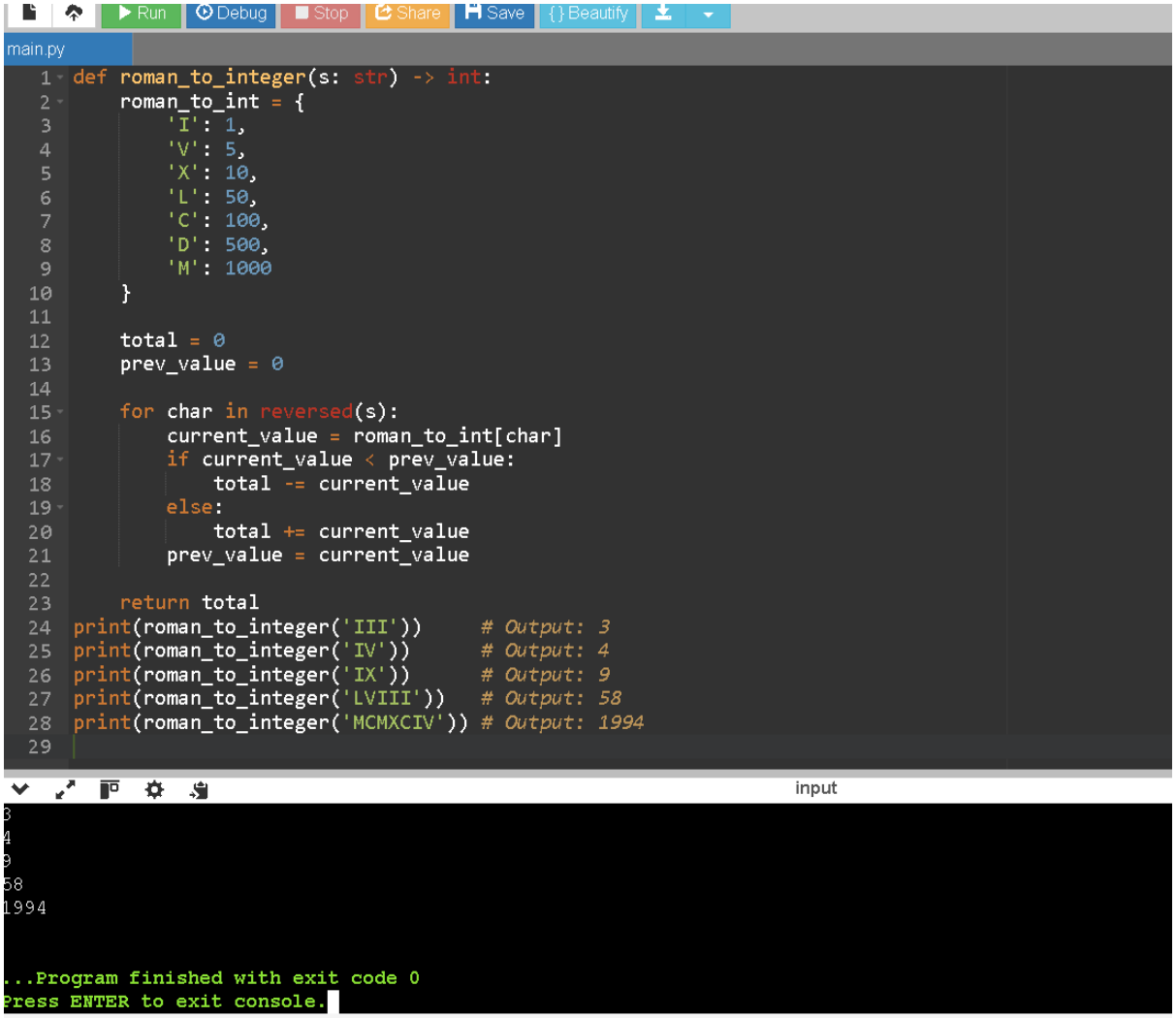
Output:

```
{9, 10, 12, 13, 15}
{10, 12, 13, 15}
{12, 13, 15}
{13, 15}
{15}
set()
```

ASSIGNMENT 2

SAVEETHA SCHOOL OF ENGINEERING

1.Converting Roman Numbers to integers



2.Bit Reserving

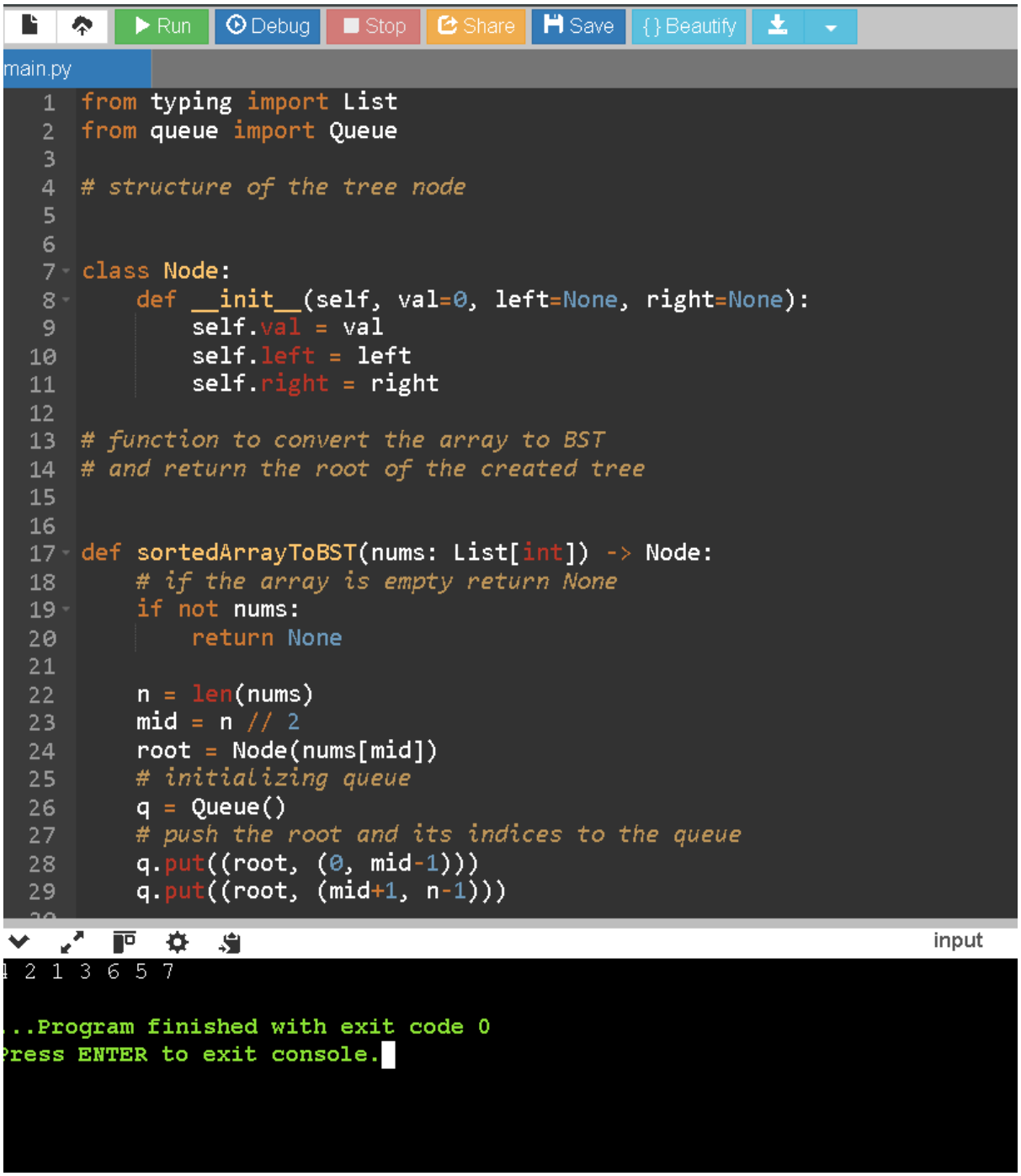
main.py

```python
def reverse_bits(n: int, bit_size: int) -> int:
    reversed_n = 0
    for _ in range(bit_size):
        reversed_n = (reversed_n << 1) | (n & 1)
        n >>= 1
    return reversed_n

# Example with 8-bit size
bit_size_8 = 8
number_8 = 13  # Binary: 00001101
reversed_number_8 = reverse_bits(number_8, bit_size_8)
print(f"Reversed bits (8-bit): {reversed_number_8:08b} ({reversed_number_8})")

# Example with 16-bit size
bit_size_16 = 16
number_16 = 29  # Binary: 0000000000011101
reversed_number_16 = reverse_bits(number_16, bit_size_16)
print(f"Reversed bits (16-bit): {reversed_number_16:016b} ({reversed_number_16})")

```

input

```
eversed bits (8-bit): 10110000 (176)
eversed bits (16-bit): 1011100000000000 (47104)


..Program finished with exit code 0
ress ENTER to exit console.
```

3. Given an integer array nums where the elements are sorted in ascending order, convert it to a height-balanced binary search tree.



4.Given a binary tree, determine if it is height-balanced

```
"""
Python3 program to check if a tree is height-balanced
"""
# A binary tree Node


class Node:
        # Constructor to create a new Node
        def __init__(self, data):
                self.data = data
                self.left = None
                self.right = None

# function to find height of binary tree


def height(root):

        # base condition when binary tree is empty
        if root is None:
                return 0
        return max(height(root.left), height(root.right)) + 1

# function to check if tree is height-balanced or not


def isBalanced(root):

        # Base condition
        if root is None:
                return True
```

```
        # for left and right subtree height
        lh = height(root.left)
        rh = height(root.right)

        # allowed values for (lh - rh) are 1, -1, 0
        if (abs(lh - rh) <= 1) and isBalanced(
                            root.left) is True and isBalanced(root.right) is True:
            return True

        # if we reach here means tree is not
        # height-balanced tree
        return False


# Driver function to test the above function
root = Node(1)
root.left = Node(2)
root.right = Node(3)
root.left.left = Node(4)
root.left.right = Node(5)
root.left.left.left = Node(8)
if isBalanced(root):
        print("Tree is balanced")
else:
        print("Tree is not balanced")

# This code is contributed by Shweta Singh
```

main.py

```python
# A binary tree Node


class Node:
    # Constructor to create a new Node
    def __init__(self, data):
        self.data = data
        self.left = None
        self.right = None

# function to find height of binary tree


def height(root):

    # base condition when binary tree is empty
    if root is None:
        return 0
    return max(height(root.left), height(root.right)) + 1

# function to check if tree is height-balanced or not


def isBalanced(root):

    # Base condition
    if root is None:
        return True

```

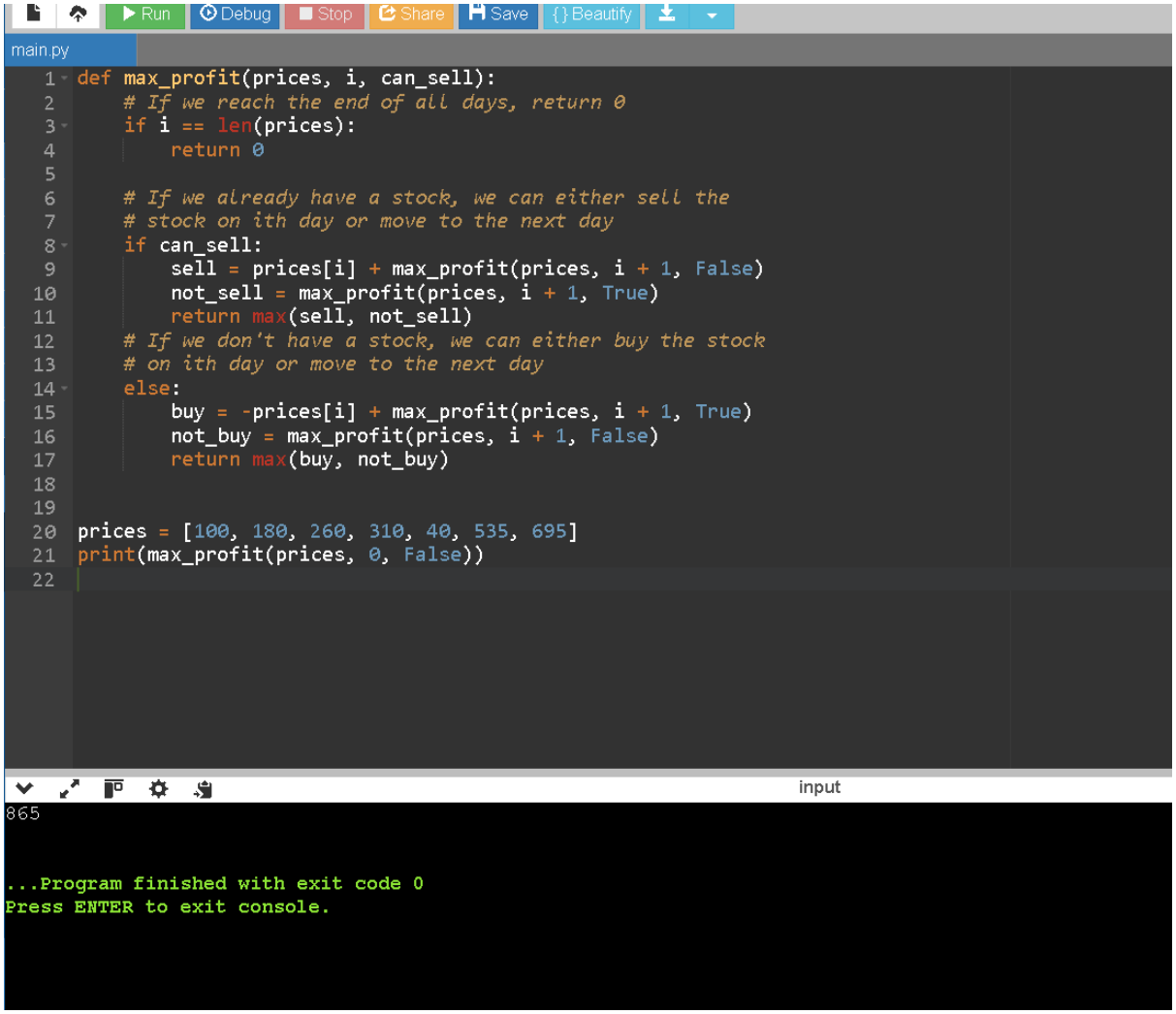
input

```
Tree is not balanced


...Program finished with exit code 0
Press ENTER to exit console.
```

5.You are given an array prices where prices[i] is the price of a given stock on the ith day.

You want to maximize your profit by choosing a single day to buy one stock and choosing a different day in the future to sell that stock.

Return the maximum profit you can achieve from this transaction. If you cannot achieve any profit, return 0.



6.Given two binary strings a and b, return their sum as a binary string.

main.py

```
def add_binary_nums(x, y):
    max_len = max(len(x), len(y))

    x = x.zfill(max_len)
    y = y.zfill(max_len)

    # initialize the result
    result = ''

    # initialize the carry
    carry = 0

    # Traverse the string
    for i in range(max_len - 1, -1, -1):
        r = carry
        r += 1 if x[i] == '1' else 0
        r += 1 if y[i] == '1' else 0
        result = ('1' if r % 2 == 1 else '0') + result
        carry = 0 if r < 2 else 1    # Compute the carry.

    if carry !=0 : result = '1' + result

    return result.zfill(max_len)

# Driver code
print(add_binary_nums('1101', '100'))

# This code is contributed
# by Anand Khatri
```

input

```
10001


...Program finished with exit code 0
Press ENTER to exit console.
```

7.You are climbing a staircase. It takes n steps to reach the top. Each time you can either climb 1 or 2 steps. In how many distinct ways can you climb to the top?

main.py

```python
def climb_stairs(n: int) -> int:
    if n == 0:
        return 1
    if n == 1:
        return 1

    prev1, prev2 = 1, 1

    for _ in range(2, n + 1):
        current = prev1 + prev2
        prev2 = prev1
        prev1 = current

    return prev1

# Example usage
print(climb_stairs(2))  # Output: 2
print(climb_stairs(3))  # Output: 3
print(climb_stairs(4))  # Output: 5
```

input

```
2
3
5


...Program finished with exit code 0
Press ENTER to exit console.
```

8. Write a function to find the longest common prefix string amongst an array of strings. If there is no common prefix, return an empty string "".

Example 1:

Input: strs = ["flower", "flow", "flight"]

Output: "fl"

Example 2:

Input: strs = ["dog", "racecar", "car"]

Output: ""

Explanation: There is no common prefix among the input strings

main.py

```python
from typing import List

def longest_common_prefix(strs: List[str]) -> str:
    if not strs:
        return ""

    # Start with the first string as the prefix
    prefix = strs[0]

    for string in strs[1:]:
        # Update the prefix length until it matches the beginning of the string
        while string[:len(prefix)] != prefix and prefix:
            prefix = prefix[:-1]

        # If at any point the prefix becomes empty, return an empty string
        if not prefix:
            return ""

    return prefix

# Example usage
strs1 = ["flower", "flow", "flight"]
print(longest_common_prefix(strs1))  # Output: "fl"

strs2 = ["dog", "racecar", "car"]
print(longest_common_prefix(strs2))  # Output: ""
```

input

```
fl


...Program finished with exit code 0
Press ENTER to exit console.
```

9.Binary tree traversal

main.py

```python
class TreeNode:
    def __init__(self, val=0, left=None, right=None):
        self.val = val
        self.left = left
        self.right = right
def inorder_traversal(root: TreeNode):
    if root is None:
        return []
    return inorder_traversal(root.left) + [root.val] + inorder_traversal(root.right)

# Example usage
root = TreeNode(1)
root.right = TreeNode(2)
root.right.left = TreeNode(3)

print(inorder_traversal(root))  # Output: [1, 3, 2]
def preorder_traversal(root: TreeNode):
    if root is None:
        return []
    return [root.val] + preorder_traversal(root.left) + preorder_traversal(root.right)

# Example usage
root = TreeNode(1)
root.right = TreeNode(2)
root.right.left = TreeNode(3)

print(preorder_traversal(root))  # Output: [1, 2, 3]

def postorder_traversal(root: TreeNode):
    if root is None:
```

input

```
[1, 3, 2]
[1, 2, 3]
[3, 2, 1]
[1, 2, 3, 4, 5, 6, 7]
```

10. Given the root of a binary tree and an integer of targetsum return true if the tree has a root to leaf such that adding up all the values