ASSIGNMENT-2

NAME: T. Nikhil Kumar Reddy

REG-NO: 192372024

SUBJECT: Python

CODE: CSA0898

1. Converting roman numbers to integers?

```python
roman = "MCMXCIV"
roman_values = {'I': 1, 'V': 5, 'X': 10, 'L': 50, 'C': 100, 'D': 500,
    'M': 1000}

total = 0
prev_value = 0

for char in reversed(roman):
    value = roman_values[char]
    if value < prev_value:
        total -= value
    else:
        total += value
    prev_value = value

print(total)
```

Output:
```
1994

=== Code Execution Successful ===
```

2. write a function to find the longest common prefix string of an array of strings. if there is no common prefix, return an empty string.

```python
def longest_common_prefix(strs):
    if not strs:
        return ""

    prefix = strs[0]
    for s in strs[1:]:
        while not s.startswith(prefix):
            prefix = prefix[:-1]
            if not prefix:
                return ""
    return prefix


strings = ["flower", "flow", "flight"]
print(longest_common_prefix(strings))
```

Output:
```
fl

=== Code Execution Successful ===
```

3. Given the root of a binary tree and an integer of target sum return true if the tree has a root to leaf such that adding up all the values?

```python
class TreeNode:
    def __init__(self, val=0, left=None, right=None):
        self.val = val
        self.left = left
        self.right = right

def has_path_sum(root, target_sum):
    if not root:
        return False
    if not root.left and not root.right:
        return root.val == target_sum

    return (has_path_sum(root.left, target_sum - root.val) or
            has_path_sum(root.right, target_sum - root.val))

root = TreeNode(5,
    TreeNode(4,
        TreeNode(11, TreeNode(7), TreeNode(2))
    ),
    TreeNode(8, TreeNode(13), TreeNode(4))
)
target_sum = 22
print(has_path_sum(root, target_sum))
```

Output:
```
True

=== Code Execution Successful ===
```

4. binary tree traversal?

```python
class TreeNode:
    def __init__(self, val=0, left=None, right=None):
        self.val = val
        self.left = left
        self.right = right
root = TreeNode(1,
    TreeNode(2, TreeNode(4), TreeNode(5)),
    TreeNode(3)
)
stack, result = [], []
current = root
while stack or current:
    while current:
        stack.append(current)
        current = current.left
    current = stack.pop()
    result.append(current.val)
    current = current.right

print("In-order:", result)

stack, result = [root], []
while stack:
    current = stack.pop()
```

Output:
```
In-order: [4, 2, 5, 1, 3]
Pre-order: [1, 2, 4, 5, 3]
Post-order: [4, 5, 2, 3, 1]

=== Code Execution Successful ===
```

5. bit reversing?

```python
class TreeNode:
    def __init__(self, val=0, left=None, right=None):
        self.val = val
        self.left = left
        self.right = right
root = TreeNode(1,
        TreeNode(2, TreeNode(4), TreeNode(5)),
        TreeNode(3)
    )
stack, result = [], []
current = root
while stack or current:
    while current:
        stack.append(current)
        current = current.left
    current = stack.pop()
    result.append(current.val)
    current = current.right

print("In-order:", result)

stack, result = [root], []
while stack:
    current = stack.pop()
```

Output:
```
In-order: [4, 2, 5, 1, 3]
Pre-order: [1, 2, 4, 5, 3]
Post-order: [4, 5, 2, 3, 1]

=== Code Execution Successful ===
```

6. convert sorted array to binary search tree given an integer array nums where the element are sorted in ascending order .convert it to a height-balanced.

```python
class TreeNode:
    def __init__(self, val=0, left=None, right=None):
        self.val = val
        self.left = left
        self.right = right
def sorted_array_to_bst(nums):
    if not nums:
        return None

    mid = len(nums) // 2
    root = TreeNode(nums[mid])
    root.left = sorted_array_to_bst(nums[:mid])
    root.right = sorted_array_to_bst(nums[mid+1:])

    return root
nums = [-10, -3, 0, 5, 9]
root = sorted_array_to_bst(nums)
stack, result = [], []
current = root
while stack or current:
    while current:
        stack.append(current)
        current = current.left
    current = stack.pop()
```

Output:
```
In-order traversal of the BST: [-10, -3, 0, 5, 9]

=== Code Execution Successful ===
```

7. Given a binary tree, determine if it is heighted-balanced?
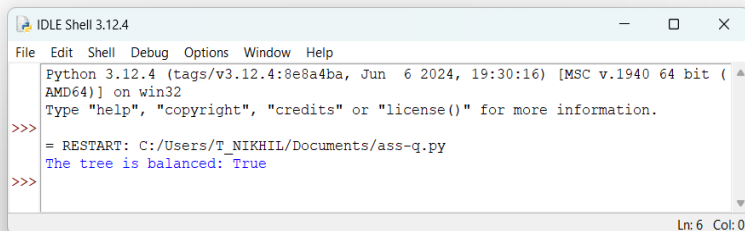
```python
class TreeNode:
    def __init__(self, val=0, left=None, right=None):
        self.val = val
        self.left = left
        self.right = right
root = TreeNode(1,
        TreeNode(2, TreeNode(4), TreeNode(5)),
        TreeNode(3)
    )
def is_balanced_tree(root):
    def check_height(node):
        if not node:
            return 0
        left_height = check_height(node.left)
        right_height = check_height(node.right)
        if left_height == -1 or right_height == -1 or abs(left_height - right_height) > 1:
            return -1
        return max(left_height, right_height) + 1

    return check_height(root) != -1

print("The tree is balanced:", is_balanced_tree(root))
```
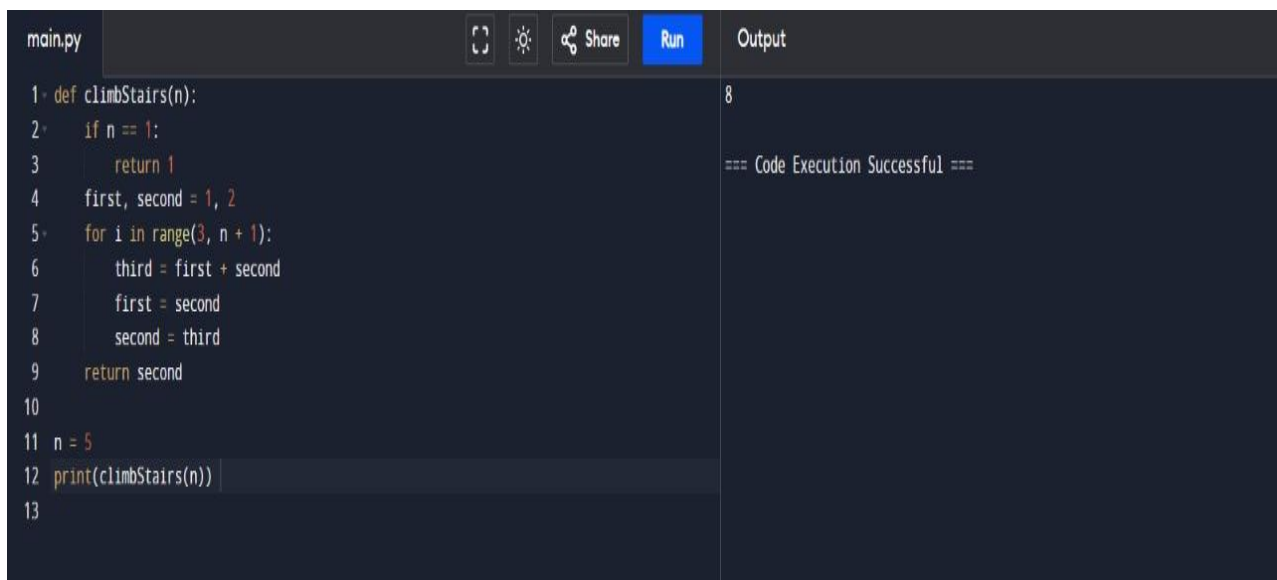
```
IDLE Shell 3.12.4                                        —    □    ×

File  Edit  Shell  Debug  Options  Window  Help
    Python 3.12.4 (tags/v3.12.4:8e8a4ba, Jun  6 2024, 19:30:16) [MSC v.1940 64 bit (
    AMD64)] on win32
    Type "help", "copyright", "credits" or "license()" for more information.
>>>
    = RESTART: C:/Users/T_NIKHIL/Documents/ass-q.py
    The tree is balanced: True
>>>
                                                          Ln: 6  Col: 0
```

8. climbing stairs, you are climbing a stair case it takes n steps to reach the top each time you can either climb 1 or 2 steps in how may distinct ways can you climb to the top?

```python
def climbStairs(n):
    if n == 1:
        return 1
    first, second = 1, 2
    for i in range(3, n + 1):
        third = first + second
        first = second
        second = third
    return second


n = 5
print(climbStairs(n))
```

```
Output

8

=== Code Execution Successful ===
```

9. best time to buy and see stock ,you are given an array prices e=where prices[i] is the price of a given stock on the ith day. you want to make your profit by choosing a single day in the futher to see stocks if you cannot archieve any profit, return 0

```python
1  prices = [7, 1, 5, 3, 6, 4]
2  min_price = float('inf')
3  max_profit = 0
4  for price in prices:
5      if price < min_price:
6          min_price = price
7      profit = price - min_price
8      if profit > max_profit:
9          max_profit = profit
10
11 print("Maximum profit:", max_profit)
12
```

```
Maximum profit: 5

=== Code Execution Successful ===
```

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

```python
1  def add_binary(a, b):
2      return bin(int(a, 2) + int(b, 2))[2:]
3
4  a = "11"
5  b = "1"
6  print(add_binary(a, b))
7
```

```
100

=== Code Execution Successful ===
```