ASSIGNMENT-1

NAME: K.C. Thimma Reddy

REG-NO: 192325025

SUBJECT: Python

CODE: CSA0898

1.Converting Roman Numbers to integers?

main.py Share Run Output

```python
def roman_to_int(s):
    roman_dict = {'I': 1, 'V': 5, 'X': 10, 'L': 50, 'C': 100, 'D': 500, 'M': 1000}
    result = 0
    prev_value = 0

    for char in s:
        value = roman_dict[char]
        result += value
        if value > prev_value:
            result -= 2 * prev_value
        prev_value = value

    return result
print(roman_to_int("IV"))
print(roman_to_int("IX"))
print(roman_to_int("LVIII"))
print(roman_to_int("MCMXCIV"))

```

```
4
9
58
1994

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

2.

main.py Share Run Output

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

    prefix = strs[0]

    for s in strs[1:]:
        while s[:len(prefix)] != prefix:
            prefix = prefix[:-1]
            if not prefix:
                return ""

    return prefix


print(longestCommonPrefix(["flower", "flow", "flight"]))
print(longestCommonPrefix(["dog", "racecar", "car"]))

```

```
fl


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

3.

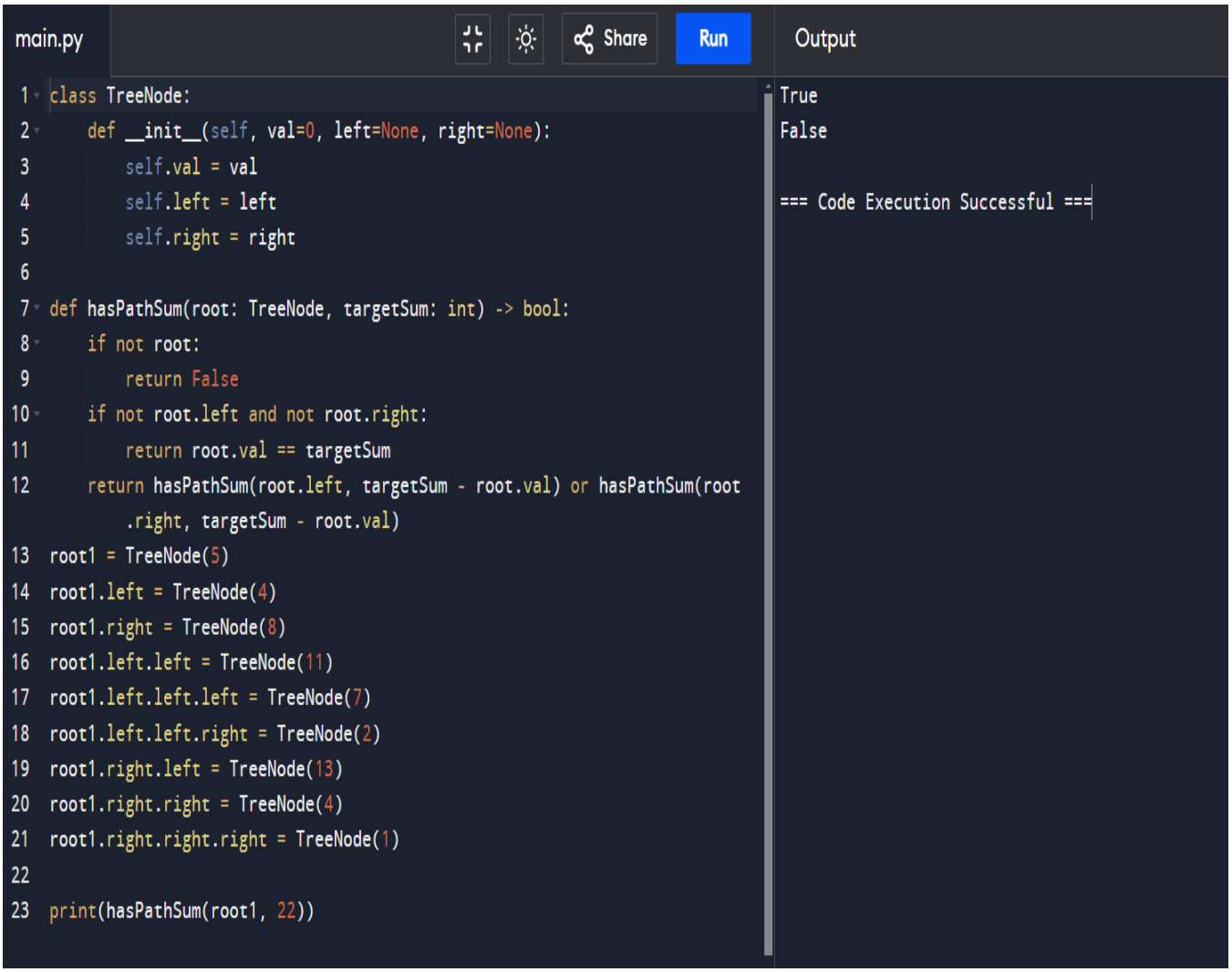
main.py

Share Run Output

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

def hasPathSum(root: TreeNode, targetSum: int) -> bool:
    if not root:
        return False
    if not root.left and not root.right:
        return root.val == targetSum
    return hasPathSum(root.left, targetSum - root.val) or hasPathSum(root
        .right, targetSum - root.val)
root1 = TreeNode(5)
root1.left = TreeNode(4)
root1.right = TreeNode(8)
root1.left.left = TreeNode(11)
root1.left.left.left = TreeNode(7)
root1.left.left.right = TreeNode(2)
root1.right.left = TreeNode(13)
root1.right.right = TreeNode(4)
root1.right.right.right = TreeNode(1)

print(hasPathSum(root1, 22))
```

```
True
False

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

4. 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



5.bit reversing?

main.py | Share | Run | Output

```python
def reverseBits(n: int) -> int:
    result = 0
    for i in range(32):
        result = (result << 1) | (n & 1)
        n >>= 1
    return result


num = 43261596
reversed_num = reverseBits(num)
print(reversed_num)

```

```
964176192

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

6. convert sorted array to binary search tree?

main.py | Share | Run | Output

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

def print_bst_inorder(root):
    if root:
        print_bst_inorder(root.left)
```

```
Inorder traversal of the BST:
-10 -3 0 5 9
=== Code Execution Successful ===
```

7. given a binary tree, determine if it is height-balanced?

main.py

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

def is_balanced(root):
    def check_height(node):
        if not node:
            return 0

        left_height = check_height(node.left)
        if left_height == -1:
            return -1

        right_height = check_height(node.right)
        if right_height == -1:
            return -1


        if abs(left_height - right_height) > 1:
            return -1

        return max(left_height, right_height) + 1
```

Output

```
Is the tree balanced? True

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

8. Climbing stairs?

main.py

```python
def climb_stairs(n):
    if n <= 1:
        return 1


    dp = [0] * (n + 1)


    dp[0] = 1
    dp[1] = 1


    for i in range(2, n + 1):
        dp[i] = dp[i - 1] + dp[i - 2]

    return dp[n]


n = 5
print("Number of ways to climb the stairs:", climb_stairs(n))
```

Output

```
Number of ways to climb the stairs: 8

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

9. best time to buy and sell stock?

main.py Share Run Output

```python
def max_profit(prices):
    if not prices or len(prices) < 2:
        return 0

    min_price = float('inf')
    max_profit = 0

    for price in prices:

        if price < min_price:
            min_price = price


        profit = price - min_price


        if profit > max_profit:
            max_profit = profit

    return max_profit


prices = [7, 1, 5, 3, 6, 4]
print("Maximum profit:", max_profit(prices))
```

```
Maximum profit: 5

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

10.sum of two binary strings?

main.py Share Run Output

```python
def add_binary(a: str, b: str) -> str:

    int_a = int(a, 2)
    int_b = int(b, 2)


    sum_int = int_a + int_b


    return bin(sum_int)[2:]


binary_str1 = "1010"
binary_str2 = "1101"
print("Sum of binary strings:", add_binary(binary_str1, binary_str2))

```

```
Sum of binary strings: 10111

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