

## **ASSIGNMENT-2**

NAME: D. Guna vardhan

REG-NO: 192372041

SUBJECT: Python

CODE: CSA0898

## 1. Converting roman numbers to integers?

main.py	Output
<pre>1 roman = "MCMXCIV" 2 roman_values = {'I': 1, 'V': 5, 'X': 10, 'L': 50, 'C': 100, 'D': 500,                   'M': 1000} 3 4 total = 0 5 prev_value = 0 6 7 for char in reversed(roman): 8     value = roman_values[char] 9     if value &lt; prev_value: 10         total -= value 11     else: 12         total += value 13     prev_value = value 14 15 print(total) 16</pre>	<pre>1994  === Code Execution Successful ===</pre>

## 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.

main.py	Output
<pre>1 def longest_common_prefix(strs): 2     if not strs: 3         return "" 4 5     prefix = strs[0] 6     for s in strs[1:]: 7         while not s.startswith(prefix): 8             prefix = prefix[:-1] 9         if not prefix: 10             return "" 11     return prefix 12 13 14 strings = ["flower", "flow", "flight"] 15 print(longest_common_prefix(strings)) 16</pre>	<pre>f1  === Code Execution Successful ===</pre>

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?

main.py	Output
<pre>1 class TreeNode: 2     def __init__(self, val=0, left=None, right=None): 3         self.val = val 4         self.left = left 5         self.right = right 6 7 def has_path_sum(root, target_sum): 8     if not root: 9         return False 10    if not root.left and not root.right: 11        return root.val == target_sum 12 13    return (has_path_sum(root.left, target_sum - root.val) or 14            has_path_sum(root.right, target_sum - root.val)) 15 16 root = TreeNode(5, 17                 TreeNode(4, 18                         TreeNode(11, TreeNode(7), TreeNode(2)) 19                         ), 20                 TreeNode(8, TreeNode(13), TreeNode(4)) 21                 ) 22 target_sum = 22 23 print(has_path_sum(root, target_sum)) 24</pre>	<pre>True === Code Execution Successful ===</pre>

4. binary tree traversal?

main.py	Output
<pre>1 class TreeNode: 2     def __init__(self, val=0, left=None, right=None): 3         self.val = val 4         self.left = left 5         self.right = right 6 7 root = TreeNode(1, 8                 TreeNode(2, TreeNode(4), TreeNode(5)), 9                 TreeNode(3) 10                ) 11 stack, result = [], [] 12 current = root 13 while stack or current: 14     while current: 15         stack.append(current) 16         current = current.left 17     current = stack.pop() 18     result.append(current.val) 19     current = current.right 20 print("In-order:", result) 21 22 stack, result = [root], [] 23 while stack: 24     current = stack.pop()</pre>	<pre>In-order: [4, 2, 5, 1, 3] Pre-order: [1, 2, 4, 5, 3] Post-order: [4, 5, 2, 3, 1] === Code Execution Successful ===</pre>

## 5. bit reversing?

```
main.py  [Icons]  Share  Run  Output

1 class TreeNode:
2     def __init__(self, val=0, left=None, right=None):
3         self.val = val
4         self.left = left
5         self.right = right
6     root = TreeNode(1,
7                     TreeNode(2, TreeNode(4), TreeNode(5)),
8                     TreeNode(3))
9
10    stack, result = [], []
11    current = root
12    while stack or current:
13        while current:
14            stack.append(current)
15            current = current.left
16        current = stack.pop()
17        result.append(current.val)
18        current = current.right
19
20    print("In-order:", result)
21
22    stack, result = [root], []
23    while stack:
24        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.

```
main.py  [Icons]  Share  Run  Output

1 class TreeNode:
2     def __init__(self, val=0, left=None, right=None):
3         self.val = val
4         self.left = left
5         self.right = right
6     def sorted_array_to_bst(nums):
7         if not nums:
8             return None
9
10        mid = len(nums) // 2
11        root = TreeNode(nums[mid])
12        root.left = sorted_array_to_bst(nums[:mid])
13        root.right = sorted_array_to_bst(nums[mid+1:])
14
15        return root
16    nums = [-10, -3, 0, 5, 9]
17    root = sorted_array_to_bst(nums)
18    stack, result = [], []
19    current = root
20    while stack or current:
21        while current:
22            stack.append(current)
23            current = current.left
24        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 height-balanced?

ass-q.py - C:/Users/T\_NIKHIL/Documents/ass-q.py (3.12.4)

File Edit Format Run Options Window Help

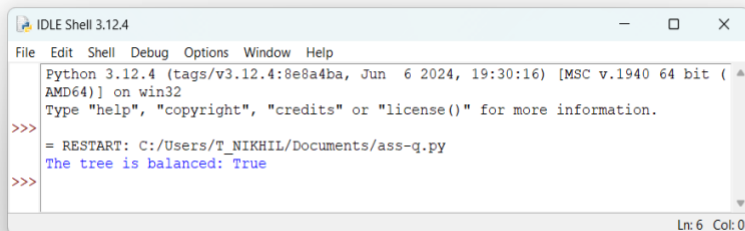
```
class TreeNode:
    def __init__(self, val=0, left=None, right=None):
        self.val = val
        self.left = left
        self.right = right

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

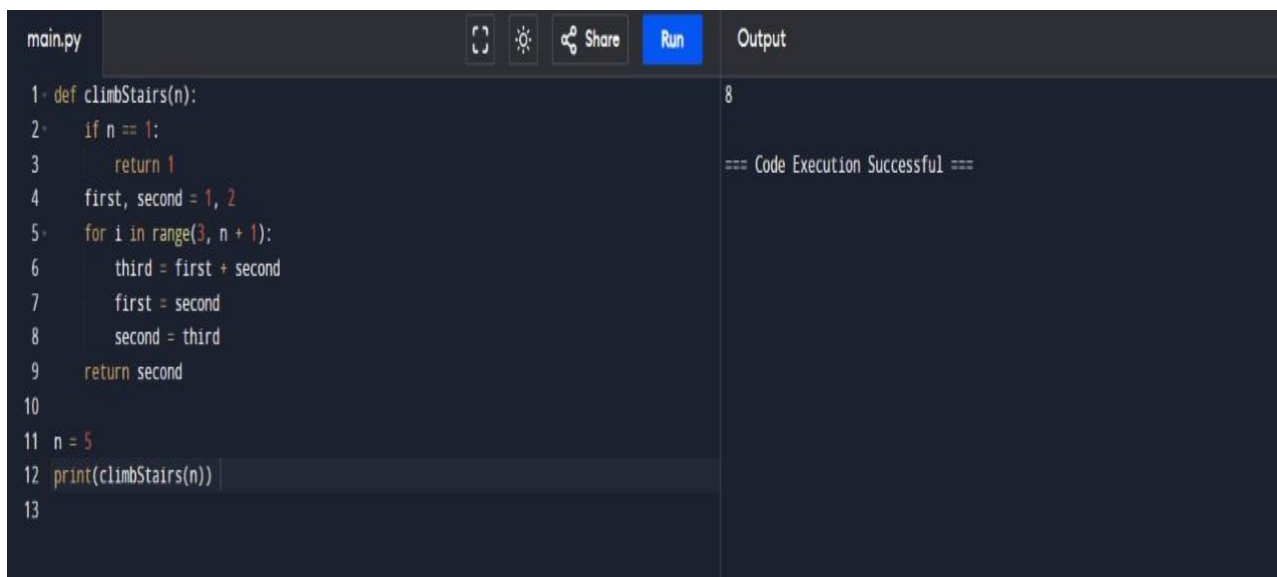
    return check_height(root) != -1

print("The tree is balanced:", is_balanced_tree(root))
```



```
IDLE Shell 3.12.4
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
>>>
```

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 many distinct ways can you climb to the top?

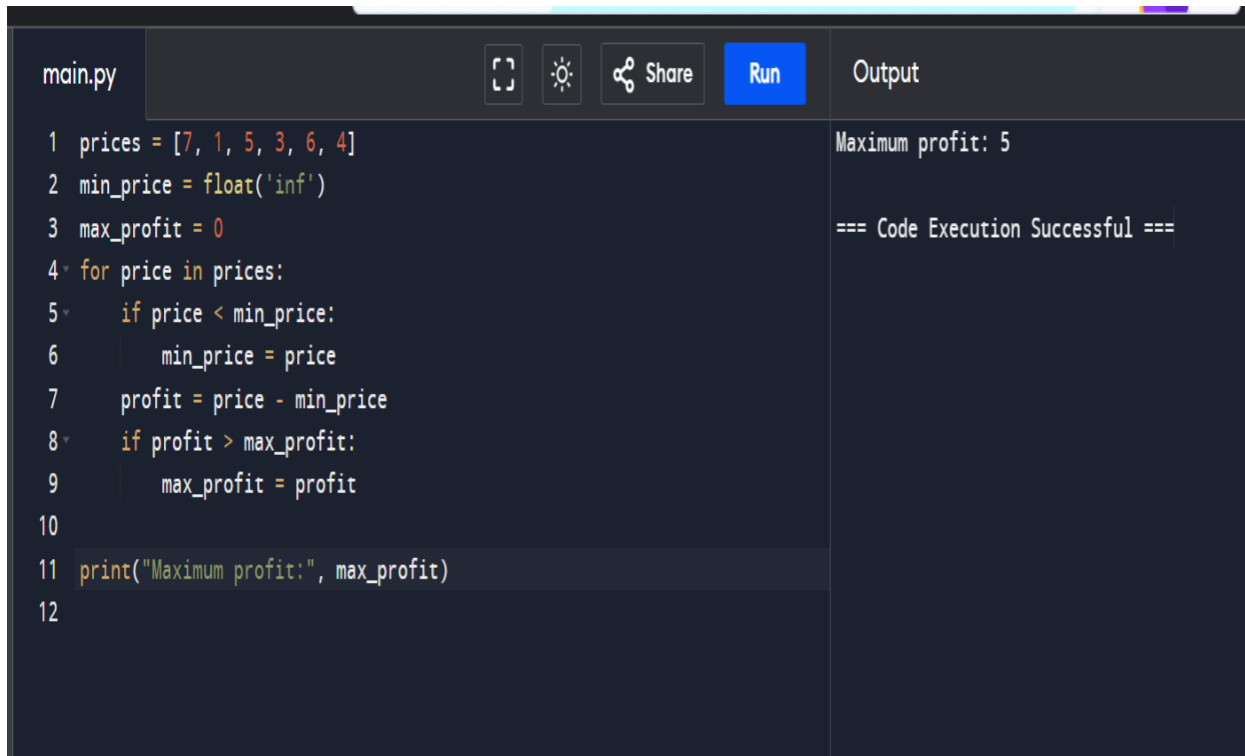


```
main.py
1 def climbStairs(n):
2     if n == 1:
3         return 1
4     first, second = 1, 2
5     for i in range(3, n + 1):
6         third = first + second
7         first = second
8         second = third
9     return second
10
11 n = 5
12 print(climbStairs(n))
13
```

8

=== Code Execution Successful ===

9. best time to buy and sell stock ,you are given an array prices 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 future to sell stocks if you cannot achieve any profit, return 0

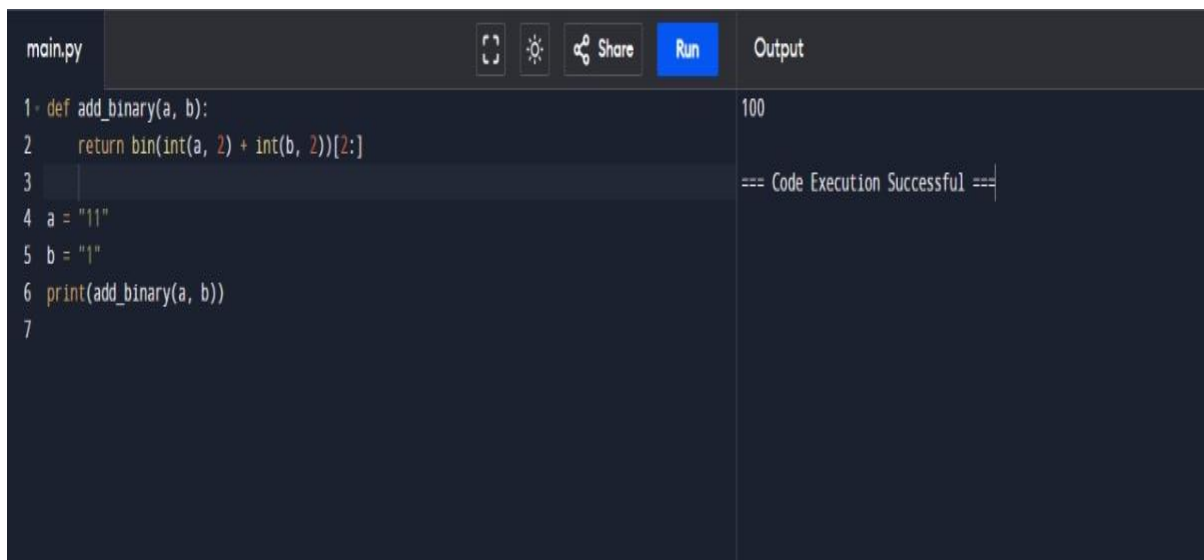


```
main.py  [Icons] Share Run Output
1 prices = [7, 1, 5, 3, 6, 4]
2 min_price = float('inf')
3 max_profit = 0
4 for price in prices:
5     if price < min_price:
6         min_price = price
7     profit = price - min_price
8     if profit > max_profit:
9         max_profit = profit
10
11 print("Maximum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



```
main.py  [Icons] Share Run Output
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 ===