ASSIGNMENT-2

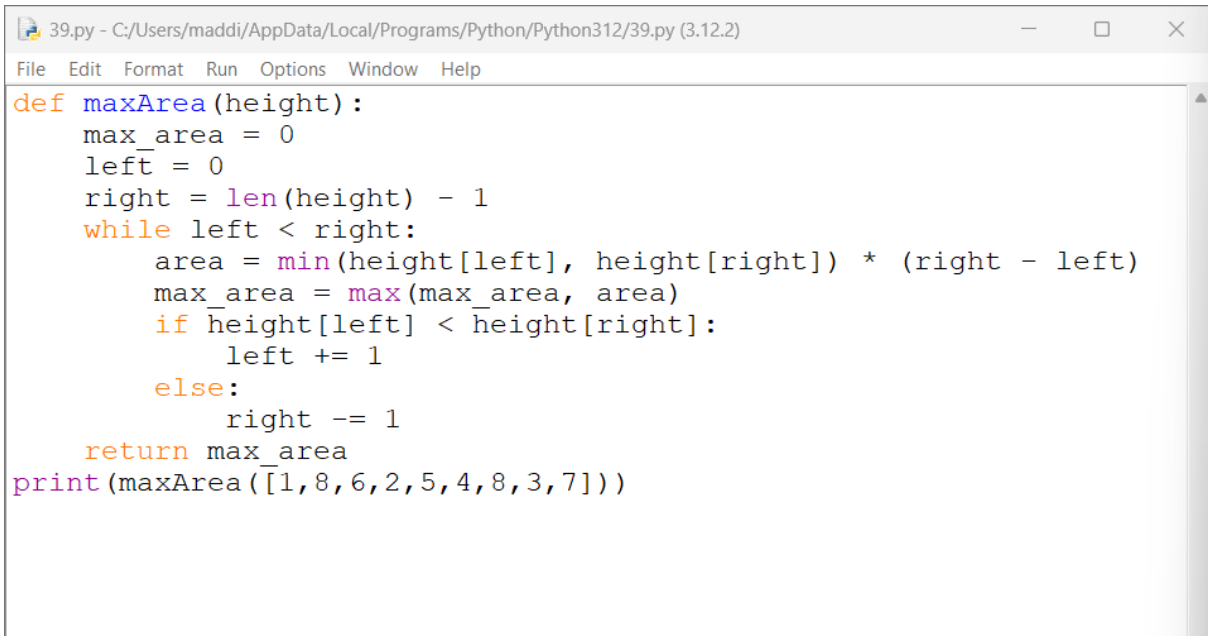
NAME : M.Mahathi

REGISTER NUMBER : 192324098

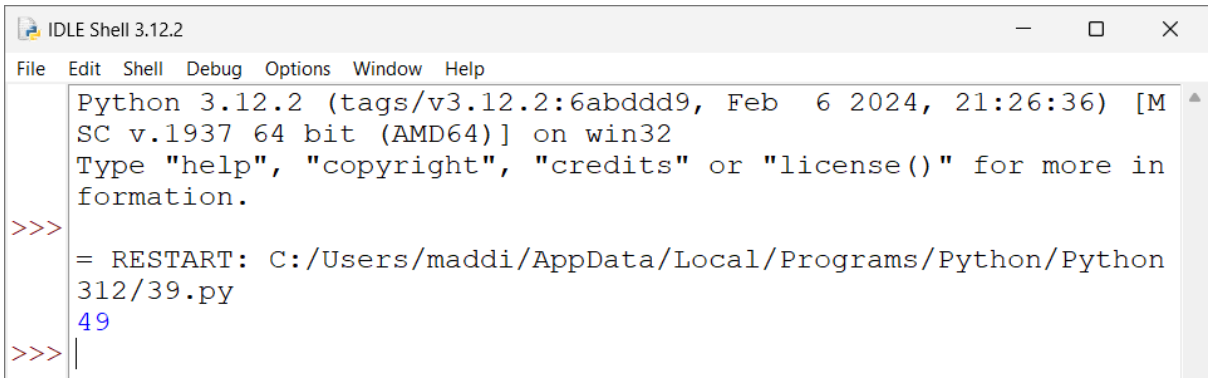
11. Container With Most Water You are given an integer array height of length n. There are n vertical lines drawn such that the two endpoints of the ith line are (i, 0) and (i, height[i]). Find two lines that together with the x-axis form a container, such that the container contains the most water. Return the maximum amount of water a container can store. Notice that you may not slant the container. Example 1: Input: height = [1,8,6,2,5,4,8,3,7]

Output: 49 Explanation: The above vertical lines are represented by array [1,8,6,2,5,4,8,3,7]. In this case, the max area of water (blue section) the container can contain is 49.

PROGRAM :



OUTPUT :



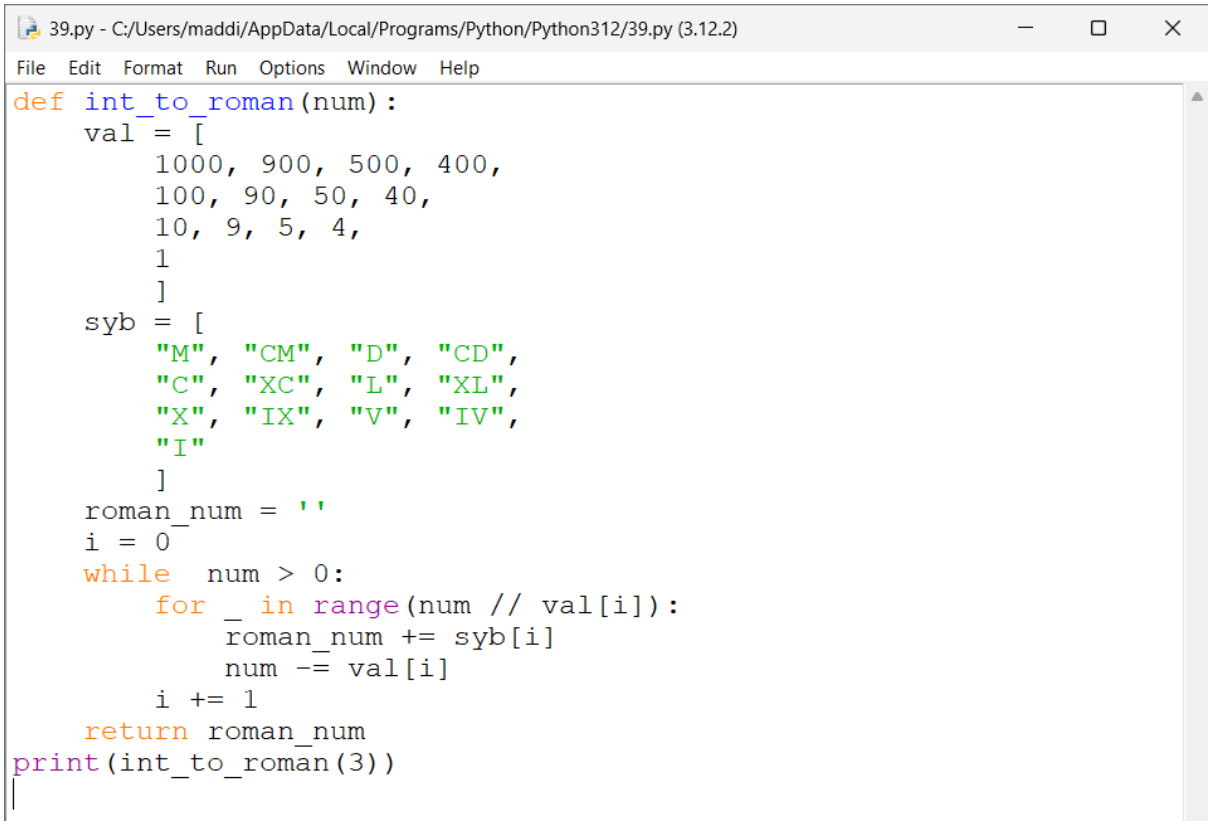
12. Integer to Roman Roman numerals are represented by seven different symbols: I, V, X, L, C, D and M. Symbol Value I 1 V 5 X 10 L 50 C 100 D 500 M 1000 For example, 2 is written as II in Roman

numeral, just two one's added together. 12 is written as XII, which is simply X + II. The number 27 is written as XXVII, which is XX + V + II. Roman numerals are usually written largest to smallest from left to right. However, the numeral for four is not IIII. Instead, the number four is written as IV. Because the one is before the five we subtract it making four. The same principle applies to the number nine, which is written as IX. There are six instances where subtraction is used:

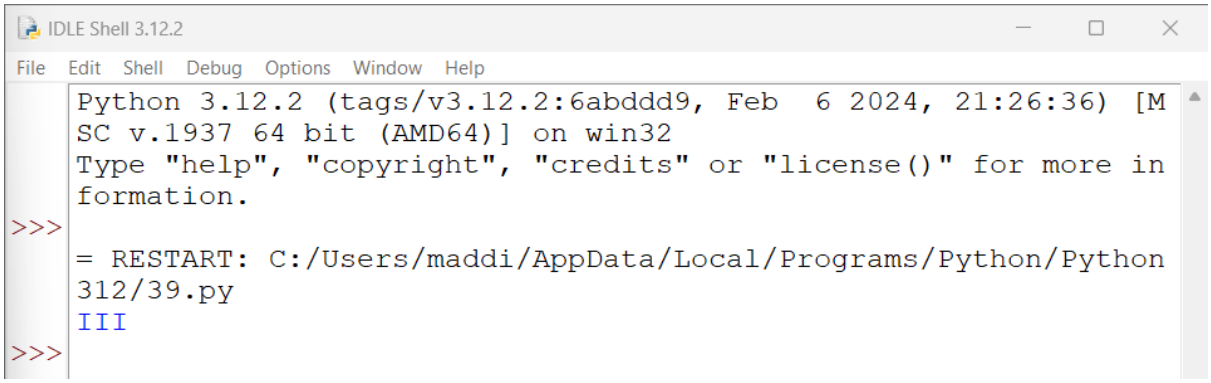
- I can be placed before V (5) and X (10) to make 4 and 9.
- X can be placed before L (50) and C (100) to make 40 and 90.
- C can be placed before D (500) and M (1000) to make 400 and 900.

Given an integer, convert it to a roman numeral. Example 1: Input: num = 3 Output: "III" Explanation: 3 is represented as 3 ones.

PROGRAM :

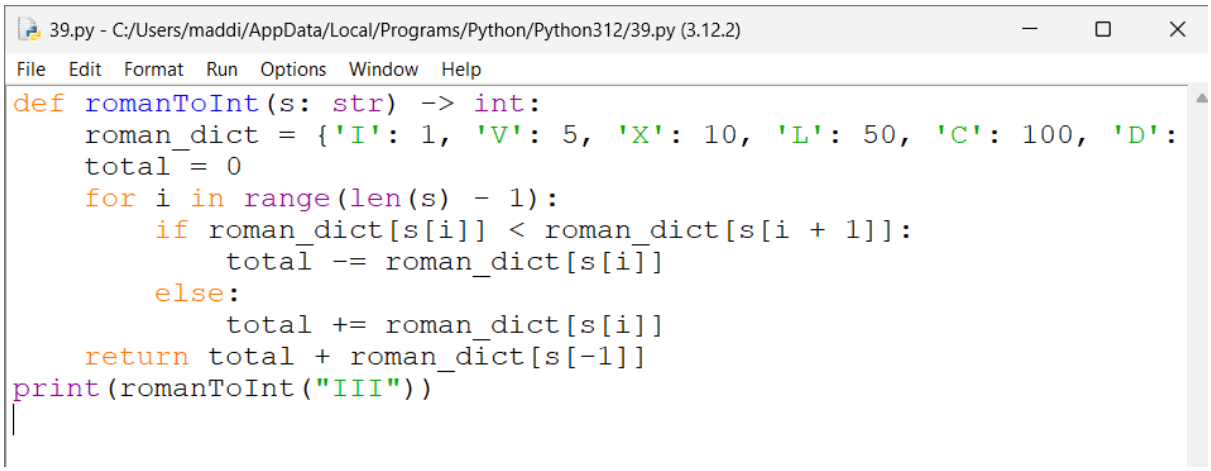


OUTPUT :

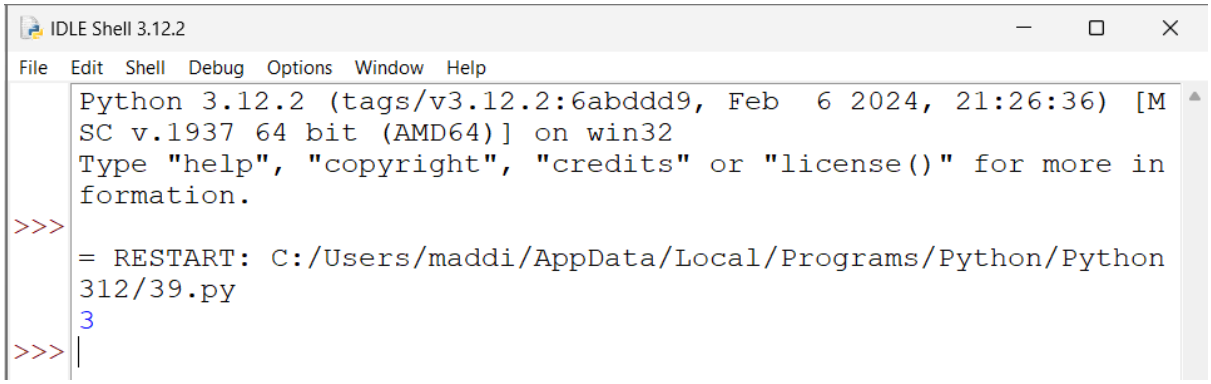


13. Roman to Integer Roman numerals are represented by seven different symbols: I, V, X, L, C, D and M. Symbol Value I 1 V 5 X 10 L 50 C 100 D 500 M 1000 For example, 2 is written as II in Roman numeral, just two ones added together. 12 is written as XII, which is simply X + II. The number 27 is written as XXVII, which is XX + V + II. Roman numerals are usually written largest to smallest from left to right. However, the numeral for four is not IIII. Instead, the number four is written as IV. Because the one is before the five we subtract it making four. The same principle applies to the number nine, which is written as IX. There are six instances where subtraction is used: ● I can be placed before V (5) and X (10) to make 4 and 9. ● X can be placed before L (50) and C (100) to make 40 and 90. ● C can be placed before D (500) and M (1000) to make 400 and 900. Given a roman numeral, convert it to an integer. Example 1: Input: s = "III" Output: 3 Explanation: III = 3.

PROGRAM :



OUTPUT :



14. Longest Common Prefix 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"

PROGRAM :

39.py - C:/Users/maddi/AppData/Local/Programs/Python/Python312/39.py (3.12.2)

File Edit Format Run Options Window Help

```
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
example1 = ["flower", "flow", "flight"]
print(longest_common_prefix(example1))
```

OUTPUT :

IDLE Shell 3.12.2

File Edit Shell Debug Options Window Help

```
Python 3.12.2 (tags/v3.12.2:6abddd9, Feb  6 2024, 21:26:36) [M
SC v.1937 64 bit (AMD64)] on win32
Type "help", "copyright", "credits" or "license()" for more in
formation.
>>>
= RESTART: C:/Users/maddi/AppData/Local/Programs/Python/Python
312/39.py
fl
>>>
```

15. 3Sum Given an integer array nums, return all the triplets [nums[i], nums[j], nums[k]] such that i != j, i != k, and j != k, and nums[i] + nums[j] + nums[k] == 0. Notice that the solution set must not contain duplicate triplets. Example 1: Input: nums = [-1,0,1,2,-1,-4] Output: [[-1,-1,2],[-1,0,1]] Explanation: nums[0] + nums[1] + nums[2] = (-1) + 0 + 1 = 0. nums[1] + nums[2] + nums[4] = 0 + 1 + (-1) = 0. nums[0] + nums[3] + nums[4] = (-1) + 2 + (-1) = 0. The distinct triplets are [-1,0,1] and [-1,-1,2]. Notice that the order of the output and the order of the triplets does not matter.

PROGRAM :

39.py - C:/Users/maddi/AppData/Local/Programs/Python/Python312/39.py (3.12.2)

File Edit Format Run Options Window Help

```
def threeSum(nums):
    nums.sort()
    result = []
    for i in range(len(nums) - 2):
        if i > 0 and nums[i] == nums[i - 1]:
            continue
        l, r = i + 1, len(nums) - 1
        while l < r:
            s = nums[i] + nums[l] + nums[r]
            if s < 0:
                l += 1
            elif s > 0:
                r -= 1
            else:
                result.append([nums[i], nums[l], nums[r]])
                while l < r and nums[l] == nums[l + 1]:
                    l += 1
                while l < r and nums[r] == nums[r - 1]:
                    r -= 1
                l += 1
                r -= 1
    return result
nums = [-1, 0, 1, 2, -1, -4]
result = threeSum(nums)
print(result)
```

OUTPUT :

IDLE Shell 3.12.2

File Edit Shell Debug Options Window Help

```
Python 3.12.2 (tags/v3.12.2:6abddd9, Feb  6 2024, 21:26:36) [M
SC v.1937 64 bit (AMD64)] on win32
Type "help", "copyright", "credits" or "license()" for more in
formation.
>>>
= RESTART: C:/Users/maddi/AppData/Local/Programs/Python/Python
312/39.py
[[-1, -1, 2], [-1, 0, 1]]
>>>
```

16. 3Sum Closest Given an integer array nums of length n and an integer target, find three integers in nums such that the sum is closest to target. Return the sum of the three integers. You may assume that each input would have exactly one solution. Example 1: Input: nums = [-1,2,1,-4], target = 1 Output: 2 Explanation: The sum that is closest to the target is 2. (-1 + 2 + 1 = 2).

PROGRAM :

39.py - C:/Users/maddi/AppData/Local/Programs/Python/Python312/39.py (3.12.2)

File Edit Format Run Options Window Help

```
def threeSumClosest(nums, target):
    nums.sort()
    closest_sum = float('inf')
    for i in range(len(nums) - 2):
        left, right = i + 1, len(nums) - 1
        while left < right:
            current_sum = nums[i] + nums[left] + nums[right]
            if current_sum == target:
                return current_sum
            if abs(current_sum - target) < abs(closest_sum - targe
                closest_sum = current_sum
            if current_sum < target:
                left += 1
            else:
                right -= 1
    return closest_sum
nums = [-1,2,1,-4]
target = 1
closest_sum = threeSumClosest(nums, target)
print(closest_sum)
```

OUTPUT :

IDLE Shell 3.12.2

File Edit Shell Debug Options Window Help

```
Python 3.12.2 (tags/v3.12.2:6abddd9, Feb  6 2024, 21:26:36) [M
SC v.1937 64 bit (AMD64)] on win32
Type "help", "copyright", "credits" or "license()" for more in
formation.
>>>
= RESTART: C:/Users/maddi/AppData/Local/Programs/Python/Python
312/39.py
2
>>>
```

17. Letter Combinations of a Phone Number Given a string containing digits from 2-9 inclusive, return all possible letter combinations that the number could represent. Return the answer in any order. A mapping of digits to letters (just like on the telephone buttons) is given below. Note that 1 does not map to any letters. Example 1: Input: digits = "23" Output:
["ad","ae","af","bd","be","bf","cd","ce","cf"]

PROGRAM :

39.py - C:/Users/maddi/AppData/Local/Programs/Python/Python312/39.py (3.12.2)

File Edit Format Run Options Window Help

```
def letterCombinations(digits):
    if not digits:
        return []
    phone = {
        '2': ['a', 'b', 'c'],
        '3': ['d', 'e', 'f'],
        '4': ['g', 'h', 'i'],
        '5': ['j', 'k', 'l'],
        '6': ['m', 'n', 'o'],
        '7': ['p', 'q', 'r', ''],
        '8': ['t', 'u', 'v'],
        '9': ['w', 'x', 'y', 'z']
    }
    def backtrack(combination, next_digits):
        if len(next_digits) == 0:
            output.append(combination)
        else:
            for letter in phone[next_digits[0]]:
                backtrack(combination + letter, next_digits[1:])

    output = []
    backtrack("", digits)
    return output
print(letterCombinations("23"))
```

OUTPUT :

IDLE Shell 3.12.2

File Edit Shell Debug Options Window Help

```
Python 3.12.2 (tags/v3.12.2:6abddd9, Feb  6 2024, 21:26:36) [M
SC v.1937 64 bit (AMD64)] on win32
Type "help", "copyright", "credits" or "license()" for more in
formation.
>>>
= RESTART: C:/Users/maddi/AppData/Local/Programs/Python/Python
312/39.py
['ad', 'ae', 'af', 'bd', 'be', 'bf', 'cd', 'ce', 'cf']
>>>
```

18. 4Sum Given an array nums of n integers, return an array of all the unique quadruplets [nums[a], nums[b], nums[c], nums[d]] such that: ● 0 <= a, b, c, d < n ● a, b, c, and d are distinct. ● nums[a] + nums[b] + nums[c] + nums[d] == target You may return the answer in any order. Example 1: Input: nums = [1,0,-1,0,-2,2], target = 0 Output: [[-2,-1,1,2],[-2,0,0,2],[-1,0,0,1]]

PROGRAM :

39.py - C:/Users/maddi/AppData/Local/Programs/Python/Python312/39.py (3.12.2)

File Edit Format Run Options Window Help

```
def fourSum(nums, target):
    nums.sort()
    result = []
    for i in range(len(nums) - 3):
        if i > 0 and nums[i] == nums[i - 1]:
            continue
        for j in range(i + 1, len(nums) - 2):
            if j > i + 1 and nums[j] == nums[j - 1]:
                continue
            left, right = j + 1, len(nums) - 1
            while left < right:
                total = nums[i] + nums[j] + nums[left] + nums[right]
                if total < target:
                    left += 1
                elif total > target:
                    right -= 1
                else:
                    result.append([nums[i], nums[j], nums[left], nums[right]])
                    while left < right and nums[left] == nums[left + 1]:
                        left += 1
                    while left < right and nums[right] == nums[right - 1]:
                        right -= 1
                    left += 1
                    right -= 1
    return result
print(fourSum([1, 0, -1, 0, -2, 2], 0))
```

OUTPUT :

IDLE Shell 3.12.2

File Edit Shell Debug Options Window Help

```
Python 3.12.2 (tags/v3.12.2:6abddd9, Feb  6 2024, 21:26:36) [MSC v.1937 64 bit (AMD64)] on win32
Type "help", "copyright", "credits" or "license()" for more information.
>>>
= RESTART: C:/Users/maddi/AppData/Local/Programs/Python/Python312/39.py
[[-2, -1, 1, 2], [-2, 0, 0, 2], [-1, 0, 0, 1]]
>>>
```

19. Remove Nth Node From End of List Given the head of a linked list, remove the nth node from the end of the list and return its head. Example 1: Input: head = [1,2,3,4,5], n = 2 Output: [1,2,3,5]

PROGRAM :

39.py - C:/Users/maddi/AppData/Local/Programs/Python/Python312/39.py (3.12.2)

File Edit Format Run Options Window Help

```
class ListNode:
    def __init__(self, val=0, next=None):
        self.val = val
        self.next = next
def removeNthFromEnd(head: ListNode, n: int) -> ListNode:
    dummy = ListNode(0, head)
    first = dummy
    second = dummy
    for _ in range(n + 1):
        first = first.next
    while first is not None:
        first = first.next
        second = second.next
    second.next = second.next.next
    return dummy.next
def list_to_linkedlist(arr):
    if not arr:
        return None
    head = ListNode(arr[0])
    current = head
    for val in arr[1:]:
        current.next = ListNode(val)
        current = current.next
    return head
def linkedlist_to_list(node):
    arr = []
    while node:
        arr.append(node.val)
        node = node.next
    return arr
head = list_to_linkedlist([1, 2, 3, 4, 5])
n = 2
new_head = removeNthFromEnd(head, n)
print(linkedlist_to_list(new_head))
```

OUTPUT :

IDLE Shell 3.12.2

File Edit Shell Debug Options Window Help

```
Python 3.12.2 (tags/v3.12.2:6abddd9, Feb  6 2024, 21:26:36) [MSC v.1937 64 bit (AMD64)] on win32
Type "help", "copyright", "credits" or "license()" for more information.
>>> 
= RESTART: C:/Users/maddi/AppData/Local/Programs/Python/Python312/39.py
[1, 2, 3, 5]
>>> 
```

20. Valid Parentheses Given a string s containing just the characters '(', ')', '{', '}', '[' and ']', determine if the input string is valid. An input string is valid if: 1. Open brackets must be closed by the same type of brackets. 2. Open brackets must be closed in the correct order. 3. Every close bracket has a corresponding open bracket of the same type. Example 1: Input: s = "()" Output: true.

PROGRAM :

39.py - C:/Users/maddi/AppData/Local/Programs/Python/Python312/39.py (3.12.2)

File Edit Format Run Options Window Help

```
def isValid(s: str) -> bool:
    stack = []
    mapping = {")": "(", "}": "{", "]": "["}
    for char in s:
        if char in mapping.values():
            stack.append(char)
        elif char in mapping:
            if not stack or mapping[char] != stack.pop():
                return False
    return not stack
print(isValid("()"))
```

OUTPUT :

IDLE Shell 3.12.2

File Edit Shell Debug Options Window Help

```
Python 3.12.2 (tags/v3.12.2:6abddd9, Feb  6 2024, 21:26:36) [M
SC v.1937 64 bit (AMD64)] on win32
Type "help", "copyright", "credits" or "license()" for more in
formation.
>>>
= RESTART: C:/Users/maddi/AppData/Local/Programs/Python/Python
312/39.py
True
>>>
```