main.py

Share Run

```python
def is_valid(s):
    stack = []
    mapping = {")": "(", "}": "{", "]": "["}

    for char in s:
        if char in mapping:
            top_element = stack.pop() if stack else '#'
            if mapping[char] != top_element:
                return False
        else:
            stack.append(char)

    return not stack
print(is_valid("()"))
print(is_valid("()[]{}"))
print(is_valid("(]"))

```

Output

```
True
True
False

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

```python
def max_area(height):
    max_water = 0
    for i in range(len(height)):
        for j in range(i+1, len(height)):
            max_water = max(max_water, min(height[i], height[j]) * (j - i))
    return max_water
height1 = [1, 8, 6, 2, 5, 4, 8, 3, 7]
print(max_area(height1))  # Output: 49

height2 = [1, 1]
print(max_area(height2))  # Output: 1

```

```
49
1

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

```python
def int_to_roman(num):
    val = [1000, 900, 500, 400, 100, 90, 50, 40, 10, 9, 5, 4, 1]
    syms = ["M", "CM", "D", "CD", "C", "XC", "L", "XL", "X", "IX", "V", "IV",
        "I"]
    roman_num = ''
    i = 0
    while num > 0:
        for _ in range(num // val[i]):
            roman_num += syms[i]
            num -= val[i]
        i += 1
    return roman_num
num = 354
print(f"Roman numeral for {num} is: {int_to_roman(num)}")

```

```
Roman numeral for 354 is: CCCLIV

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

```python
def roman_to_int(s: str) -> int:
    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 prev_value < value:
            result -= 2 * prev_value
        prev_value = value

    return result
input_roman_numeral = "III"
output_integer = roman_to_int(input_roman_numeral)
print(f"Input: {input_roman_numeral}")
print(f"Output: {output_integer}")

```

```
Input: III
Output: 3

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

main.py

Share Run

Output

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

    prefix = ""
    for i in range(len(min(strs))):
        if all(s[i] == strs[0][i] for s in strs):
            prefix += strs[0][i]
        else:
            break

    return prefix
strs = ["flower", "flow", "flight"]
print(longest_common_prefix(strs))  # Output: "fl"

```

```
fl

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

```python
def three_sum(nums):
    nums.sort()
    result = []
    for i in range(len(nums) - 2):
        if i > 0 and nums[i] == nums[i - 1]:
            continue
        left, right = i + 1, len(nums) - 1
        while left < right:
            total = nums[i] + nums[left] + nums[right]
            if total < 0:
                left += 1
            elif total > 0:
                right -= 1
            else:
                result.append([nums[i], 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
nums = [-1, 0, 1, 2, -1, -4]
print(three_sum(nums))

```

```
[[-1, -1, 2], [-1, 0, 1]]

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

```python
def fourSum(nums, target):
    nums.sort()
    n = len(nums)
    result = []

    for i in range(n - 3):
        if i > 0 and nums[i] == nums[i - 1]:
            continue

        for j in range(i + 1, n - 2):
            if j > i + 1 and nums[j] == nums[j - 1]:
                continue

            left, right = j + 1, n - 1
            while left < right:
                total = nums[i] + nums[j] + nums[left] + nums[right]
                if total == target:
                    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
                elif total < target:
                    left += 1
                else:
                    right -= 1
```

```
[[-2, -1, 1, 2], [-2, 0, 0, 2], [-1, 0, 0, 1]]

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