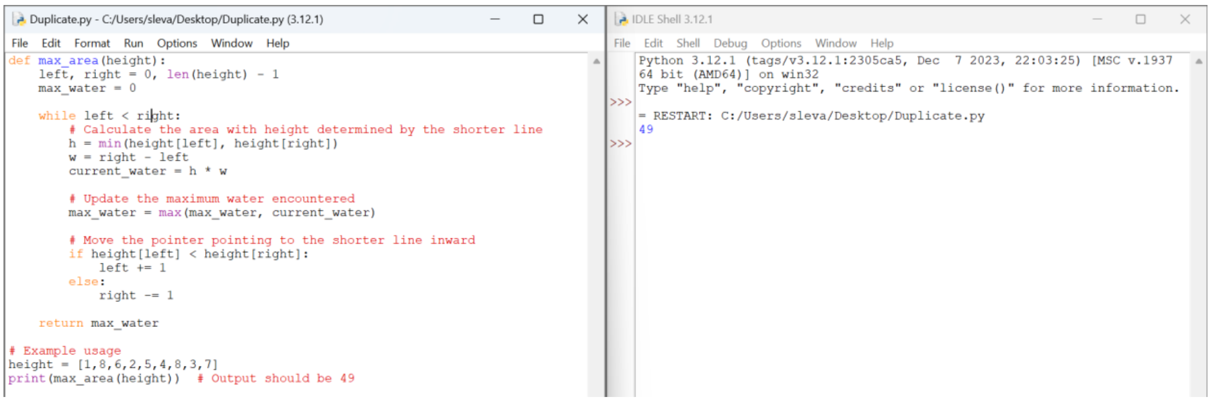
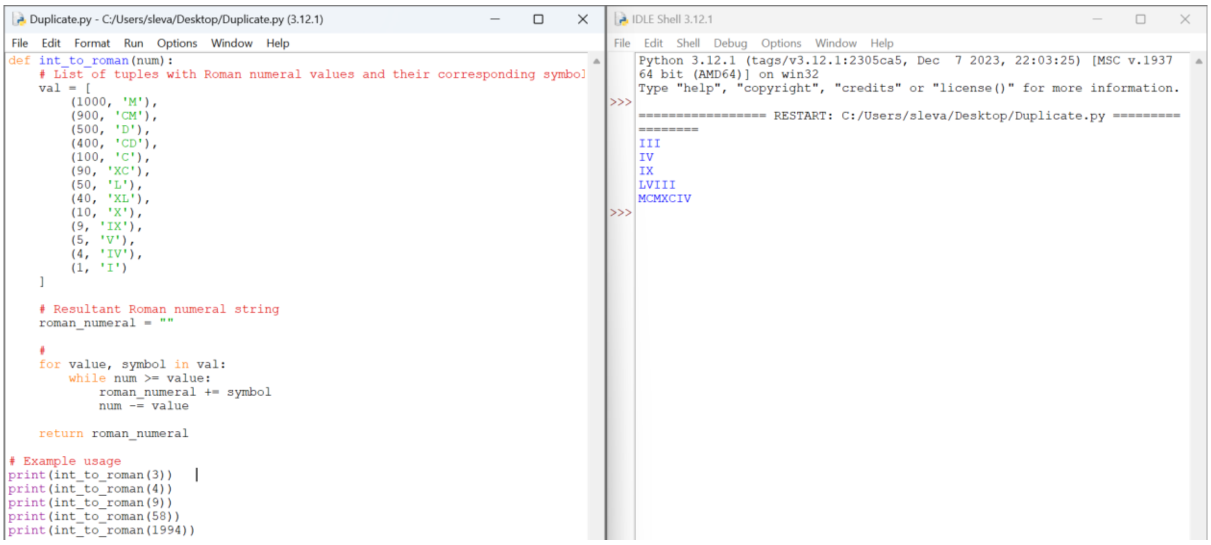
1.Container With Most Water



```python
def max_area(height):
    left, right = 0, len(height) - 1
    max_water = 0

    while left < right:
        # Calculate the area with height determined by the shorter line
        h = min(height[left], height[right])
        w = right - left
        current_water = h * w

        # Update the maximum water encountered
        max_water = max(max_water, current_water)

        # Move the pointer pointing to the shorter line inward
        if height[left] < height[right]:
            left += 1
        else:
            right -= 1

    return max_water

# Example usage
height = [1,8,6,2,5,4,8,3,7]
print(max_area(height))  # Output should be 49
```

```
Python 3.12.1 (tags/v3.12.1:2305ca5, Dec  7 2023, 22:03:25) [MSC v.1937
64 bit (AMD64)] on win32
Type "help", "copyright", "credits" or "license()" for more information.
>>>
= RESTART: C:/Users/sleva/Desktop/Duplicate.py
49
>>>
```

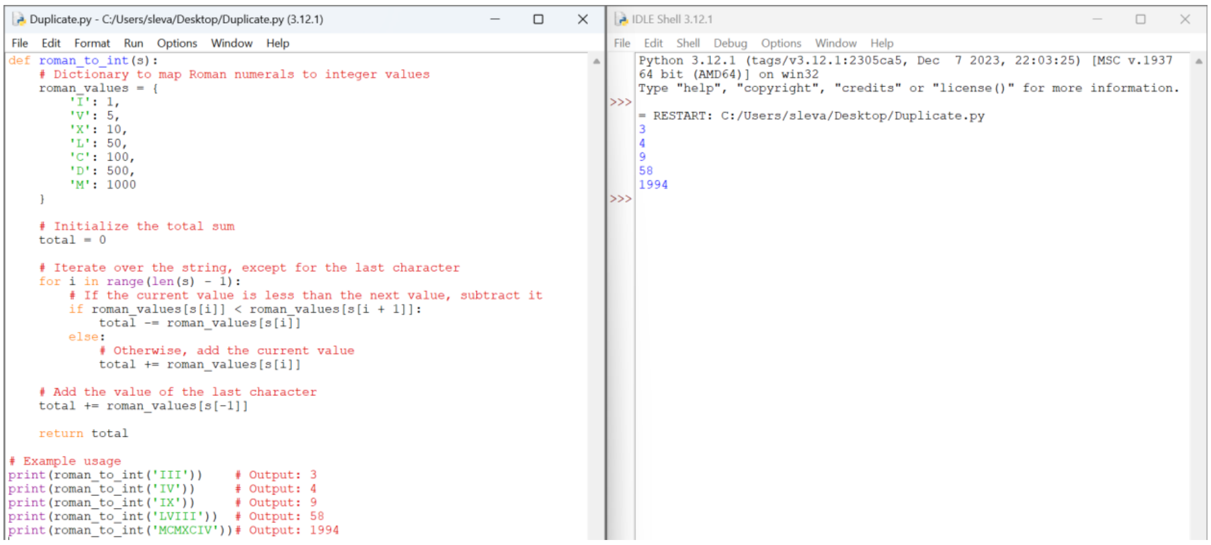
2. Integer to Roman



```python
def int_to_roman(num):
    # List of tuples with Roman numeral values and their corresponding symbol
    val = [
        (1000, 'M'),
        (900, 'CM'),
        (500, 'D'),
        (400, 'CD'),
        (100, 'C'),
        (90, 'XC'),
        (50, 'L'),
        (40, 'XL'),
        (10, 'X'),
        (9, 'IX'),
        (5, 'V'),
        (4, 'IV'),
        (1, 'I')
    ]

    # Resultant Roman numeral string
    roman_numeral = ""

    #
    for value, symbol in val:
        while num >= value:
            roman_numeral += symbol
            num -= value

    return roman_numeral

# Example usage
print(int_to_roman(3))
print(int_to_roman(4))
print(int_to_roman(9))
print(int_to_roman(58))
print(int_to_roman(1994))
```

```
Python 3.12.1 (tags/v3.12.1:2305ca5, Dec  7 2023, 22:03:25) [MSC v.1937
64 bit (AMD64)] on win32
Type "help", "copyright", "credits" or "license()" for more information.
>>>
================ RESTART: C:/Users/sleva/Desktop/Duplicate.py ========
========
III
IV
IX
LVIII
MCMXCIV
>>>
```

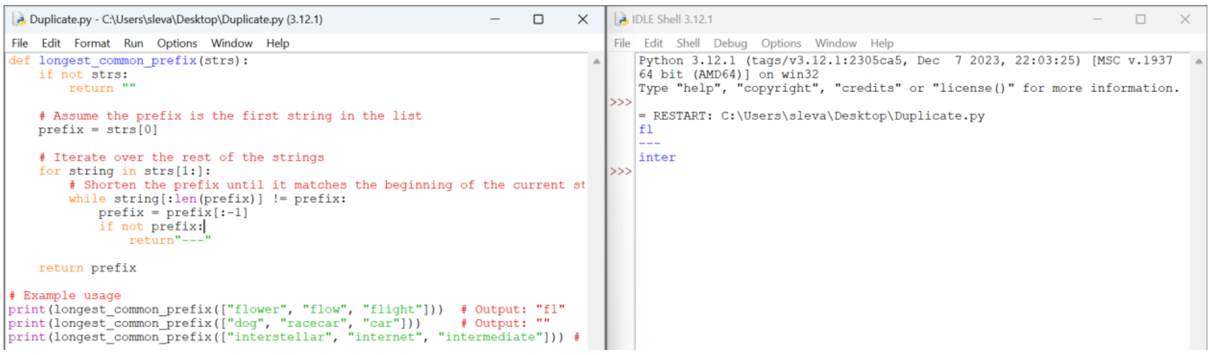
3. Roman to Integer



```python
def roman_to_int(s):
    # Dictionary to map Roman numerals to integer values
    roman_values = {
        'I': 1,
        'V': 5,
        'X': 10,
        'L': 50,
        'C': 100,
        'D': 500,
        'M': 1000
    }

    # Initialize the total sum
    total = 0

    # Iterate over the string, except for the last character
    for i in range(len(s) - 1):
        # If the current value is less than the next value, subtract it
        if roman_values[s[i]] < roman_values[s[i + 1]]:
            total -= roman_values[s[i]]
        else:
            # Otherwise, add the current value
            total += roman_values[s[i]]

    # Add the value of the last character
    total += roman_values[s[-1]]

    return total

# Example usage
print(roman_to_int('III'))      # Output: 3
print(roman_to_int('IV'))       # Output: 4
print(roman_to_int('IX'))       # Output: 9
print(roman_to_int('LVIII'))    # Output: 58
print(roman_to_int('MCMXCIV'))# Output: 1994
```

```
Python 3.12.1 (tags/v3.12.1:2305ca5, Dec  7 2023, 22:03:25) [MSC v.1937
64 bit (AMD64)] on win32
Type "help", "copyright", "credits" or "license()" for more information.
>>>
= RESTART: C:/Users/sleva/Desktop/Duplicate.py
3
4
9
58
1994
>>>
```

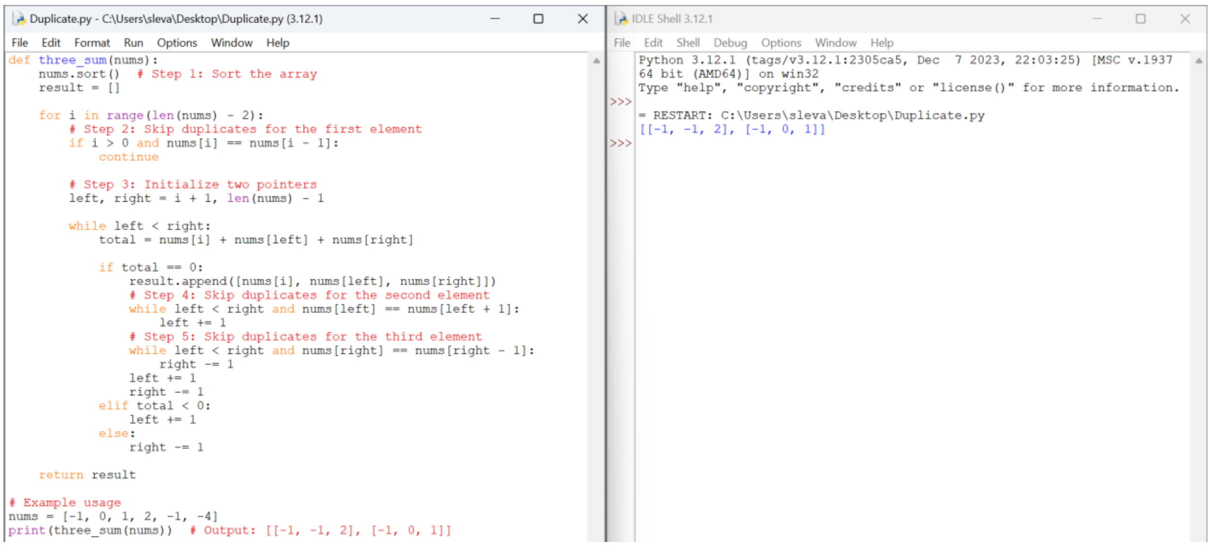
4. Longest Common Prefix


Duplicate.py - C:\Users\sleva\Desktop\Duplicate.py (3.12.1)
File Edit Format Run Options Window Help
IDLE Shell 3.12.1
File Edit Shell Debug Options Window Help

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

    # Assume the prefix is the first string in the list
    prefix = strs[0]

    # Iterate over the rest of the strings
    for string in strs[1:]:
        # Shorten the prefix until it matches the beginning of the current st
        while string[:len(prefix)] != prefix:
            prefix = prefix[:-1]
            if not prefix:
                return"---"

    return prefix

# Example usage
print(longest_common_prefix(["flower", "flow", "flight"]))  # Output: "fl"
print(longest_common_prefix(["dog", "racecar", "car"]))     # Output: ""
print(longest_common_prefix(["interstellar", "internet", "intermediate"])) #
```

```
Python 3.12.1 (tags/v3.12.1:2305ca5, Dec  7 2023, 22:03:25) [MSC v.1937
64 bit (AMD64)] on win32
Type "help", "copyright", "credits" or "license()" for more information.
>>>
= RESTART: C:\Users\sleva\Desktop\Duplicate.py
fl
---
inter
>>>
```

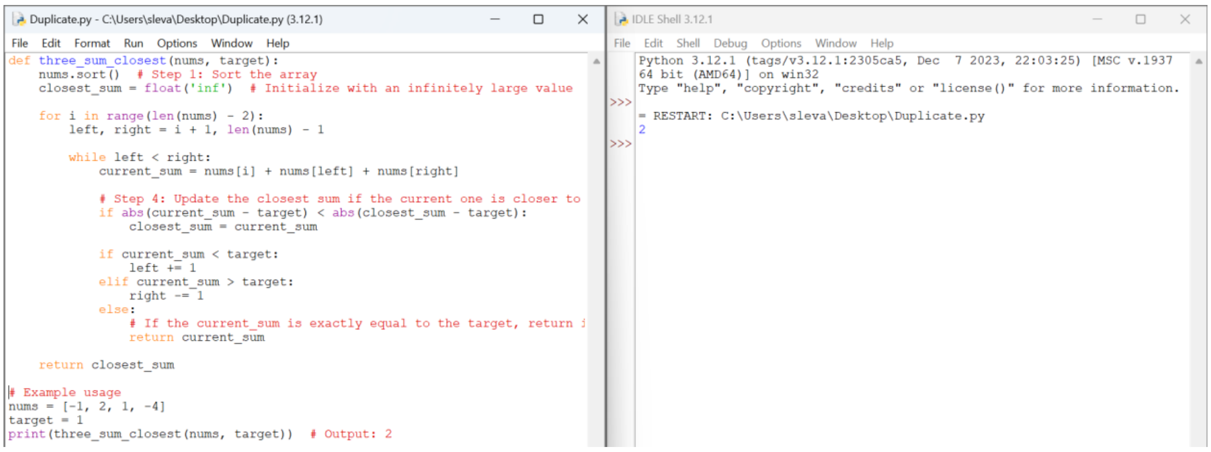
5. 3Sum


Duplicate.py - C:\Users\sleva\Desktop\Duplicate.py (3.12.1)
File Edit Format Run Options Window Help
IDLE Shell 3.12.1
File Edit Shell Debug Options Window Help

```python
def three_sum(nums):
    nums.sort()  # Step 1: Sort the array
    result = []

    for i in range(len(nums) - 2):
        # Step 2: Skip duplicates for the first element
        if i > 0 and nums[i] == nums[i - 1]:
            continue

        # Step 3: Initialize two pointers
        left, right = i + 1, len(nums) - 1

        while left < right:
            total = nums[i] + nums[left] + nums[right]

            if total == 0:
                result.append([nums[i], nums[left], nums[right]])
                # Step 4: Skip duplicates for the second element
                while left < right and nums[left] == nums[left + 1]:
                    left += 1
                # Step 5: Skip duplicates for the third element
                while left < right and nums[right] == nums[right - 1]:
                    right -= 1
                left += 1
                right -= 1
            elif total < 0:
                left += 1
            else:
                right -= 1

    return result

# Example usage
nums = [-1, 0, 1, 2, -1, -4]
print(three_sum(nums))  # Output: [[-1, -1, 2], [-1, 0, 1]]
```

```
Python 3.12.1 (tags/v3.12.1:2305ca5, Dec  7 2023, 22:03:25) [MSC v.1937
64 bit (AMD64)] on win32
Type "help", "copyright", "credits" or "license()" for more information.
>>>
= RESTART: C:\Users\sleva\Desktop\Duplicate.py
[[-1, -1, 2], [-1, 0, 1]]
>>>
```

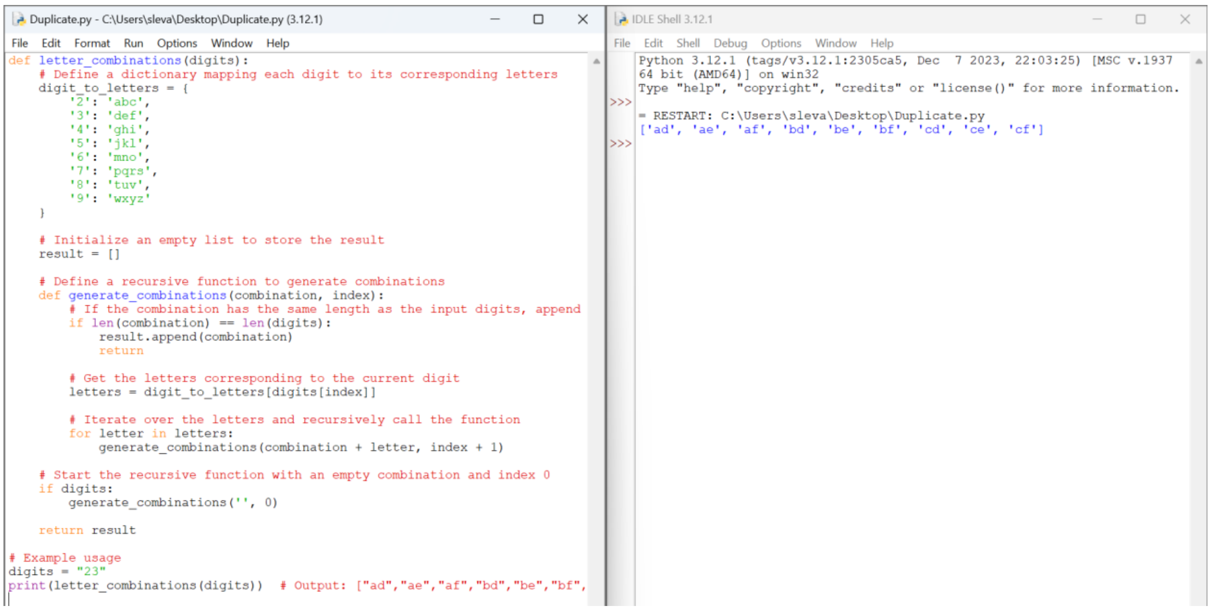
6. 3Sum Closest


Duplicate.py - C:\Users\sleva\Desktop\Duplicate.py (3.12.1)
File Edit Format Run Options Window Help
IDLE Shell 3.12.1
File Edit Shell Debug Options Window Help

```python
def three_sum_closest(nums, target):
    nums.sort()  # Step 1: Sort the array
    closest_sum = float('inf')  # Initialize with an infinitely large value

    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]

            # Step 4: Update the closest sum if the current one is closer to
            if abs(current_sum - target) < abs(closest_sum - target):
                closest_sum = current_sum

            if current_sum < target:
                left += 1
            elif current_sum > target:
                right -= 1
            else:
                # If the current_sum is exactly equal to the target, return i
                return current_sum

    return closest_sum

# Example usage
nums = [-1, 2, 1, -4]
target = 1
print(three_sum_closest(nums, target))  # Output: 2
```

```
Python 3.12.1 (tags/v3.12.1:2305ca5, Dec  7 2023, 22:03:25) [MSC v.1937
64 bit (AMD64)] on win32
Type "help", "copyright", "credits" or "license()" for more information.
>>>
= RESTART: C:\Users\sleva\Desktop\Duplicate.py
2
>>>
```

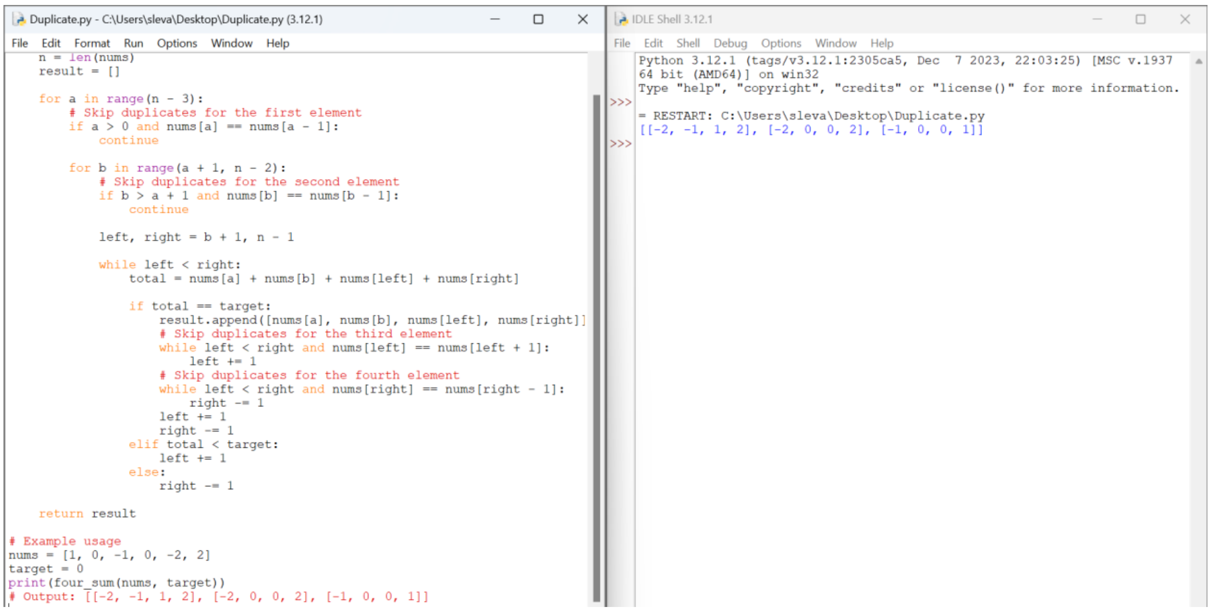
7. Letter Combinations of a Phone Number


Duplicate.py - C:\Users\sleva\Desktop\Duplicate.py (3.12.1)
File Edit Format Run Options Window Help
IDLE Shell 3.12.1
File Edit Shell Debug Options Window Help

```python
def letter_combinations(digits):
    # Define a dictionary mapping each digit to its corresponding letters
    digit_to_letters = {
        '2': 'abc',
        '3': 'def',
        '4': 'ghi',
        '5': 'jkl',
        '6': 'mno',
        '7': 'pqrs',
        '8': 'tuv',
        '9': 'wxyz'
    }

    # Initialize an empty list to store the result
    result = []

    # Define a recursive function to generate combinations
    def generate_combinations(combination, index):
        # If the combination has the same length as the input digits, append
        if len(combination) == len(digits):
            result.append(combination)
            return

        # Get the letters corresponding to the current digit
        letters = digit_to_letters[digits[index]]

        # Iterate over the letters and recursively call the function
        for letter in letters:
            generate_combinations(combination + letter, index + 1)

    # Start the recursive function with an empty combination and index 0
    if digits:
        generate_combinations('', 0)

    return result

# Example usage
digits = "23"
print(letter_combinations(digits))  # Output: ["ad","ae","af","bd","be","bf",
```

```
Python 3.12.1 (tags/v3.12.1:2305ca5, Dec  7 2023, 22:03:25) [MSC v.1937
64 bit (AMD64)] on win32
Type "help", "copyright", "credits" or "license()" for more information.
>>>
= RESTART: C:\Users\sleva\Desktop\Duplicate.py
['ad', 'ae', 'af', 'bd', 'be', 'bf', 'cd', 'ce', 'cf']
>>>
```

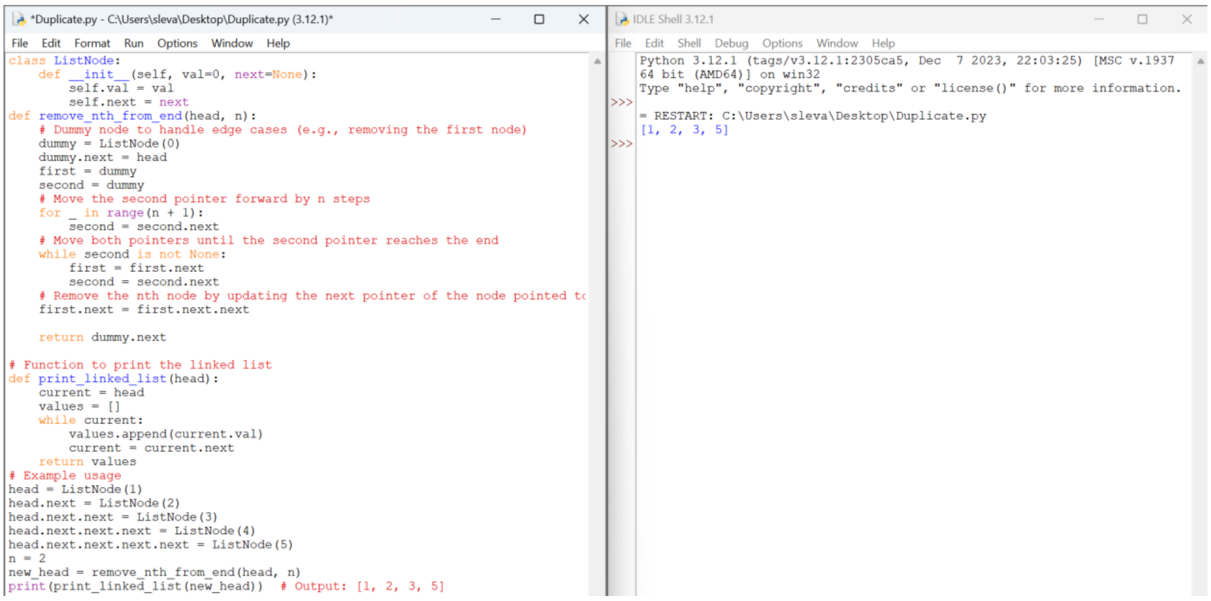
8. 4Sum


Duplicate.py - C:\Users\sleva\Desktop\Duplicate.py (3.12.1)
File Edit Format Run Options Window Help
IDLE Shell 3.12.1
File Edit Shell Debug Options Window Help

```python
    n = len(nums)
    result = []

    for a in range(n - 3):
        # Skip duplicates for the first element
        if a > 0 and nums[a] == nums[a - 1]:
            continue

        for b in range(a + 1, n - 2):
            # Skip duplicates for the second element
            if b > a + 1 and nums[b] == nums[b - 1]:
                continue

            left, right = b + 1, n - 1

            while left < right:
                total = nums[a] + nums[b] + nums[left] + nums[right]

                if total == target:
                    result.append([nums[a], nums[b], nums[left], nums[right]]
                    # Skip duplicates for the third element
                    while left < right and nums[left] == nums[left + 1]:
                        left += 1
                    # Skip duplicates for the fourth element
                    while left < right and nums[right] == nums[right - 1]:
                        right -= 1
                    left += 1
                    right -= 1
                elif total < target:
                    left += 1
                else:
                    right -= 1

    return result

# Example usage
nums = [1, 0, -1, 0, -2, 2]
target = 0
print(four_sum(nums, target))
# Output: [[-2, -1, 1, 2], [-2, 0, 0, 2], [-1, 0, 0, 1]]
```

```
Python 3.12.1 (tags/v3.12.1:2305ca5, Dec  7 2023, 22:03:25) [MSC v.1937
64 bit (AMD64)] on win32
Type "help", "copyright", "credits" or "license()" for more information.
>>>
= RESTART: C:\Users\sleva\Desktop\Duplicate.py
[[-2, -1, 1, 2], [-2, 0, 0, 2], [-1, 0, 0, 1]]
>>>
```

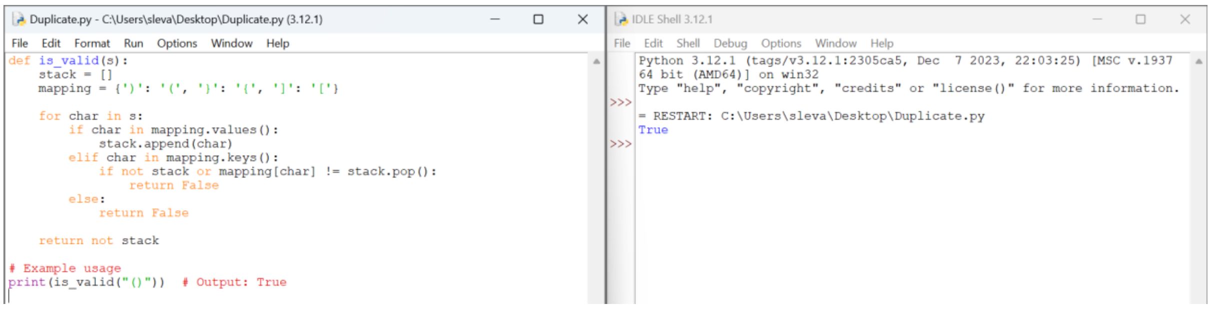
9. Remove Nth Node From End of List


Duplicate.py - C:\Users\sleva\Desktop\Duplicate.py (3.12.1)
File Edit Format Run Options Window Help
IDLE Shell 3.12.1
File Edit Shell Debug Options Window Help

```python
class ListNode:
    def __init__(self, val=0, next=None):
        self.val = val
        self.next = next
def remove_nth_from_end(head, n):
    # Dummy node to handle edge cases (e.g., removing the first node)
    dummy = ListNode(0)
    dummy.next = head
    first = dummy
    second = dummy
    # Move the second pointer forward by n steps
    for _ in range(n + 1):
        second = second.next
    # Move both pointers until the second pointer reaches the end
    while second is not None:
        first = first.next
        second = second.next
    # Remove the nth node by updating the next pointer of the node pointed to
    first.next = first.next.next

    return dummy.next

# Function to print the linked list
def print_linked_list(head):
    current = head
    values = []
    while current:
        values.append(current.val)
        current = current.next
    return values
# Example usage
head = ListNode(1)
head.next = ListNode(2)
head.next.next = ListNode(3)
head.next.next.next = ListNode(4)
head.next.next.next.next = ListNode(5)
n = 2
new_head = remove_nth_from_end(head, n)
print(print_linked_list(new_head))  # Output: [1, 2, 3, 5]
```

```
Python 3.12.1 (tags/v3.12.1:2305ca5, Dec  7 2023, 22:03:25) [MSC v.1937
64 bit (AMD64)] on win32
Type "help", "copyright", "credits" or "license()" for more information.
>>>
= RESTART: C:\Users\sleva\Desktop\Duplicate.py
[1, 2, 3, 5]
>>>
```

10. Valid Parentheses


Duplicate.py - C:\Users\sleva\Desktop\Duplicate.py (3.12.1)
File Edit Format Run Options Window Help
IDLE Shell 3.12.1
File Edit Shell Debug Options Window Help

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

    for char in s:
        if char in mapping.values():
            stack.append(char)
        elif char in mapping.keys():
            if not stack or mapping[char] != stack.pop():
                return False
        else:
            return False

    return not stack

# Example usage
print(is_valid("()"))  # Output: True
```

```
Python 3.12.1 (tags/v3.12.1:2305ca5, Dec  7 2023, 22:03:25) [MSC v.1937
64 bit (AMD64)] on win32
Type "help", "copyright", "credits" or "license()" for more information.
>>>
= RESTART: C:\Users\sleva\Desktop\Duplicate.py
True
>>>
```