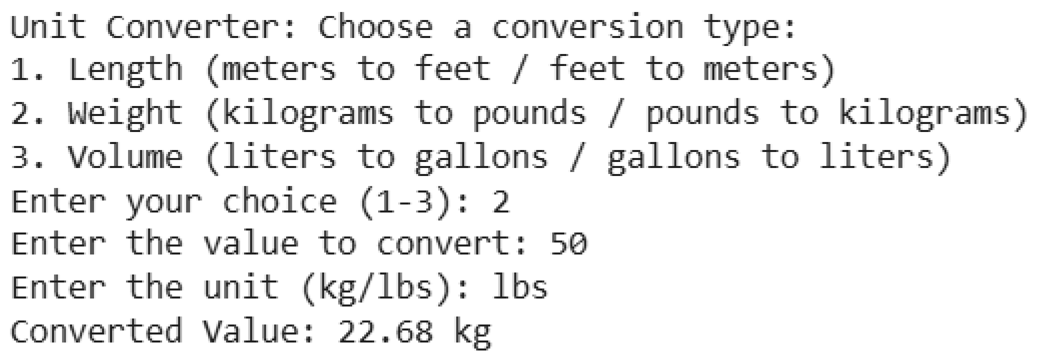
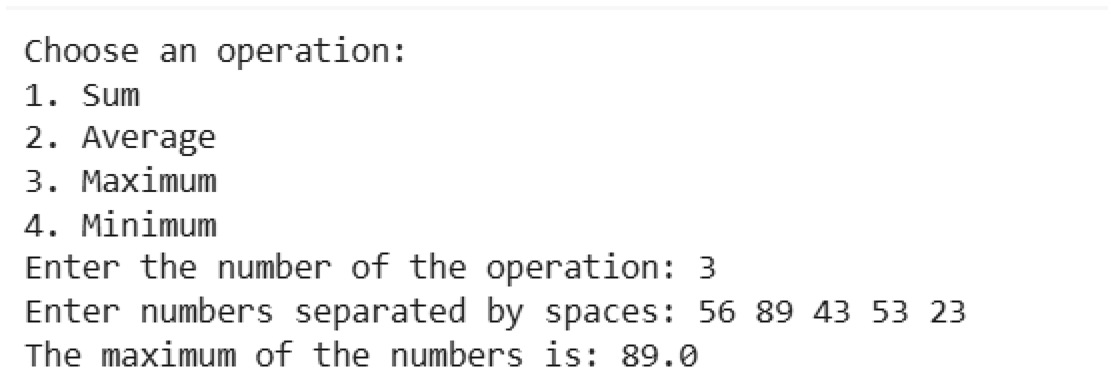
Task 1



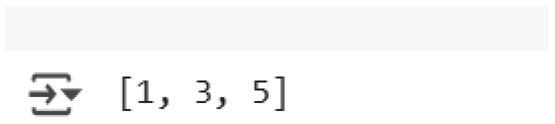
```
Unit Converter: Choose a conversion type:
1. Length (meters to feet / feet to meters)
2. Weight (kilograms to pounds / pounds to kilograms)
3. Volume (liters to gallons / gallons to liters)
Enter your choice (1-3): 2
Enter the value to convert: 50
Enter the unit (kg/lbs): lbs
Converted Value: 22.68 kg
```

Task 2



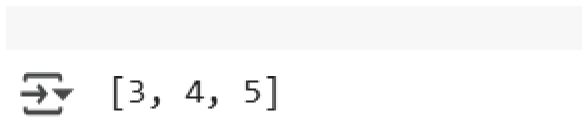
```
Choose an operation:
1. Sum
2. Average
3. Maximum
4. Minimum
Enter the number of the operation: 3
Enter numbers separated by spaces: 56 89 43 53 23
The maximum of the numbers is: 89.0
```

Task 3



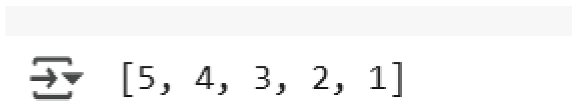
```
[1, 3, 5]
```

Task 4



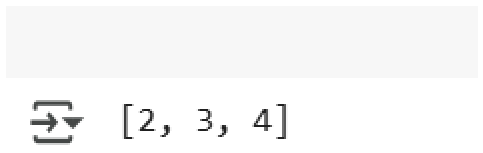
```
[3, 4, 5]
```

Task 5



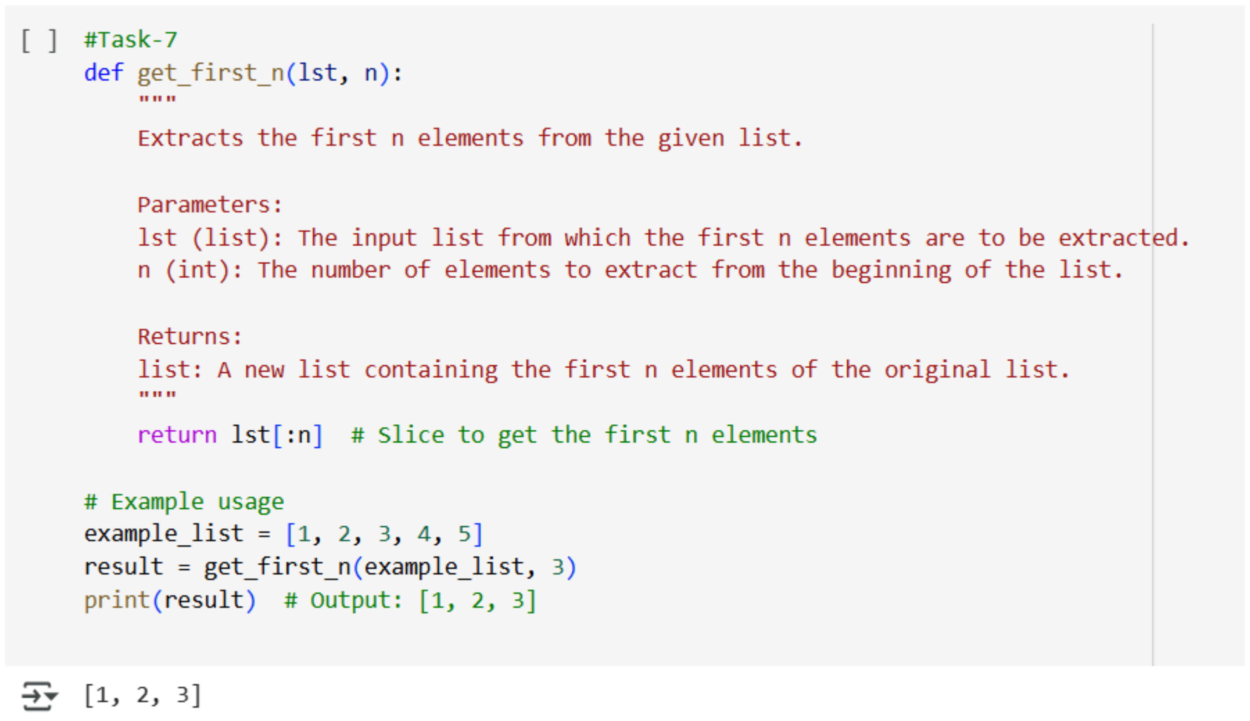
```
[5, 4, 3, 2, 1]
```

Task 6



```
[2, 3, 4]
```

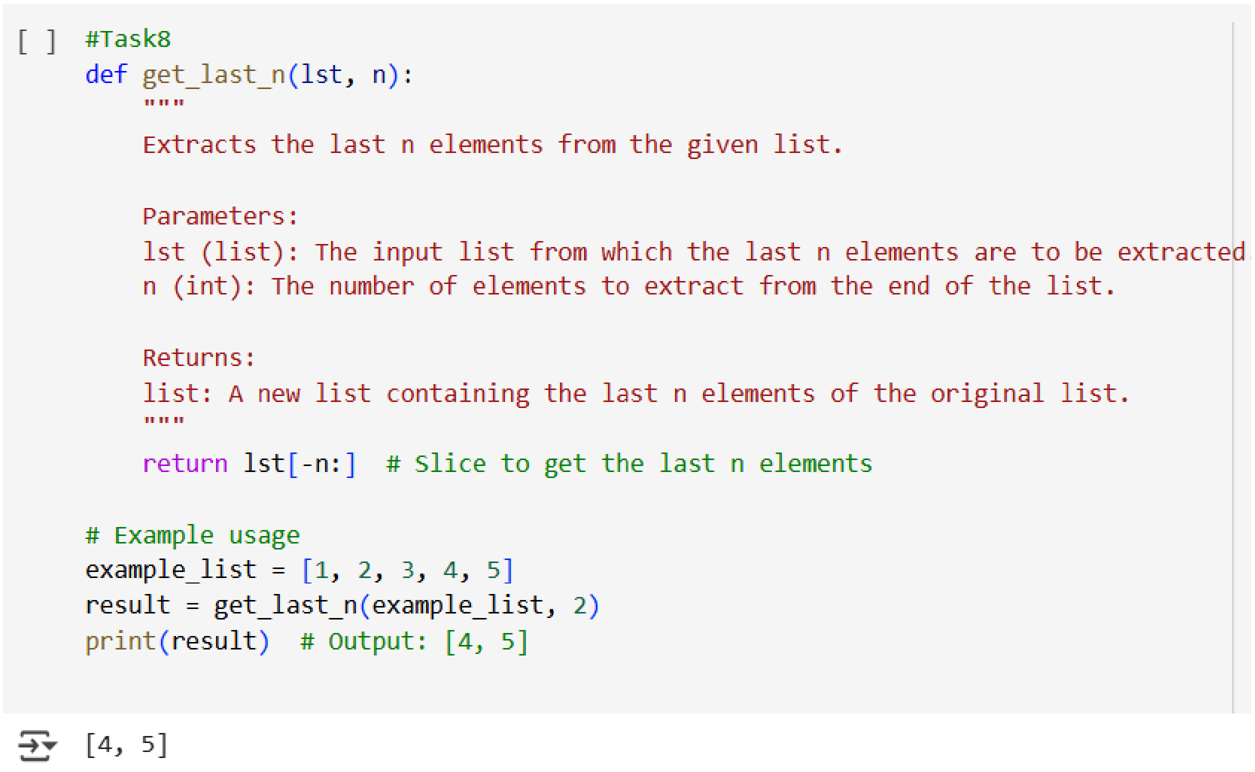
Task 7



```
[ ] #Task-7
    def get_first_n(lst, n):
        """
        Extracts the first n elements from the given list.

        Parameters:
        lst (list): The input list from which the first n elements are to be extracted.
        n (int): The number of elements to extract from the beginning of the list.

        Returns:
        list: A new list containing the first n elements of the original list.
        """
        return lst[:n]  # Slice to get the first n elements

    # Example usage
    example_list = [1, 2, 3, 4, 5]
    result = get_first_n(example_list, 3)
    print(result)  # Output: [1, 2, 3]
```

```
[1, 2, 3]
```

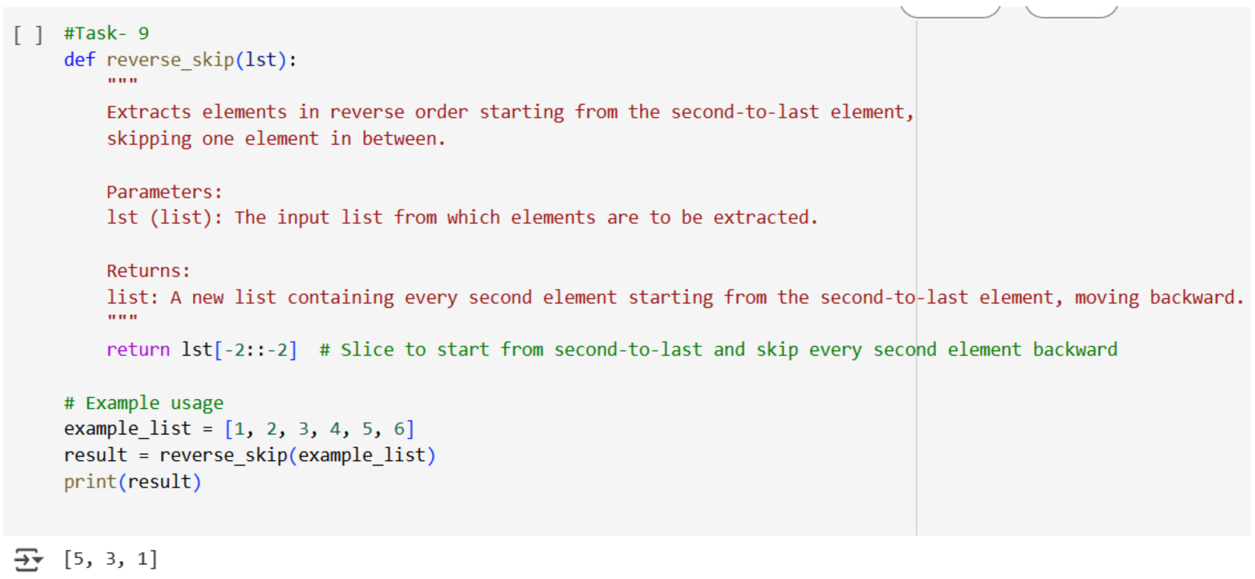
Task 8



```
[ ] #Task8
    def get_last_n(lst, n):
        """
        Extracts the last n elements from the given list.

        Parameters:
        lst (list): The input list from which the last n elements are to be extracted
        n (int): The number of elements to extract from the end of the list.

        Returns:
        list: A new list containing the last n elements of the original list.
        """
        return lst[-n:]  # Slice to get the last n elements

    # Example usage
    example_list = [1, 2, 3, 4, 5]
    result = get_last_n(example_list, 2)
    print(result)  # Output: [4, 5]
```

```
[4, 5]
```

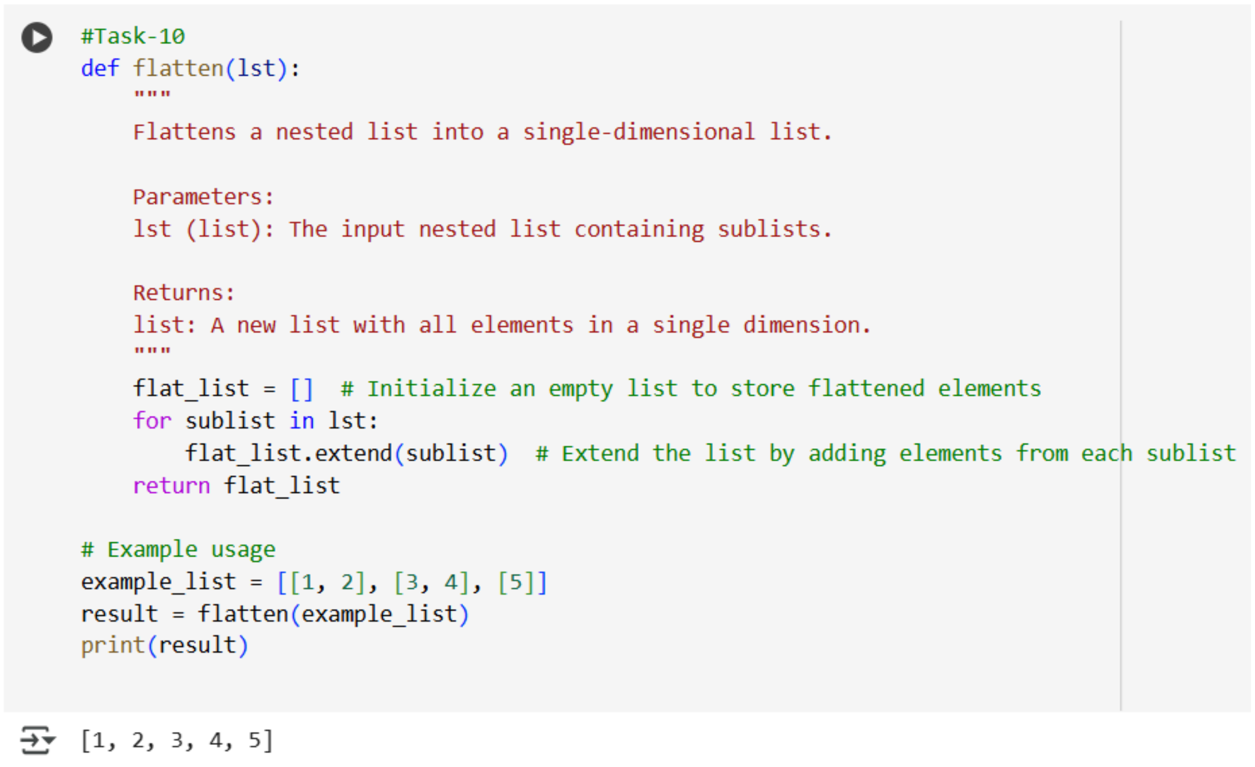
Task 9



```
[ ] #Task- 9
    def reverse_skip(lst):
        """
        Extracts elements in reverse order starting from the second-to-last element,
        skipping one element in between.

        Parameters:
        lst (list): The input list from which elements are to be extracted.

        Returns:
        list: A new list containing every second element starting from the second-to-last element, moving backward.
        """
        return lst[-2::-2]  # Slice to start from second-to-last and skip every second element backward

    # Example usage
    example_list = [1, 2, 3, 4, 5, 6]
    result = reverse_skip(example_list)
    print(result)
```

```
[5, 3, 1]
```

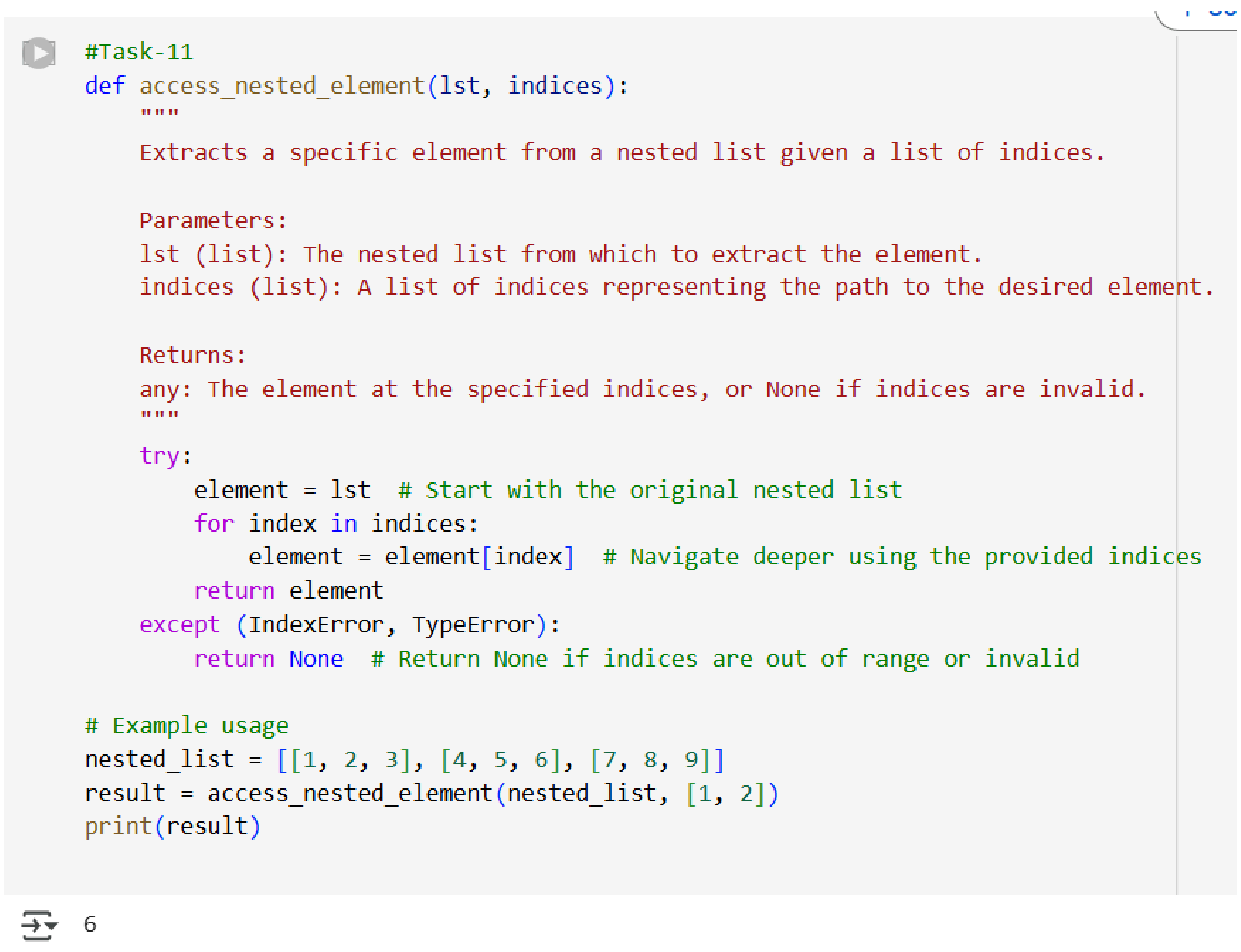
Task 10



```python
#Task-10
def flatten(lst):
    """
    Flattens a nested list into a single-dimensional list.

    Parameters:
    lst (list): The input nested list containing sublists.

    Returns:
    list: A new list with all elements in a single dimension.
    """
    flat_list = []  # Initialize an empty list to store flattened elements
    for sublist in lst:
        flat_list.extend(sublist)  # Extend the list by adding elements from each sublist
    return flat_list

# Example usage
example_list = [[1, 2], [3, 4], [5]]
result = flatten(example_list)
print(result)
```

```
[1, 2, 3, 4, 5]
```

Task 11



```python
#Task-11
def access_nested_element(lst, indices):
    """
    Extracts a specific element from a nested list given a list of indices.

    Parameters:
    lst (list): The nested list from which to extract the element.
    indices (list): A list of indices representing the path to the desired element.

    Returns:
    any: The element at the specified indices, or None if indices are invalid.
    """
    try:
        element = lst  # Start with the original nested list
        for index in indices:
            element = element[index]  # Navigate deeper using the provided indices
        return element
    except (IndexError, TypeError):
        return None  # Return None if indices are out of range or invalid

# Example usage
nested_list = [[1, 2, 3], [4, 5, 6], [7, 8, 9]]
result = access_nested_element(nested_list, [1, 2])
print(result)
```

```
6
```