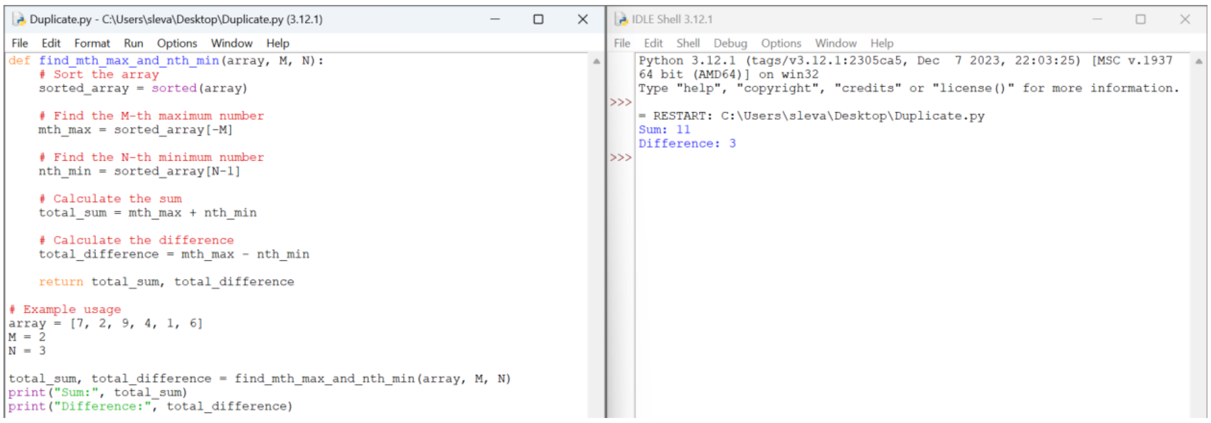
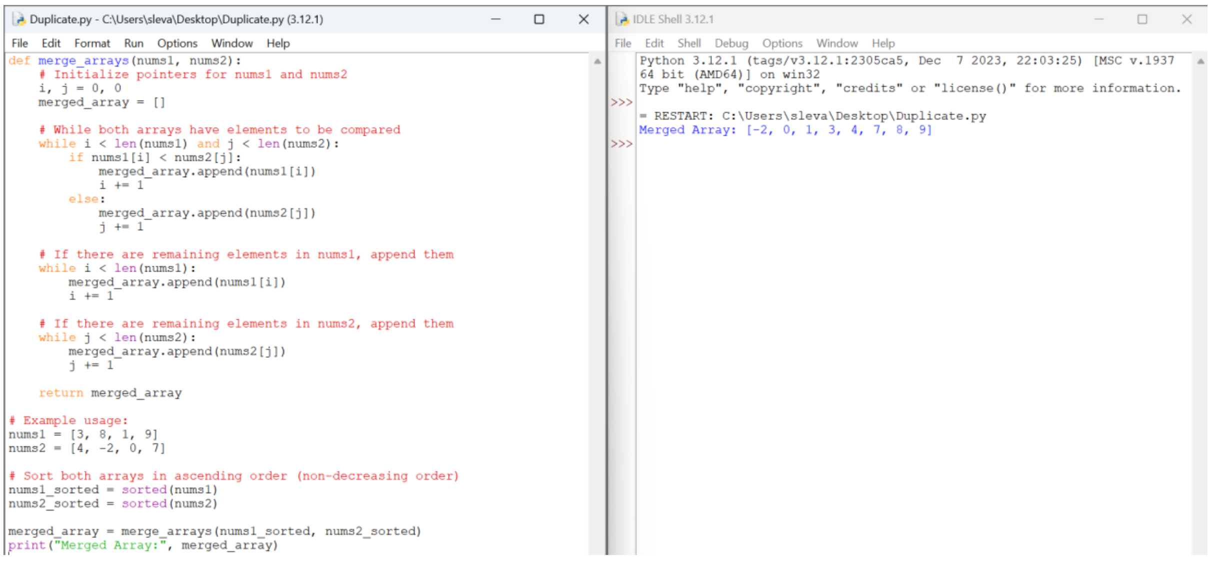
1. Find Mth maximum and Nth minimum Number in an array



```python
def find_mth_max_and_nth_min(array, M, N):
    # Sort the array
    sorted_array = sorted(array)

    # Find the M-th maximum number
    mth_max = sorted_array[-M]

    # Find the N-th minimum number
    nth_min = sorted_array[N-1]

    # Calculate the sum
    total_sum = mth_max + nth_min

    # Calculate the difference
    total_difference = mth_max - nth_min

    return total_sum, total_difference

# Example usage
array = [7, 2, 9, 4, 1, 6]
M = 2
N = 3

total_sum, total_difference = find_mth_max_and_nth_min(array, M, N)
print("Sum:", total_sum)
print("Difference:", total_difference)
```

```
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
Sum: 11
Difference: 3
>>>
```

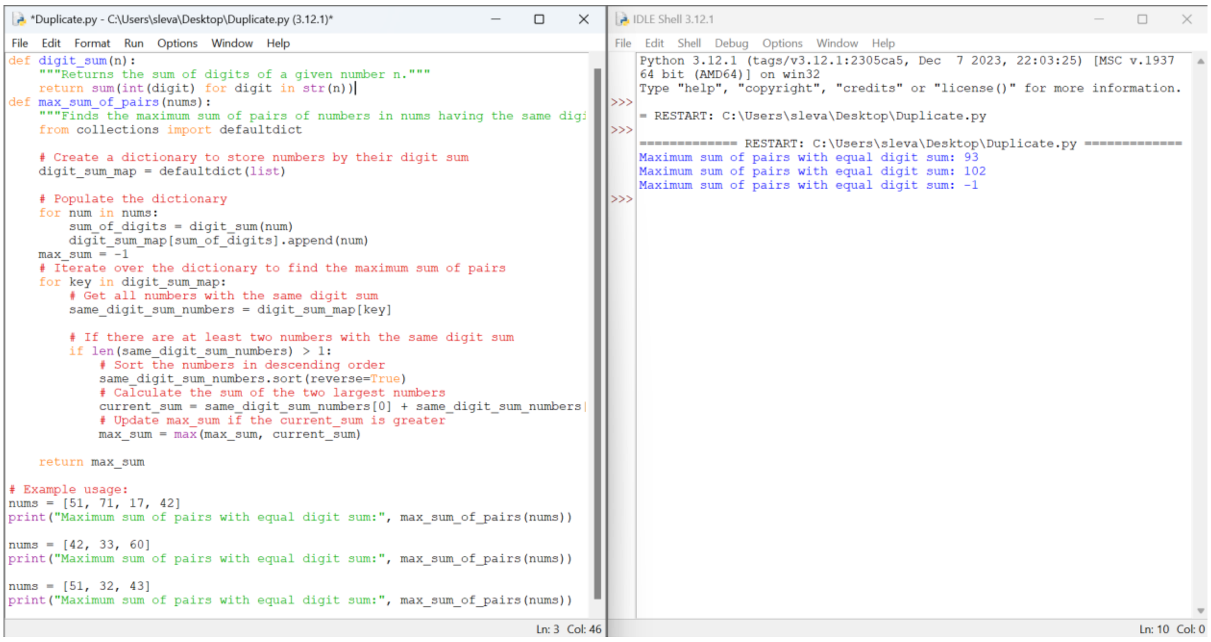
2. Merge Sort



```python
def merge_arrays(nums1, nums2):
    # Initialize pointers for nums1 and nums2
    i, j = 0, 0
    merged_array = []

    # While both arrays have elements to be compared
    while i < len(nums1) and j < len(nums2):
        if nums1[i] < nums2[j]:
            merged_array.append(nums1[i])
            i += 1
        else:
            merged_array.append(nums2[j])
            j += 1

    # If there are remaining elements in nums1, append them
    while i < len(nums1):
        merged_array.append(nums1[i])
        i += 1

    # If there are remaining elements in nums2, append them
    while j < len(nums2):
        merged_array.append(nums2[j])
        j += 1

    return merged_array

# Example usage:
nums1 = [3, 8, 1, 9]
nums2 = [4, -2, 0, 7]

# Sort both arrays in ascending order (non-decreasing order)
nums1_sorted = sorted(nums1)
nums2_sorted = sorted(nums2)

merged_array = merge_arrays(nums1_sorted, nums2_sorted)
print("Merged Array:", merged_array)
```

```
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
Merged Array: [-2, 0, 1, 3, 4, 7, 8, 9]
>>>
```

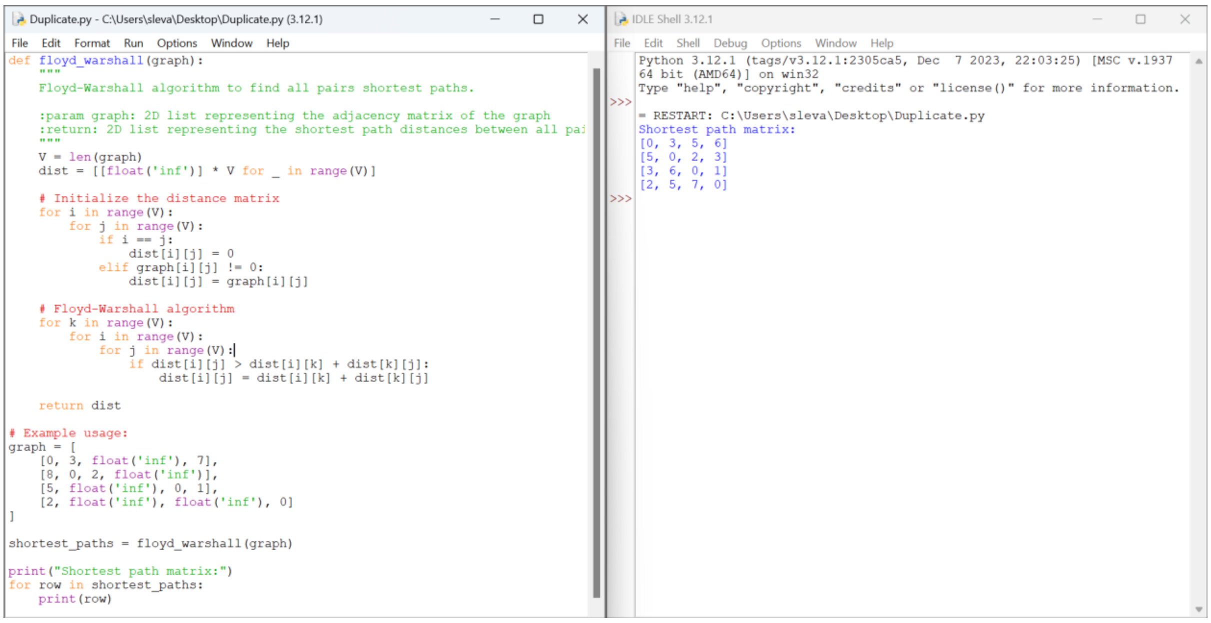
3. Maximum sum of digits with equal pairs



```python
def digit_sum(n):
    """Returns the sum of digits of a given number n."""
    return sum(int(digit) for digit in str(n))
def max_sum_of_pairs(nums):
    """Finds the maximum sum of pairs of numbers in nums having the same digi
    from collections import defaultdict

    # Create a dictionary to store numbers by their digit sum
    digit_sum_map = defaultdict(list)

    # Populate the dictionary
    for num in nums:
        sum_of_digits = digit_sum(num)
        digit_sum_map[sum_of_digits].append(num)
    max_sum = -1
    # Iterate over the dictionary to find the maximum sum of pairs
    for key in digit_sum_map:
        # Get all numbers with the same digit sum
        same_digit_sum_numbers = digit_sum_map[key]

        # If there are at least two numbers with the same digit sum
        if len(same_digit_sum_numbers) > 1:
            # Sort the numbers in descending order
            same_digit_sum_numbers.sort(reverse=True)
            # Calculate the sum of the two largest numbers
            current_sum = same_digit_sum_numbers[0] + same_digit_sum_numbers
            # Update max_sum if the current_sum is greater
            max_sum = max(max_sum, current_sum)

    return max_sum

# Example usage:
nums = [51, 71, 17, 42]
print("Maximum sum of pairs with equal digit sum:", max_sum_of_pairs(nums))

nums = [42, 33, 60]
print("Maximum sum of pairs with equal digit sum:", max_sum_of_pairs(nums))

nums = [51, 32, 43]
print("Maximum sum of pairs with equal digit sum:", max_sum_of_pairs(nums))
```

```
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
>>>
============= RESTART: C:\Users\sleva\Desktop\Duplicate.py =============
Maximum sum of pairs with equal digit sum: 93
Maximum sum of pairs with equal digit sum: 102
Maximum sum of pairs with equal digit sum: -1
>>>
```

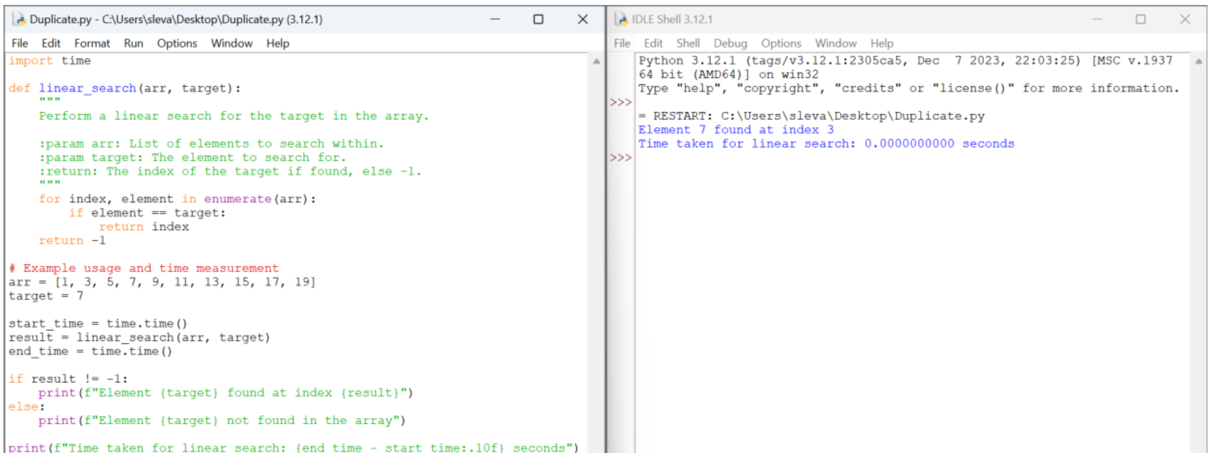
4. Shortest path using Floyd’s technique


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 floyd_warshall(graph):
    """
    Floyd-Warshall algorithm to find all pairs shortest paths.

    :param graph: 2D list representing the adjacency matrix of the graph
    :return: 2D list representing the shortest path distances between all pai
    """
    V = len(graph)
    dist = [[float('inf')] * V for _ in range(V)]

    # Initialize the distance matrix
    for i in range(V):
        for j in range(V):
            if i == j:
                dist[i][j] = 0
            elif graph[i][j] != 0:
                dist[i][j] = graph[i][j]

    # Floyd-Warshall algorithm
    for k in range(V):
        for i in range(V):
            for j in range(V):
                if dist[i][j] > dist[i][k] + dist[k][j]:
                    dist[i][j] = dist[i][k] + dist[k][j]

    return dist

# Example usage:
graph = [
    [0, 3, float('inf'), 7],
    [8, 0, 2, float('inf')],
    [5, float('inf'), 0, 1],
    [2, float('inf'), float('inf'), 0]
]

shortest_paths = floyd_warshall(graph)

print("Shortest path matrix:")
for row in shortest_paths:
    print(row)
```

```
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
Shortest path matrix:
[0, 3, 5, 6]
[5, 0, 2, 3]
[3, 6, 0, 1]
[2, 5, 7, 0]
>>>
```

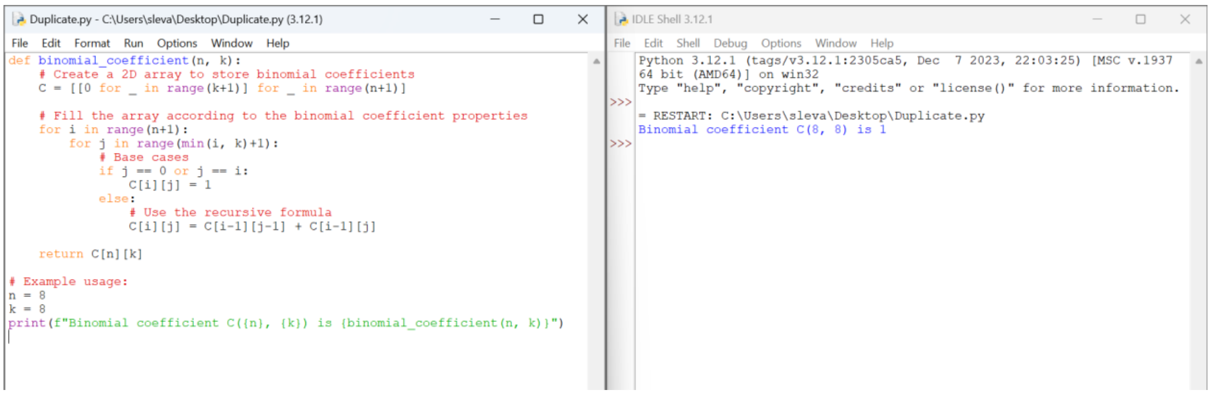
5. Linear Search


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
import time

def linear_search(arr, target):
    """
    Perform a linear search for the target in the array.

    :param arr: List of elements to search within.
    :param target: The element to search for.
    :return: The index of the target if found, else -1.
    """
    for index, element in enumerate(arr):
        if element == target:
            return index
    return -1

# Example usage and time measurement
arr = [1, 3, 5, 7, 9, 11, 13, 15, 17, 19]
target = 7

start_time = time.time()
result = linear_search(arr, target)
end_time = time.time()

if result != -1:
    print(f"Element {target} found at index {result}")
else:
    print(f"Element {target} not found in the array")

print(f"Time taken for linear search: {end_time - start_time:.10f} seconds")
```

```
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
Element 7 found at index 3
Time taken for linear search: 0.0000000000 seconds
>>>
```

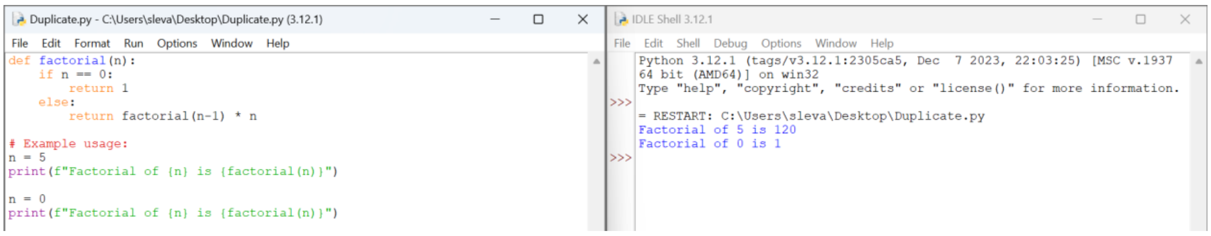
6.Binomial Coefficient


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 binomial_coefficient(n, k):
    # Create a 2D array to store binomial coefficients
    C = [[0 for _ in range(k+1)] for _ in range(n+1)]

    # Fill the array according to the binomial coefficient properties
    for i in range(n+1):
        for j in range(min(i, k)+1):
            # Base cases
            if j == 0 or j == i:
                C[i][j] = 1
            else:
                # Use the recursive formula
                C[i][j] = C[i-1][j-1] + C[i-1][j]

    return C[n][k]

# Example usage:
n = 8
k = 8
print(f"Binomial coefficient C({n}, {k}) is {binomial_coefficient(n, k)}")
```

```
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
Binomial coefficient C(8, 8) is 1
>>>
```

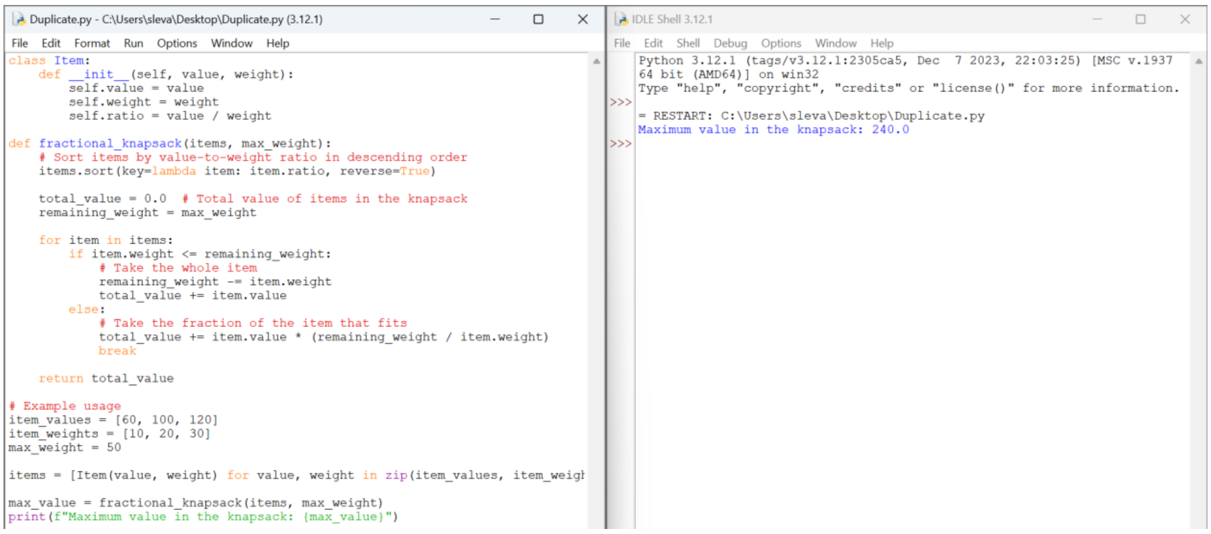
7. Factorial of a number


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

```
def factorial(n):
    if n == 0:
        return 1
    else:
        return factorial(n-1) * n

# Example usage:
n = 5
print(f"Factorial of {n} is {factorial(n)}")

n = 0
print(f"Factorial of {n} is {factorial(n)}")
```

```
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
Factorial of 5 is 120
Factorial of 0 is 1
>>>
```

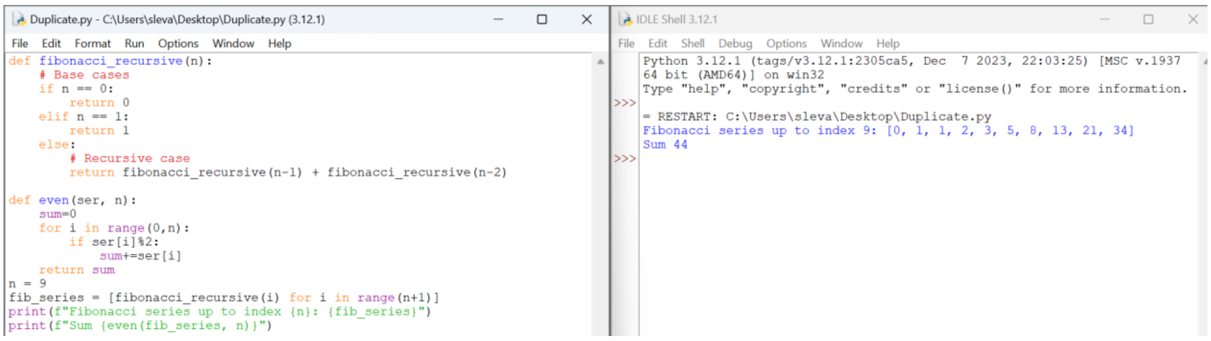
8. Knapsack Problem


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

```
class Item:
    def __init__(self, value, weight):
        self.value = value
        self.weight = weight
        self.ratio = value / weight

def fractional_knapsack(items, max_weight):
    # Sort items by value-to-weight ratio in descending order
    items.sort(key=lambda item: item.ratio, reverse=True)

    total_value = 0.0  # Total value of items in the knapsack
    remaining_weight = max_weight

    for item in items:
        if item.weight <= remaining_weight:
            # Take the whole item
            remaining_weight -= item.weight
            total_value += item.value
        else:
            # Take the fraction of the item that fits
            total_value += item.value * (remaining_weight / item.weight)
            break

    return total_value

# Example usage
item_values = [60, 100, 120]
item_weights = [10, 20, 30]
max_weight = 50

items = [Item(value, weight) for value, weight in zip(item_values, item_weig

max_value = fractional_knapsack(items, max_weight)
print(f"Maximum value in the knapsack: {max_value}")
```

```
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
Maximum value in the knapsack: 240.0
>>>
```

9. Fibonacci services


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

```
def fibonacci_recursive(n):
    # Base cases
    if n == 0:
        return 0
    elif n == 1:
        return 1
    else:
        # Recursive case
        return fibonacci_recursive(n-1) + fibonacci_recursive(n-2)

def even(ser, n):
    sum=0
    for i in range(0,n):
        if ser[i]%2:
            sum+=ser[i]
    return sum
n = 9
fib_series = [fibonacci_recursive(i) for i in range(n+1)]
print(f"Fibonacci series up to index {n}: {fib_series}")
print(f"Sum {even(fib_series, n)}")
```

```
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
Fibonacci series up to index 9: [0, 1, 1, 2, 3, 5, 8, 13, 21, 34]
Sum 44
>>>
```