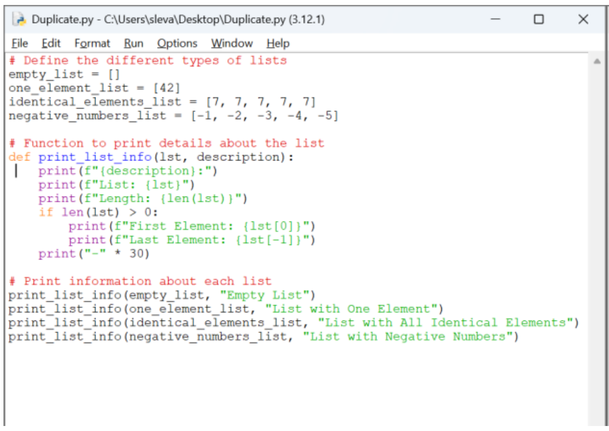
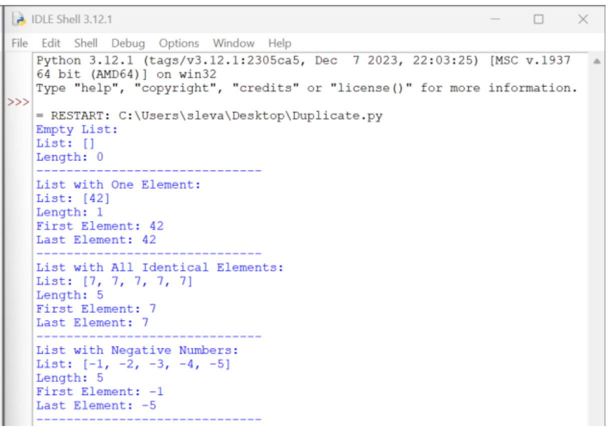
1. Different Lists

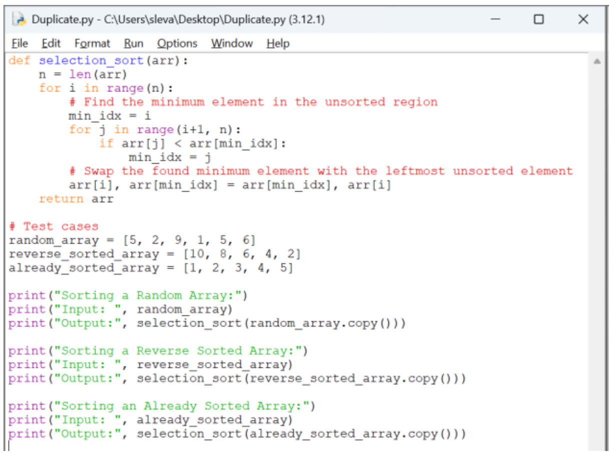

Duplicate.py - C:\Users\sleva\Desktop\Duplicate.py (3.12.1)
File Edit Format Run Options Window Help

```python
# Define the different types of lists
empty_list = []
one_element_list = [42]
identical_elements_list = [7, 7, 7, 7, 7]
negative_numbers_list = [-1, -2, -3, -4, -5]

# Function to print details about the list
def print_list_info(lst, description):
    print(f"{description}:")
    print(f"List: {lst}")
    print(f"Length: {len(lst)}")
    if len(lst) > 0:
        print(f"First Element: {lst[0]}")
        print(f"Last Element: {lst[-1]}")
    print("-" * 30)

# Print information about each list
print_list_info(empty_list, "Empty List")
print_list_info(one_element_list, "List with One Element")
print_list_info(identical_elements_list, "List with All Identical Elements")
print_list_info(negative_numbers_list, "List with Negative Numbers")
```

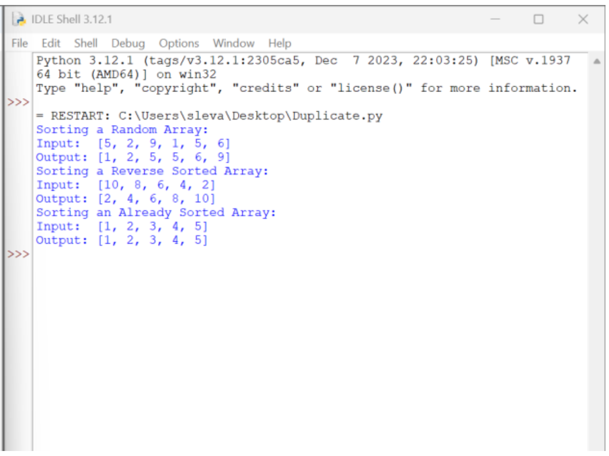


```
Python 3.12.1 (tags/v3.12.1:2305ca5, Dec  7 2023, 22:03:25) [MSC v.1937
64 bit (AMD64)] on win32
Type "help", "copyright", "credits" or "license()" for more information.
>>>
= RESTART: C:\Users\sleva\Desktop\Duplicate.py
Empty List:
List: []
Length: 0
------------------------------
List with One Element:
List: [42]
Length: 1
First Element: 42
Last Element: 42
------------------------------
List with All Identical Elements:
List: [7, 7, 7, 7, 7]
Length: 5
First Element: 7
Last Element: 7
------------------------------
List with Negative Numbers:
List: [-1, -2, -3, -4, -5]
Length: 5
First Element: -1
Last Element: -5
------------------------------
```

2. Selection Sort

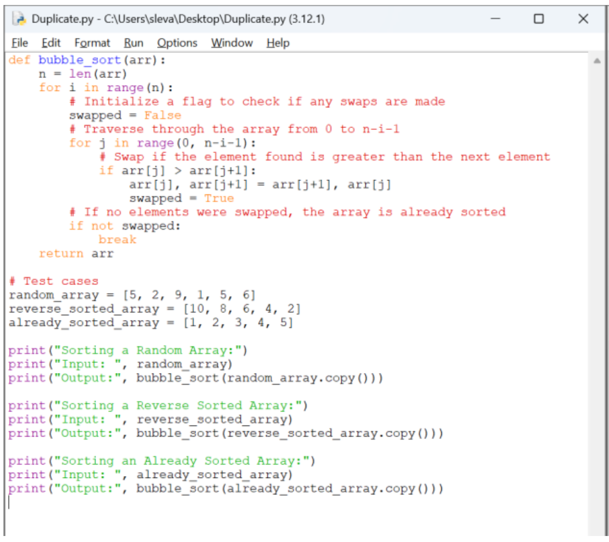

Duplicate.py - C:\Users\sleva\Desktop\Duplicate.py (3.12.1)
File Edit Format Run Options Window Help

```python
def selection_sort(arr):
    n = len(arr)
    for i in range(n):
        # Find the minimum element in the unsorted region
        min_idx = i
        for j in range(i+1, n):
            if arr[j] < arr[min_idx]:
                min_idx = j
        # Swap the found minimum element with the leftmost unsorted element
        arr[i], arr[min_idx] = arr[min_idx], arr[i]
    return arr

# Test cases
random_array = [5, 2, 9, 1, 5, 6]
reverse_sorted_array = [10, 8, 6, 4, 2]
already_sorted_array = [1, 2, 3, 4, 5]

print("Sorting a Random Array:")
print("Input: ", random_array)
print("Output:", selection_sort(random_array.copy()))

print("Sorting a Reverse Sorted Array:")
print("Input: ", reverse_sorted_array)
print("Output:", selection_sort(reverse_sorted_array.copy()))

print("Sorting an Already Sorted Array:")
print("Input: ", already_sorted_array)
print("Output:", selection_sort(already_sorted_array.copy()))
```

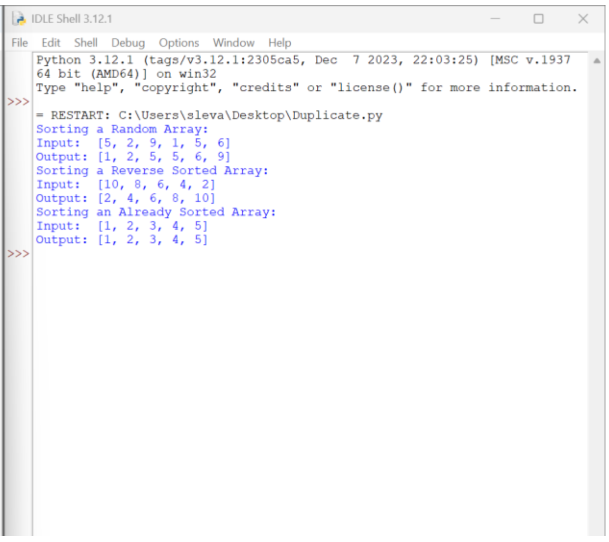


```
Python 3.12.1 (tags/v3.12.1:2305ca5, Dec  7 2023, 22:03:25) [MSC v.1937
64 bit (AMD64)] on win32
Type "help", "copyright", "credits" or "license()" for more information.
>>>
= RESTART: C:\Users\sleva\Desktop\Duplicate.py
Sorting a Random Array:
Input:  [5, 2, 9, 1, 5, 6]
Output: [1, 2, 5, 5, 6, 9]
Sorting a Reverse Sorted Array:
Input:  [10, 8, 6, 4, 2]
Output: [2, 4, 6, 8, 10]
Sorting an Already Sorted Array:
Input:  [1, 2, 3, 4, 5]
Output: [1, 2, 3, 4, 5]
>>>
```

3. Bubble Sort

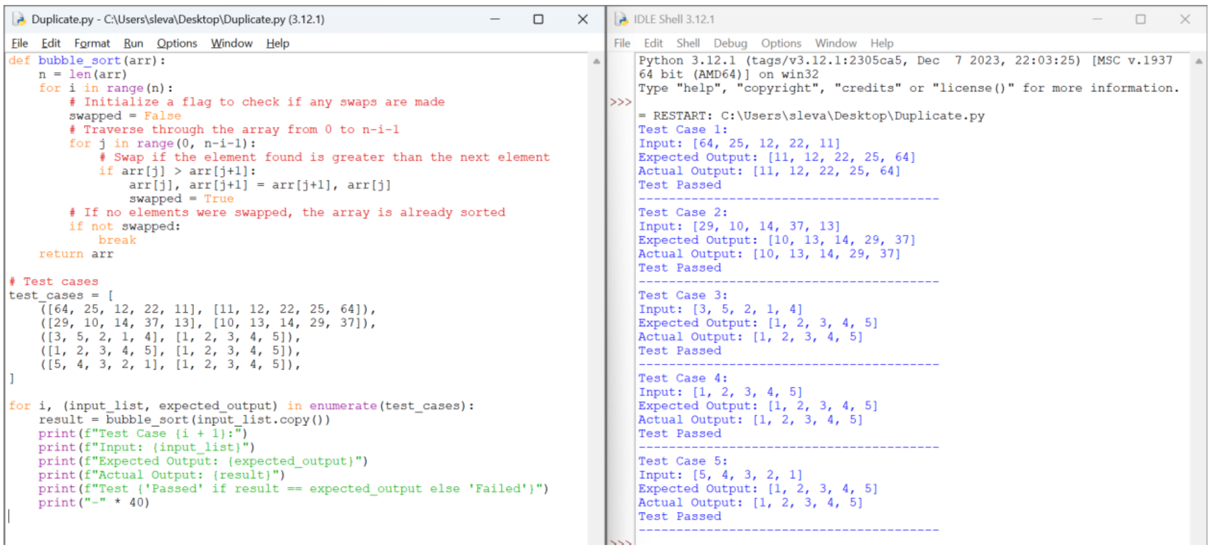

Duplicate.py - C:\Users\sleva\Desktop\Duplicate.py (3.12.1)
File Edit Format Run Options Window Help

```python
def bubble_sort(arr):
    n = len(arr)
    for i in range(n):
        # Initialize a flag to check if any swaps are made
        swapped = False
        # Traverse through the array from 0 to n-i-1
        for j in range(0, n-i-1):
            # Swap if the element found is greater than the next element
            if arr[j] > arr[j+1]:
                arr[j], arr[j+1] = arr[j+1], arr[j]
                swapped = True
        # If no elements were swapped, the array is already sorted
        if not swapped:
            break
    return arr

# Test cases
random_array = [5, 2, 9, 1, 5, 6]
reverse_sorted_array = [10, 8, 6, 4, 2]
already_sorted_array = [1, 2, 3, 4, 5]

print("Sorting a Random Array:")
print("Input: ", random_array)
print("Output:", bubble_sort(random_array.copy()))

print("Sorting a Reverse Sorted Array:")
print("Input: ", reverse_sorted_array)
print("Output:", bubble_sort(reverse_sorted_array.copy()))

print("Sorting an Already Sorted Array:")
print("Input: ", already_sorted_array)
print("Output:", bubble_sort(already_sorted_array.copy()))
```



```
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
Sorting a Random Array:
Input:  [5, 2, 9, 1, 5, 6]
Output: [1, 2, 5, 5, 6, 9]
Sorting a Reverse Sorted Array:
Input:  [10, 8, 6, 4, 2]
Output: [2, 4, 6, 8, 10]
Sorting an Already Sorted Array:
Input:  [1, 2, 3, 4, 5]
Output: [1, 2, 3, 4, 5]
>>>
```

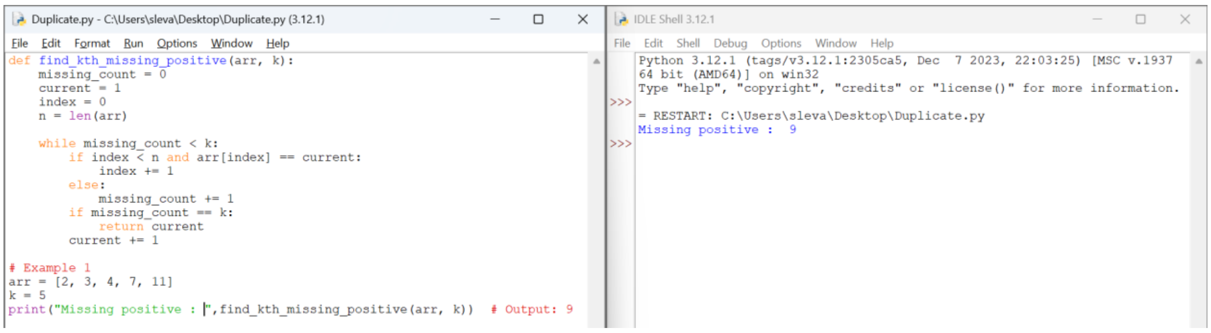
4. Sorting


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 bubble_sort(arr):
    n = len(arr)
    for i in range(n):
        # Initialize a flag to check if any swaps are made
        swapped = False
        # Traverse through the array from 0 to n-i-1
        for j in range(0, n-i-1):
            # Swap if the element found is greater than the next element
            if arr[j] > arr[j+1]:
                arr[j], arr[j+1] = arr[j+1], arr[j]
                swapped = True
        # If no elements were swapped, the array is already sorted
        if not swapped:
            break
    return arr

# Test cases
test_cases = [
    ([64, 25, 12, 22, 11], [11, 12, 22, 25, 64]),
    ([29, 10, 14, 37, 13], [10, 13, 14, 29, 37]),
    ([3, 5, 2, 1, 4], [1, 2, 3, 4, 5]),
    ([1, 2, 3, 4, 5], [1, 2, 3, 4, 5]),
    ([5, 4, 3, 2, 1], [1, 2, 3, 4, 5]),
]

for i, (input_list, expected_output) in enumerate(test_cases):
    result = bubble_sort(input_list.copy())
    print(f"Test Case {i + 1}:")
    print(f"Input: {input_list}")
    print(f"Expected Output: {expected_output}")
    print(f"Actual Output: {result}")
    print(f"Test {'Passed' if result == expected_output else 'Failed'}")
    print("-" * 40)
```

```
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
Test Case 1:
Input: [64, 25, 12, 22, 11]
Expected Output: [11, 12, 22, 25, 64]
Actual Output: [11, 12, 22, 25, 64]
Test Passed
----------------------------------------
Test Case 2:
Input: [29, 10, 14, 37, 13]
Expected Output: [10, 13, 14, 29, 37]
Actual Output: [10, 13, 14, 29, 37]
Test Passed
----------------------------------------
Test Case 3:
Input: [3, 5, 2, 1, 4]
Expected Output: [1, 2, 3, 4, 5]
Actual Output: [1, 2, 3, 4, 5]
Test Passed
----------------------------------------
Test Case 4:
Input: [1, 2, 3, 4, 5]
Expected Output: [1, 2, 3, 4, 5]
Actual Output: [1, 2, 3, 4, 5]
Test Passed
----------------------------------------
Test Case 5:
Input: [5, 4, 3, 2, 1]
Expected Output: [1, 2, 3, 4, 5]
Actual Output: [1, 2, 3, 4, 5]
Test Passed
----------------------------------------
```

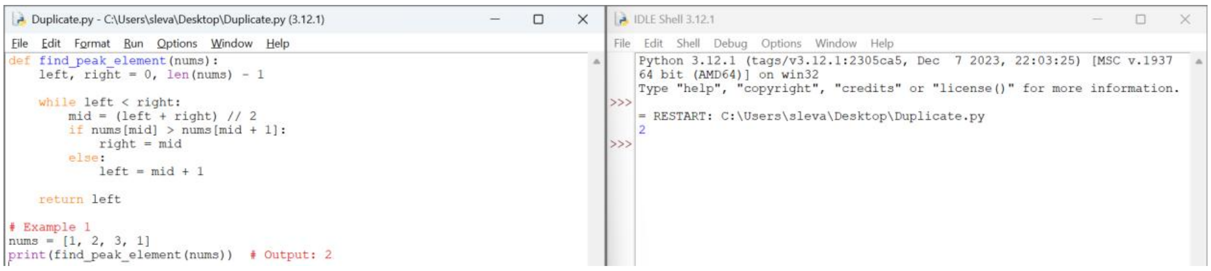
5. Kth missing positive


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 find_kth_missing_positive(arr, k):
    missing_count = 0
    current = 1
    index = 0
    n = len(arr)

    while missing_count < k:
        if index < n and arr[index] == current:
            index += 1
        else:
            missing_count += 1
        if missing_count == k:
            return current
        current += 1

# Example 1
arr = [2, 3, 4, 7, 11]
k = 5
print("Missing positive : ",find_kth_missing_positive(arr, k))  # Output: 9
```

```
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
Missing positive :  9
>>>
```

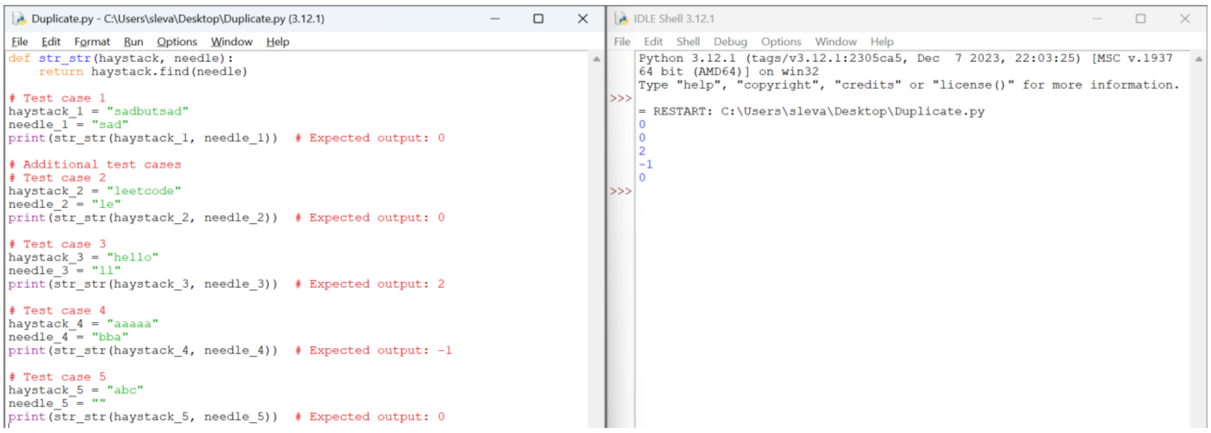
6. Peak element


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 find_peak_element(nums):
    left, right = 0, len(nums) - 1

    while left < right:
        mid = (left + right) // 2
        if nums[mid] > nums[mid + 1]:
            right = mid
        else:
            left = mid + 1

    return left

# Example 1
nums = [1, 2, 3, 1]
print(find_peak_element(nums))  # 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
>>>
```

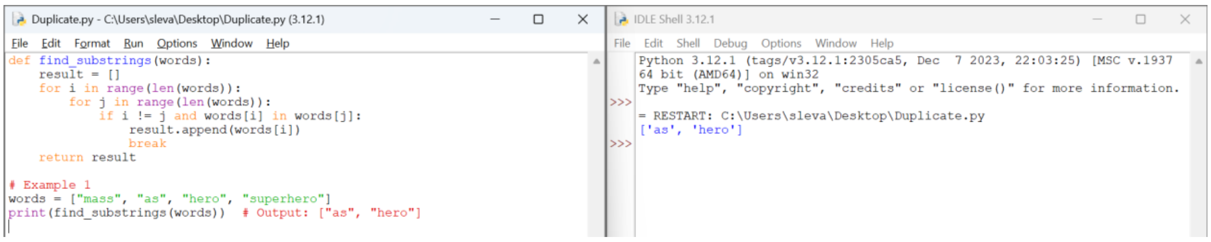
7. Haystack


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 str_str(haystack, needle):
    return haystack.find(needle)

# Test case 1
haystack_1 = "sadbutsad"
needle_1 = "sad"
print(str_str(haystack_1, needle_1))  # Expected output: 0

# Additional test cases
# Test case 2
haystack_2 = "leetcode"
needle_2 = "le"
print(str_str(haystack_2, needle_2))  # Expected output: 0

# Test case 3
haystack_3 = "hello"
needle_3 = "ll"
print(str_str(haystack_3, needle_3))  # Expected output: 2

# Test case 4
haystack_4 = "aaaaa"
needle_4 = "bba"
print(str_str(haystack_4, needle_4))  # Expected output: -1

# Test case 5
haystack_5 = "abc"
needle_5 = ""
print(str_str(haystack_5, needle_5))  # Expected output: 0
```

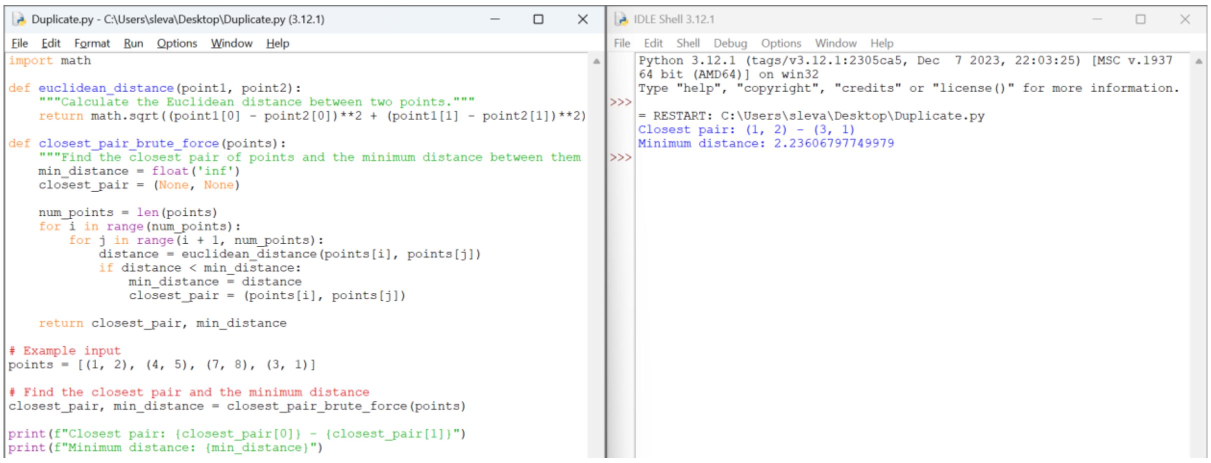
```
Python 3.12.1 (tags/v3.12.1:2305ca5, Dec  7 2023, 22:03:25) [MSC v.1937
64 bit (AMD64)] on win32
Type "help", "copyright", "credits" or "license()" for more information.
>>>
= RESTART: C:\Users\sleva\Desktop\Duplicate.py
0
0
2
-1
0
>>>
```

8. Finding Substring


```python
def find_substrings(words):
    result = []
    for i in range(len(words)):
        for j in range(len(words)):
            if i != j and words[i] in words[j]:
                result.append(words[i])
                break
    return result

# Example 1
words = ["mass", "as", "hero", "superhero"]
print(find_substrings(words))  # Output: ["as", "hero"]
```

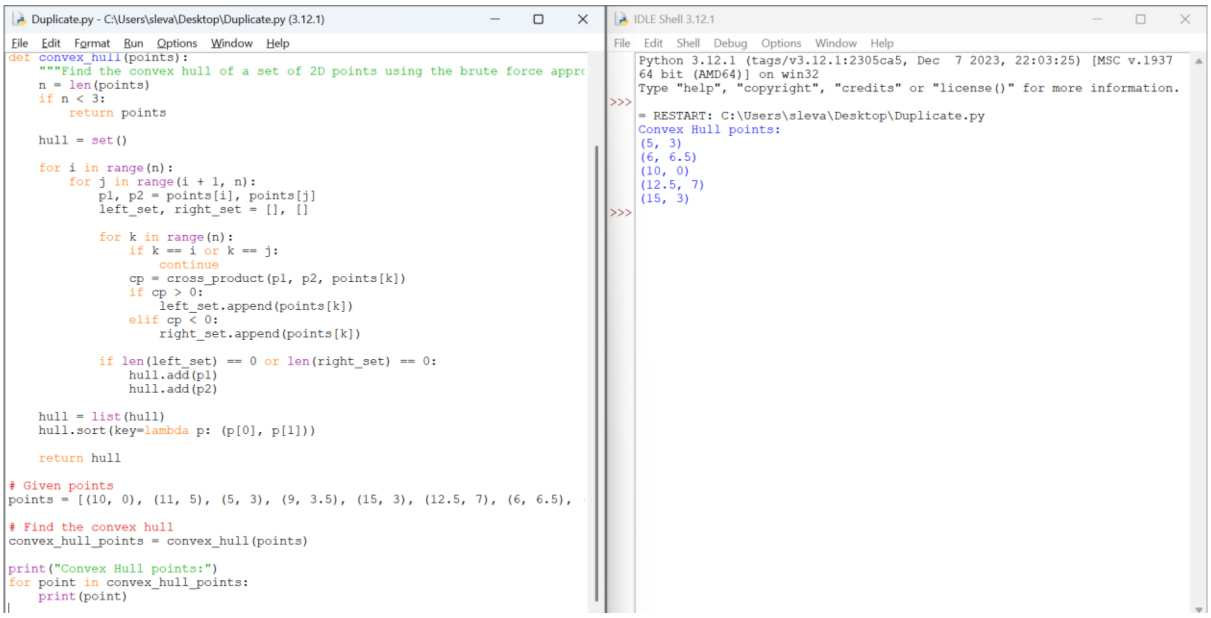
```
Python 3.12.1 (tags/v3.12.1:2305ca5, Dec  7 2023, 22:03:25) [MSC v.1937
64 bit (AMD64)] on win32
Type "help", "copyright", "credits" or "license()" for more information.
>>>
= RESTART: C:\Users\sleva\Desktop\Duplicate.py
['as', 'hero']
>>>
```

9. Closest pair of points


```python
import math

def euclidean_distance(point1, point2):
    """Calculate the Euclidean distance between two points."""
    return math.sqrt((point1[0] - point2[0])**2 + (point1[1] - point2[1])**2)

def closest_pair_brute_force(points):
    """Find the closest pair of points and the minimum distance between them
    min_distance = float('inf')
    closest_pair = (None, None)

    num_points = len(points)
    for i in range(num_points):
        for j in range(i + 1, num_points):
            distance = euclidean_distance(points[i], points[j])
            if distance < min_distance:
                min_distance = distance
                closest_pair = (points[i], points[j])

    return closest_pair, min_distance

# Example input
points = [(1, 2), (4, 5), (7, 8), (3, 1)]

# Find the closest pair and the minimum distance
closest_pair, min_distance = closest_pair_brute_force(points)

print(f"Closest pair: {closest_pair[0]} - {closest_pair[1]}")
print(f"Minimum distance: {min_distance}")
```

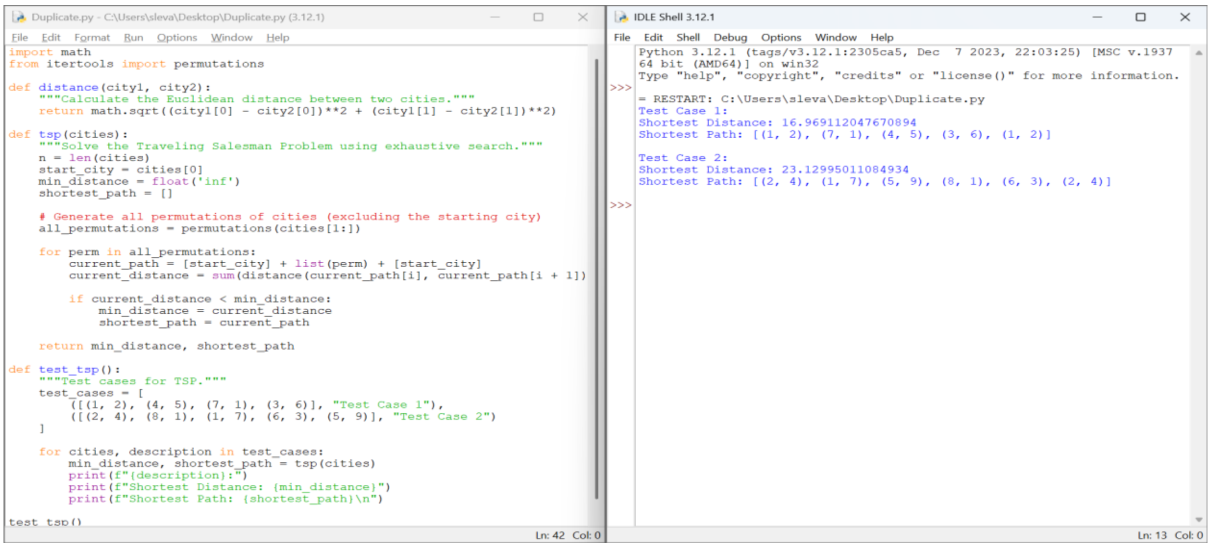
```
Python 3.12.1 (tags/v3.12.1:2305ca5, Dec  7 2023, 22:03:25) [MSC v.1937
64 bit (AMD64)] on win32
Type "help", "copyright", "credits" or "license()" for more information.
>>>
= RESTART: C:\Users\sleva\Desktop\Duplicate.py
Closest pair: (1, 2) - (3, 1)
Minimum distance: 2.23606797749979
>>>
```

11. Convex Hull


```python
def convex_hull(points):
    """Find the convex hull of a set of 2D points using the brute force appr
    n = len(points)
    if n < 3:
        return points

    hull = set()

    for i in range(n):
        for j in range(i + 1, n):
            p1, p2 = points[i], points[j]
            left_set, right_set = [], []

            for k in range(n):
                if k == i or k == j:
                    continue
                cp = cross_product(p1, p2, points[k])
                if cp > 0:
                    left_set.append(points[k])
                elif cp < 0:
                    right_set.append(points[k])

            if len(left_set) == 0 or len(right_set) == 0:
                hull.add(p1)
                hull.add(p2)

    hull = list(hull)
    hull.sort(key=lambda p: (p[0], p[1]))

    return hull

# Given points
points = [(10, 0), (11, 5), (5, 3), (9, 3.5), (15, 3), (12.5, 7), (6, 6.5),

# Find the convex hull
convex_hull_points = convex_hull(points)

print("Convex Hull points:")
for point in convex_hull_points:
    print(point)
```

```
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
Convex Hull points:
(5, 3)
(6, 6.5)
(10, 0)
(12.5, 7)
(15, 3)
>>>
```

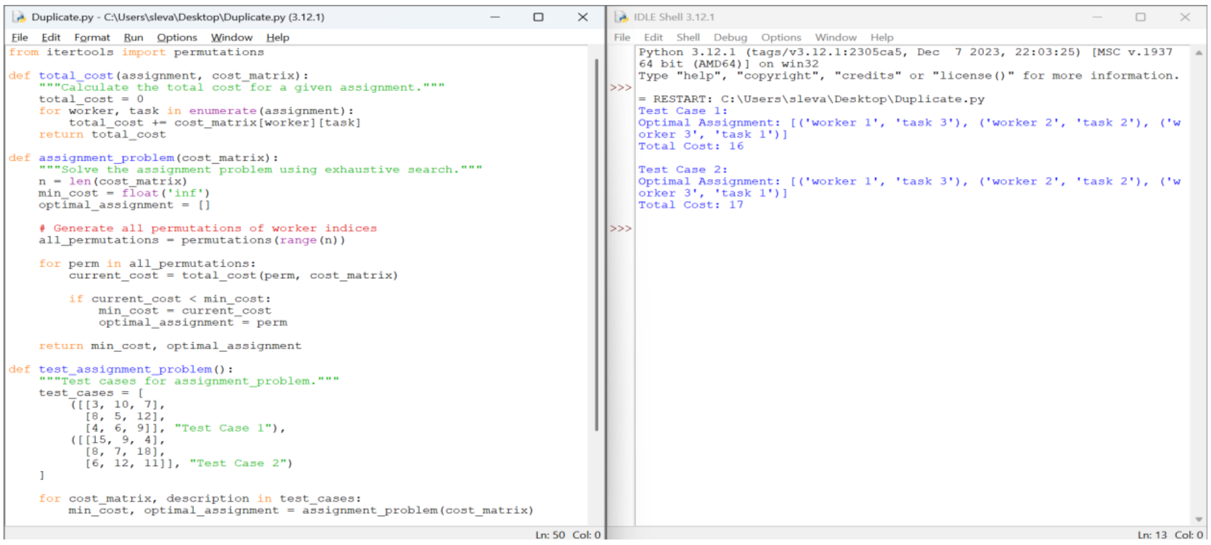
12. TSP using Exhaustive Search


Duplicate.py - C:\Users\sleva\Desktop\Duplicate.py (3.12.1)
File Edit Format Run Options Window Help
Ln: 42 Col: 0
IDLE Shell 3.12.1
File Edit Shell Debug Options Window Help
Ln: 13 Col: 0

```python
import math
from itertools import permutations

def distance(city1, city2):
    """Calculate the Euclidean distance between two cities."""
    return math.sqrt((city1[0] - city2[0])**2 + (city1[1] - city2[1])**2)

def tsp(cities):
    """Solve the Traveling Salesman Problem using exhaustive search."""
    n = len(cities)
    start_city = cities[0]
    min_distance = float('inf')
    shortest_path = []

    # Generate all permutations of cities (excluding the starting city)
    all_permutations = permutations(cities[1:])

    for perm in all_permutations:
        current_path = [start_city] + list(perm) + [start_city]
        current_distance = sum(distance(current_path[i], current_path[i + 1])

        if current_distance < min_distance:
            min_distance = current_distance
            shortest_path = current_path

    return min_distance, shortest_path

def test_tsp():
    """Test cases for TSP."""
    test_cases = [
        ([(1, 2), (4, 5), (7, 1), (3, 6)], "Test Case 1"),
        ([(2, 4), (8, 1), (1, 7), (6, 3), (5, 9)], "Test Case 2")
    ]

    for cities, description in test_cases:
        min_distance, shortest_path = tsp(cities)
        print(f"{description}:")
        print(f"Shortest Distance: {min_distance}")
        print(f"Shortest Path: {shortest_path}\n")

test_tsp()
```

```
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
Test Case 1:
Shortest Distance: 16.969112047670894
Shortest Path: [(1, 2), (7, 1), (4, 5), (3, 6), (1, 2)]

Test Case 2:
Shortest Distance: 23.12995011084934
Shortest Path: [(2, 4), (1, 7), (5, 9), (8, 1), (6, 3), (2, 4)]

>>>
```

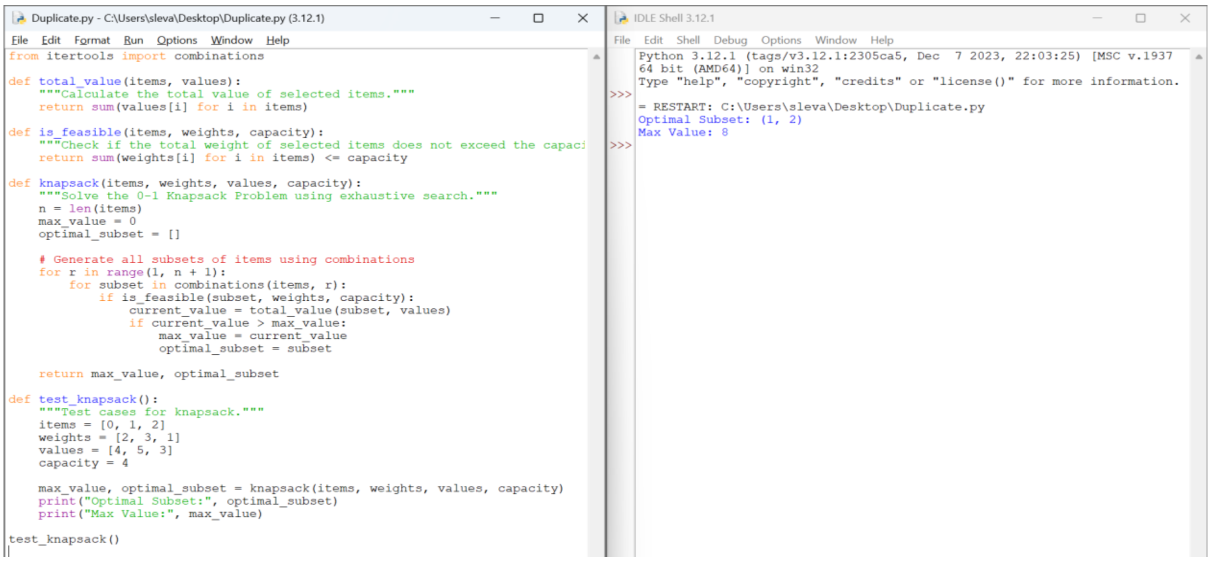
13. Assignment Problem using Exhaustive Search


Duplicate.py - C:\Users\sleva\Desktop\Duplicate.py (3.12.1)
File Edit Format Run Options Window Help
Ln: 50 Col: 0
IDLE Shell 3.12.1
File Edit Shell Debug Options Window Help
Ln: 13 Col: 0

```python
from itertools import permutations

def total_cost(assignment, cost_matrix):
    """Calculate the total cost for a given assignment."""
    total_cost = 0
    for worker, task in enumerate(assignment):
        total_cost += cost_matrix[worker][task]
    return total_cost

def assignment_problem(cost_matrix):
    """Solve the assignment problem using exhaustive search."""
    n = len(cost_matrix)
    min_cost = float('inf')
    optimal_assignment = []

    # Generate all permutations of worker indices
    all_permutations = permutations(range(n))

    for perm in all_permutations:
        current_cost = total_cost(perm, cost_matrix)

        if current_cost < min_cost:
            min_cost = current_cost
            optimal_assignment = perm

    return min_cost, optimal_assignment

def test_assignment_problem():
    """Test cases for assignment_problem."""
    test_cases = [
        ([[3, 10, 7],
          [8, 5, 12],
          [4, 6, 9]], "Test Case 1"),
        ([[15, 9, 4],
          [8, 7, 18],
          [6, 12, 11]], "Test Case 2")
    ]

    for cost_matrix, description in test_cases:
        min_cost, optimal_assignment = assignment_problem(cost_matrix)
```

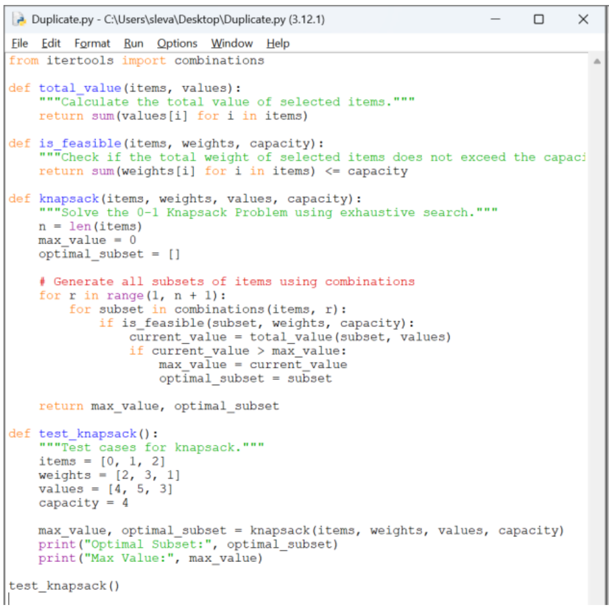
```
Python 3.12.1 (tags/v3.12.1:2305ca5, Dec  7 2023, 22:03:25) [MSC v.1937
64 bit (AMD64)] on win32
Type "help", "copyright", "credits" or "license()" for more information.
>>>
= RESTART: C:\Users\sleva\Desktop\Duplicate.py
Test Case 1:
Optimal Assignment: [('worker 1', 'task 3'), ('worker 2', 'task 2'), ('w
orker 3', 'task 1')]
Total Cost: 16

Test Case 2:
Optimal Assignment: [('worker 1', 'task 3'), ('worker 2', 'task 2'), ('w
orker 3', 'task 1')]
Total Cost: 17

>>>
```

14. Kanpsack Problem using Exhaustive Search

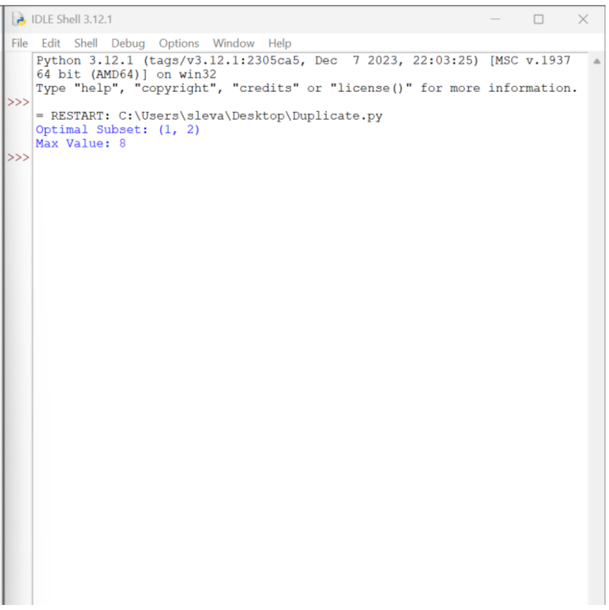

Duplicate.py - C:\Users\sleva\Desktop\Duplicate.py (3.12.1)
File Edit Format Run Options Window Help
IDLE Shell 3.12.1
File Edit Shell Debug Options Window Help

```python
from itertools import combinations

def total_value(items, values):
    """Calculate the total value of selected items."""
    return sum(values[i] for i in items)

def is_feasible(items, weights, capacity):
    """Check if the total weight of selected items does not exceed the capaci
    return sum(weights[i] for i in items) <= capacity

def knapsack(items, weights, values, capacity):
    """Solve the 0-1 Knapsack Problem using exhaustive search."""
    n = len(items)
    max_value = 0
    optimal_subset = []

    # Generate all subsets of items using combinations
    for r in range(1, n + 1):
        for subset in combinations(items, r):
            if is_feasible(subset, weights, capacity):
                current_value = total_value(subset, values)
                if current_value > max_value:
                    max_value = current_value
                    optimal_subset = subset

    return max_value, optimal_subset

def test_knapsack():
    """Test cases for knapsack."""
    items = [0, 1, 2]
    weights = [2, 3, 1]
    values = [4, 5, 3]
    capacity = 4

    max_value, optimal_subset = knapsack(items, weights, values, capacity)
    print("Optimal Subset:", optimal_subset)
    print("Max Value:", max_value)

test_knapsack()
```

```
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
Optimal Subset: (1, 2)
Max Value: 8
>>>
```


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

```python
from itertools import combinations

def total_value(items, values):
    """Calculate the total value of selected items."""
    return sum(values[i] for i in items)

def is_feasible(items, weights, capacity):
    """Check if the total weight of selected items does not exceed the capaci
    return sum(weights[i] for i in items) <= capacity

def knapsack(items, weights, values, capacity):
    """Solve the 0-1 Knapsack Problem using exhaustive search."""
    n = len(items)
    max_value = 0
    optimal_subset = []

    # Generate all subsets of items using combinations
    for r in range(1, n + 1):
        for subset in combinations(items, r):
            if is_feasible(subset, weights, capacity):
                current_value = total_value(subset, values)
                if current_value > max_value:
                    max_value = current_value
                    optimal_subset = subset

    return max_value, optimal_subset

def test_knapsack():
    """Test cases for knapsack."""
    items = [0, 1, 2]
    weights = [2, 3, 1]
    values = [4, 5, 3]
    capacity = 4

    max_value, optimal_subset = knapsack(items, weights, values, capacity)
    print("Optimal Subset:", optimal_subset)
    print("Max Value:", max_value)

test_knapsack()
```



```
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
Optimal Subset: (1, 2)
Max Value: 8
>>>
```