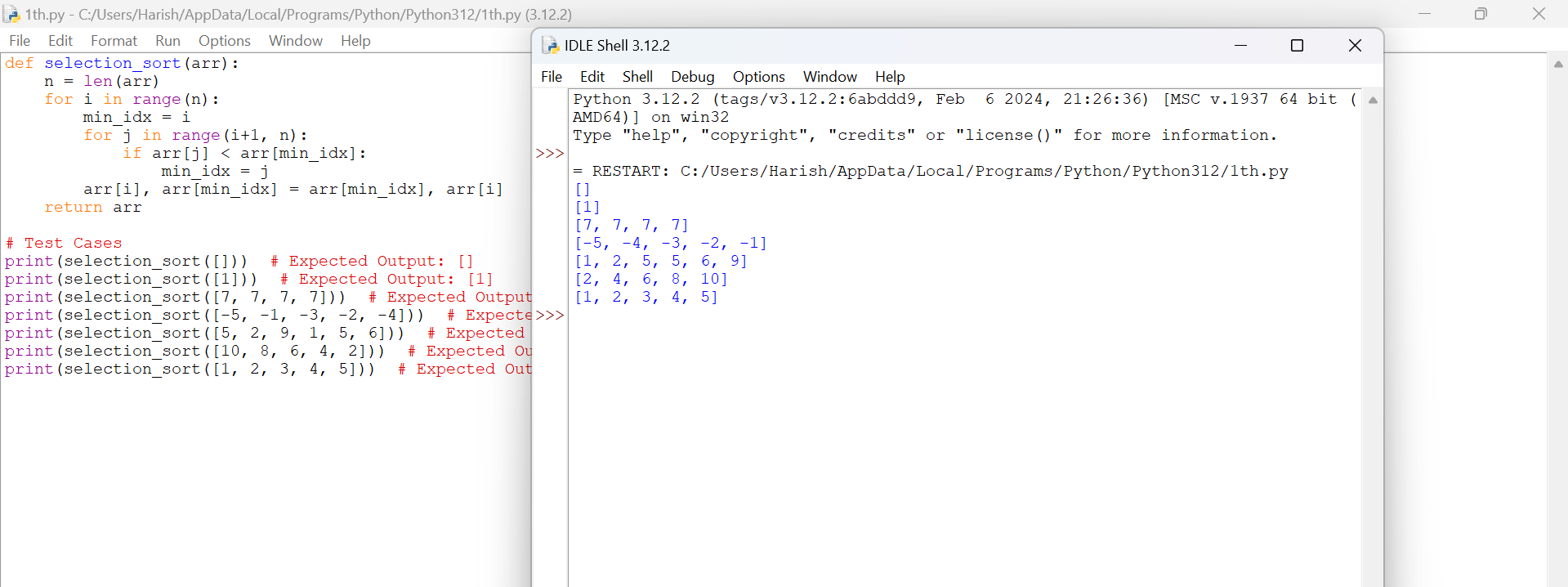
**DAY-2 LAB PRACTICE DAA**

1. Write a program to perform the following

* + An empty list
  + A list with one element
  + A list with all identical elements
  + A list with negative numbers

**Test Cases:**

1. **Input:** []
   * **Expected Output:** []

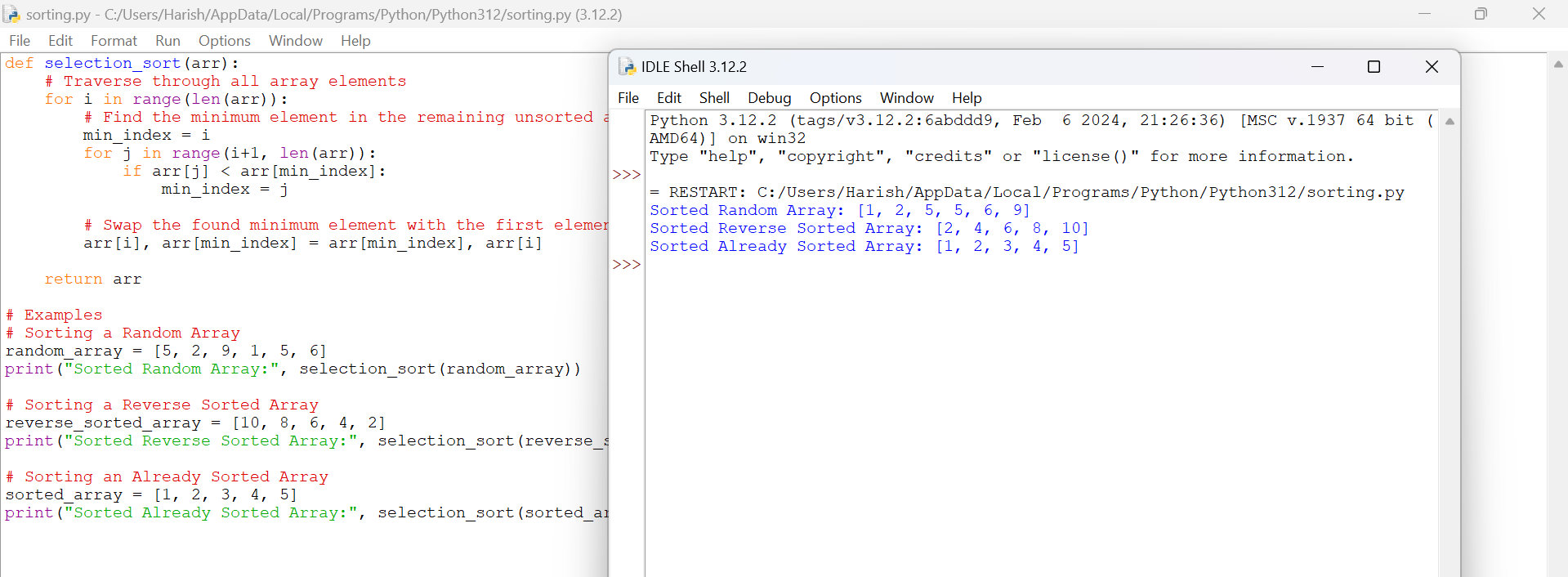


2.Describe the Selection Sort algorithm's process of sorting an array. Selection Sort works by dividing the array into a sorted and an unsorted region. Initially, the sorted region is empty, and the unsorted region contains all elements. The algorithm repeatedly selects the smallest element from the unsorted region and swaps it with the leftmost unsorted element, then moves the boundary of the sorted region one element to the right. Explain why Selection Sort is simple to understand and implement but is inefficient for large datasets. Provide examples to illustrate step-by-step how Selection Sort rearranges the elements into ascending order, ensuring clarity in your explanation of the algorithm's mechanics and effectiveness.

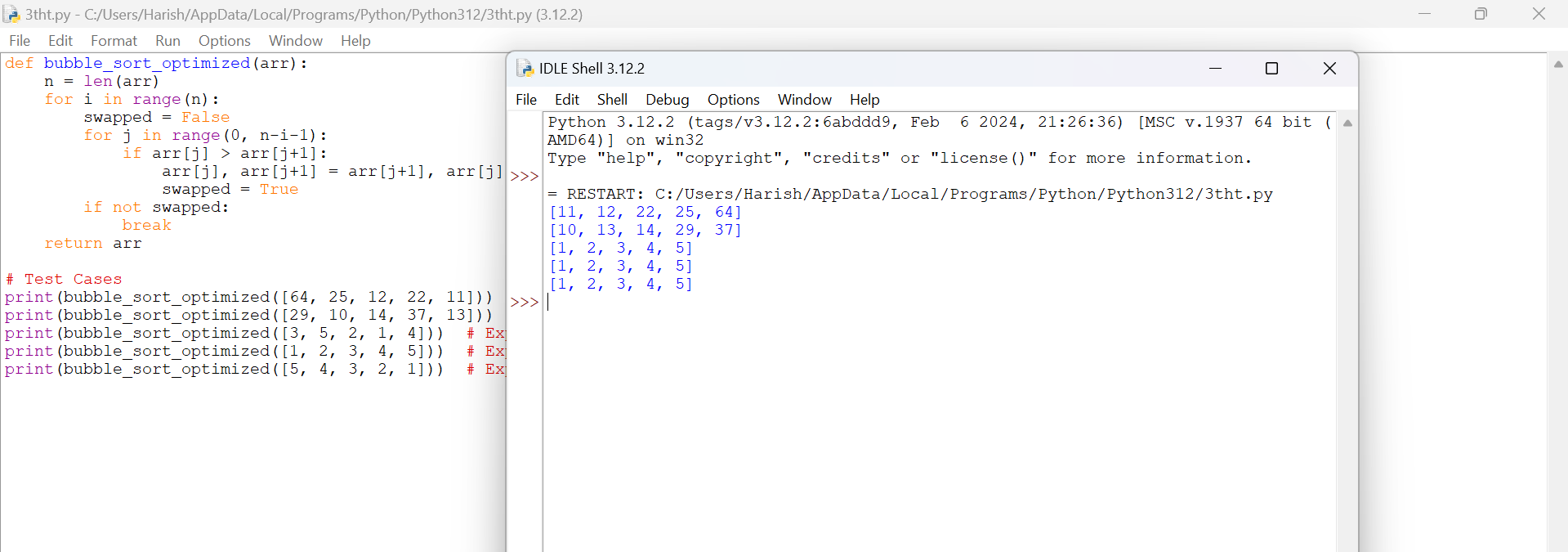
**Sorting a Random Array**:

**Input**: [5, 2, 9, 1, 5, 6]

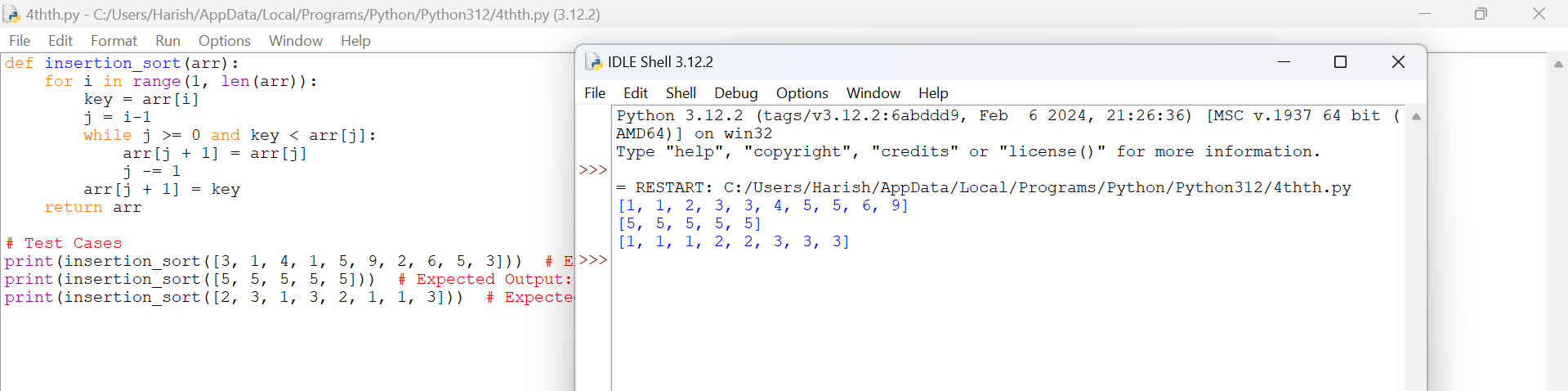
**Output**: [1, 2, 5, 5, 6, 9]



3.Write code to modify bubble sort function to stop early if the list becomes sorted before all passes are completed



4.Write code for Insertion Sort that manages arrays with duplicate elements during the sorting process. Ensure the algorithm's behavior when encountering duplicate values, including whether it preserves the relative order of duplicates and how it affects the overall sorting outcome.

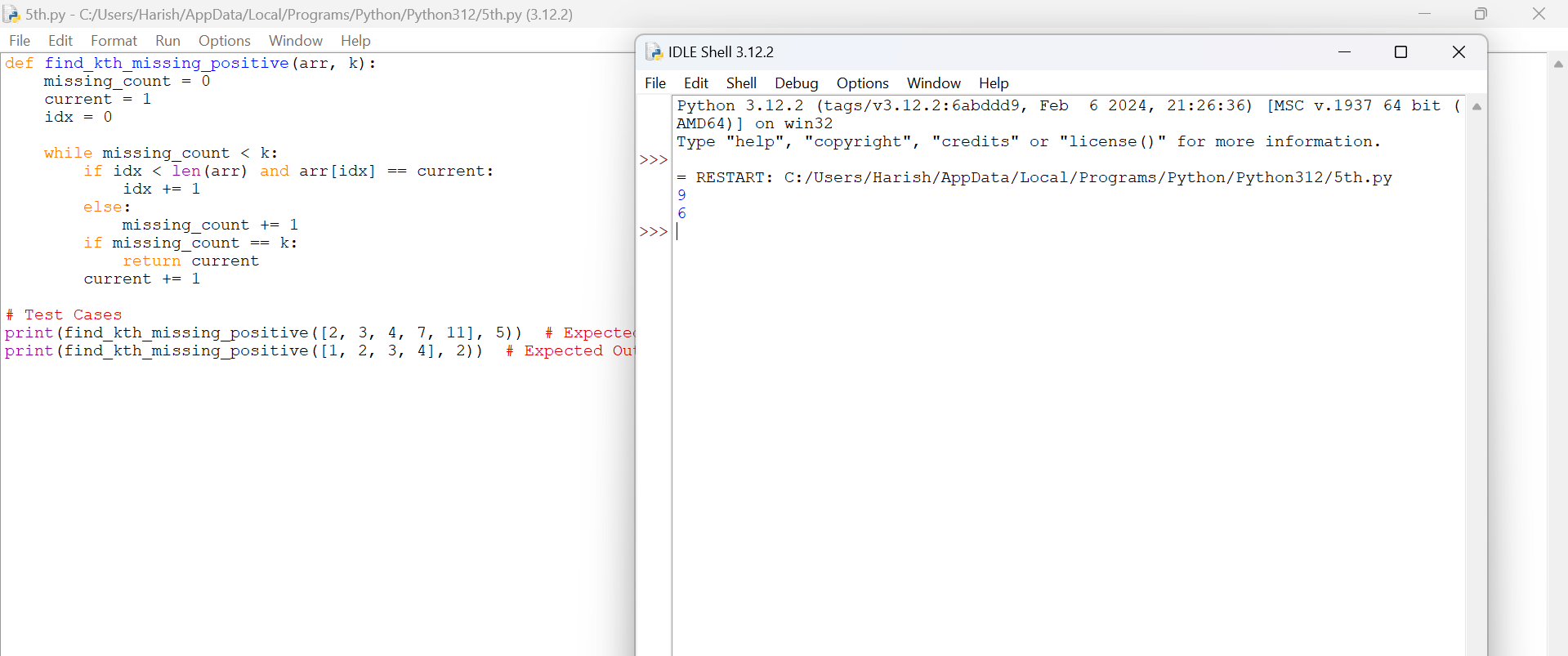


5.Given an array arr of positive integers sorted in a strictly increasing order, and an integer k. return the kth positive integer that is missing from this array.

Example 1:

Input: arr = [2,3,4,7,11], k = 5

Output: 9

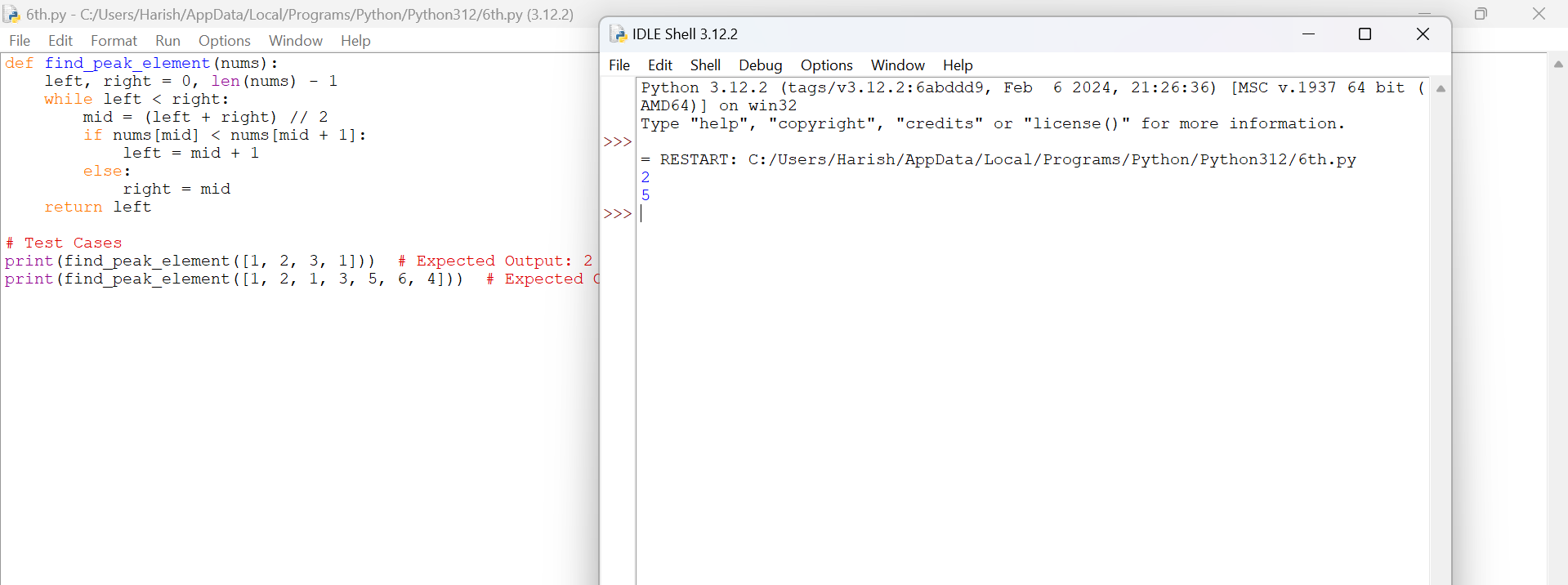


6.A peak element is an element that is strictly greater than its neighbors. Given a 0-indexed integer array nums, find a peak element, and return its index. If the array contains multiple peaks, return the index to any of the peaks. You may imagine that nums[-1] = nums[n] = -∞. In other words, an element is always considered to be strictly greater than a neighbor that is outside the array. You must write an algorithm that runs in O(log n) time.

Example 1:

Input: nums = [1,2,3,1]

Output: 2

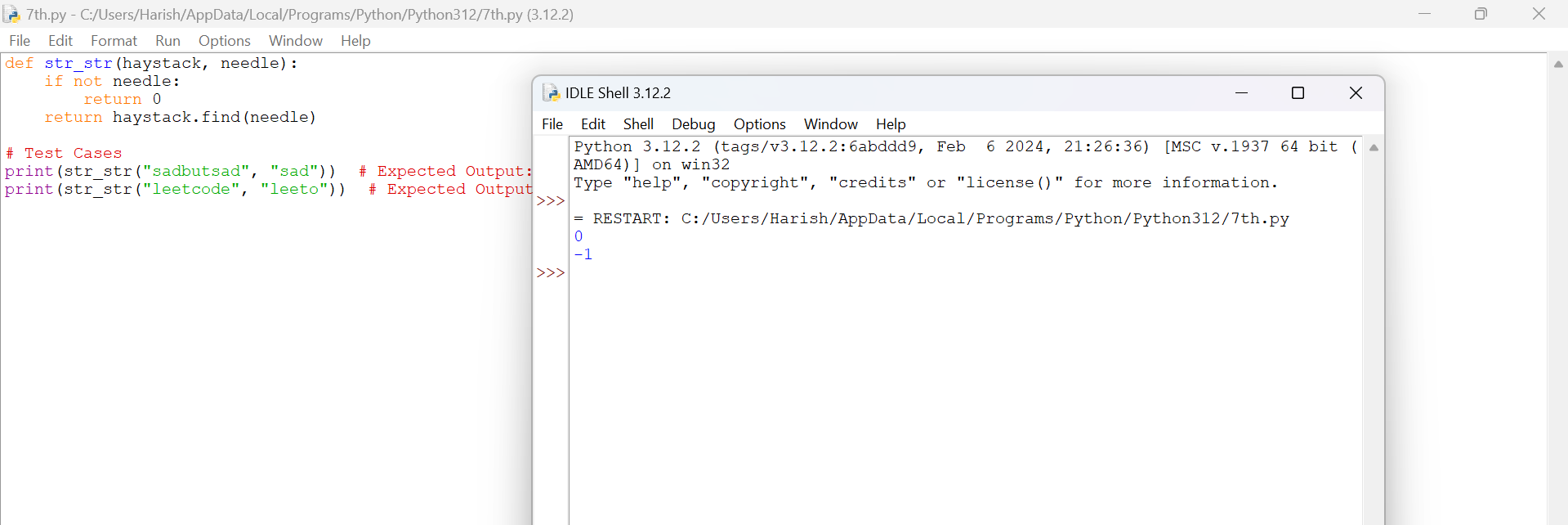


7.Given two strings needle and haystack, return the index of the first occurrence of needle in haystack, or -1 if needle is not part of haystack.

Example 1:

Input: haystack = "sadbutsad", needle = "sad"

Output: 0

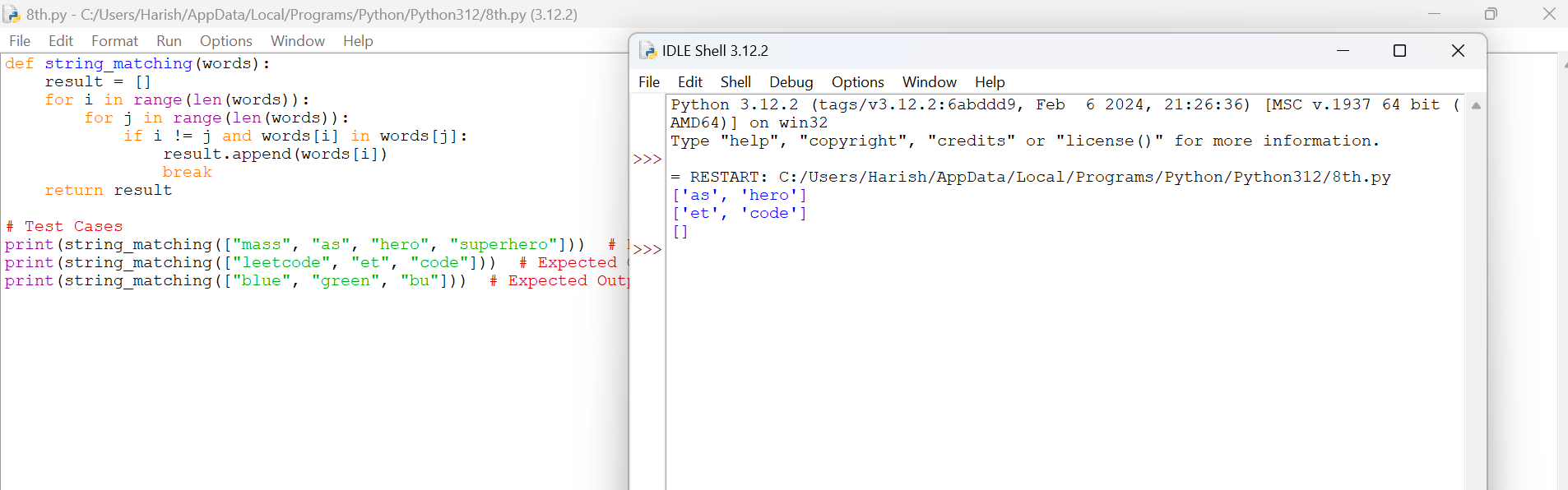


8.Given an array of string words, return all strings in words that is a substring of another word. You can return the answer in any order. A substring is a contiguous sequence of characters within a string

Example 1:

Input: words = ["mass","as","hero","superhero"]

Output: ["as","hero"]



9.Write a program that finds the closest pair of points in a set of 2D points using the brute force approach.

Input:

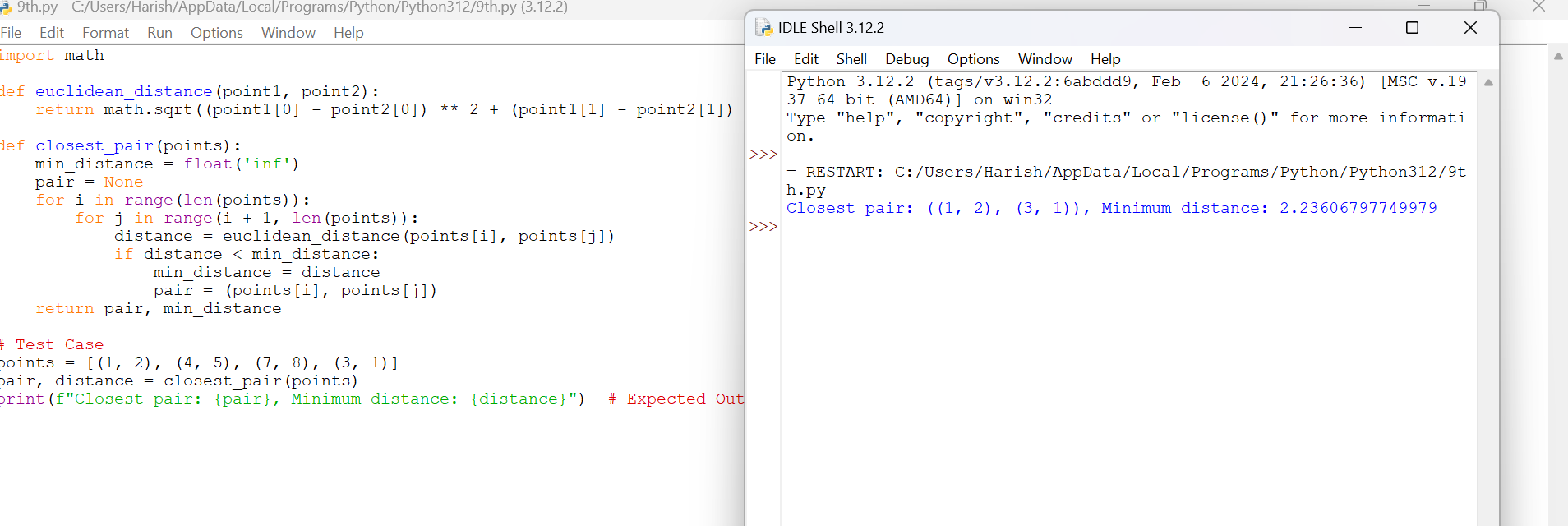
* A list or array of points represented by coordinates (x, y).

Points: [(1, 2), (4, 5), (7, 8), (3, 1)]

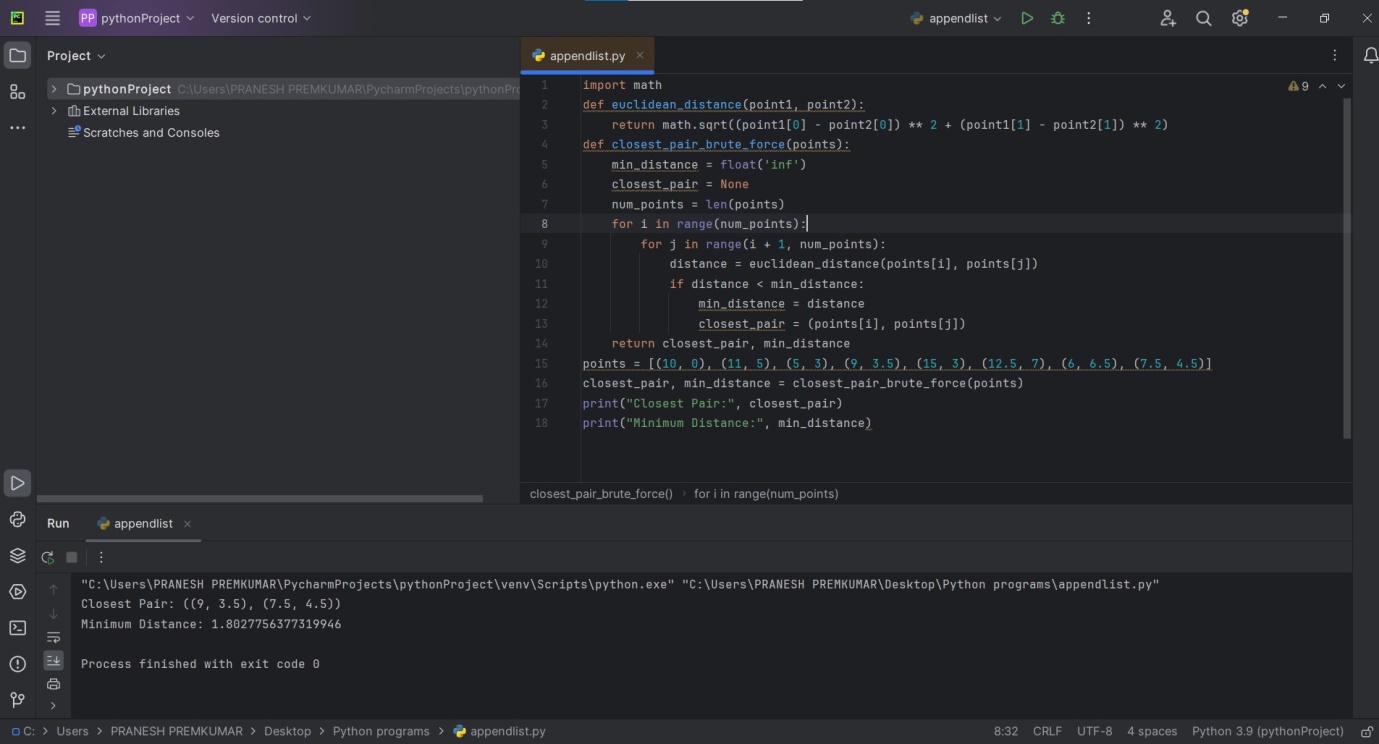
Output:

* The two points with the minimum distance between them.
* The minimum distance itself.

Closest pair: (1, 2) - (3, 1) Minimum distance: 1.4142135623730951



10.Write a program to find the closest pair of points in a given set using the brute force approach. Analyze the time complexity of your implementation. Define a function to calculate the Euclidean distance between two points. Implement a function to find the closest pair of points using the brute force method. Test your program with a sample set of points and verify the correctness of your results. Analyze the time complexity of your implementation. Write a brute-force algorithm to solve the convex hull problem for the following set S of points? P1 (10,0)P2 (11,5)P3 (5, 3)P4 (9, 3.5)P5 (15, 3)P6 (12.5, 7)P7 (6, 6.5)P8 (7.5, 4.5).How do you modify your brute force algorithm to handle multiple points that are lying on the sameline?



11.Write a program that finds the convex hull of a set of 2D points using the brute force approach.

**Input:**

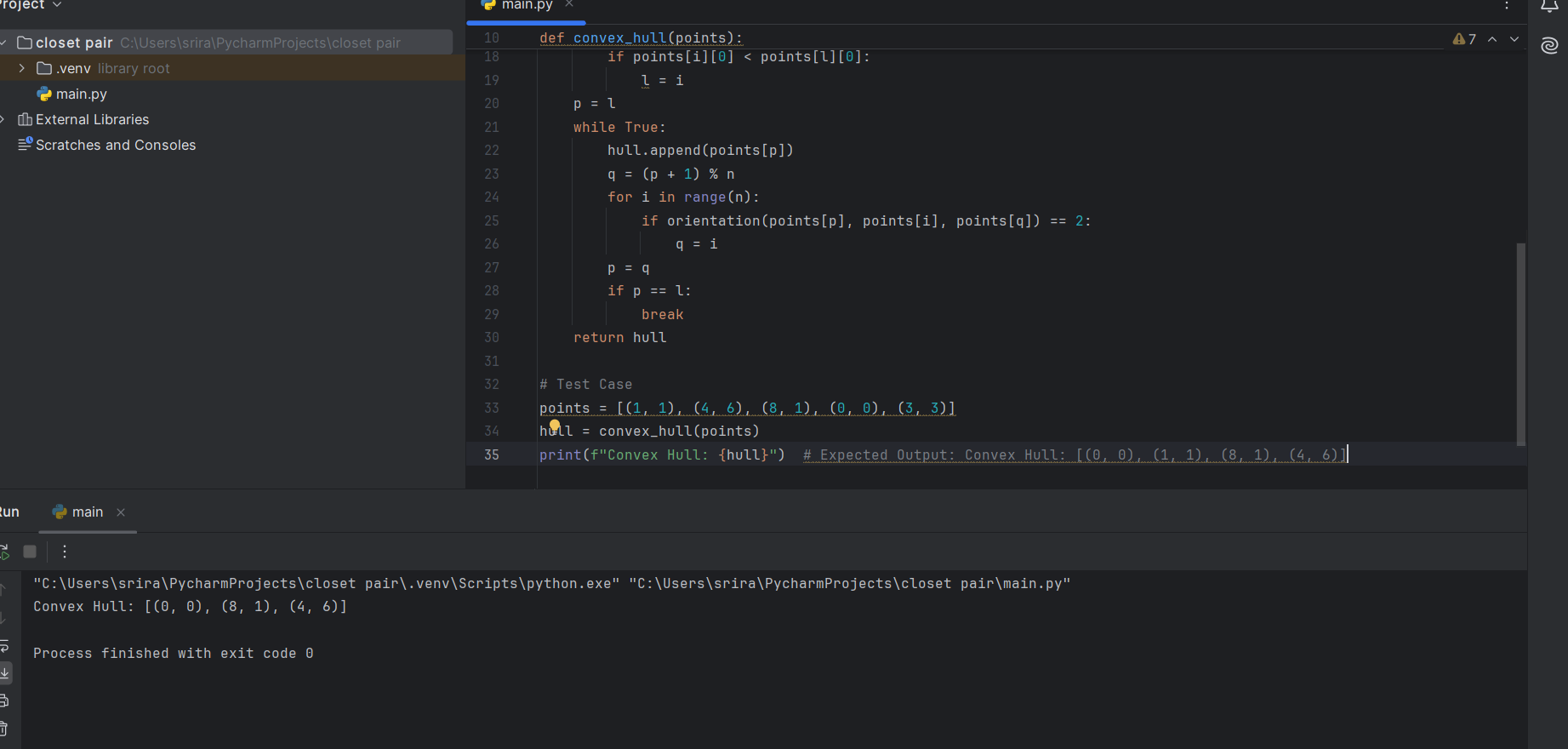
* A list or array of points represented by coordinates (x, y).

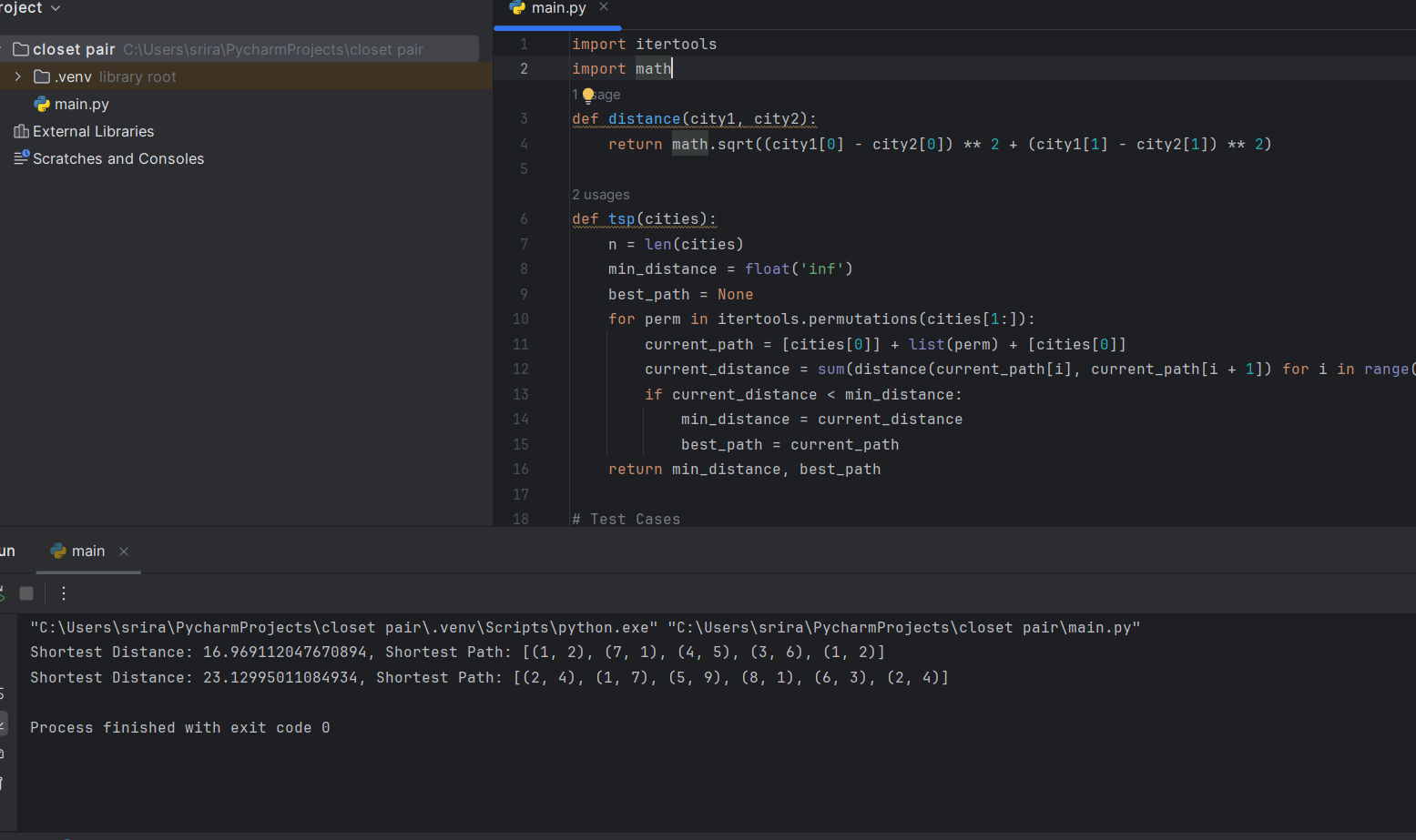
Points: [(1, 1), (4, 6), (8, 1), (0, 0), (3, 3)]

**Output:**

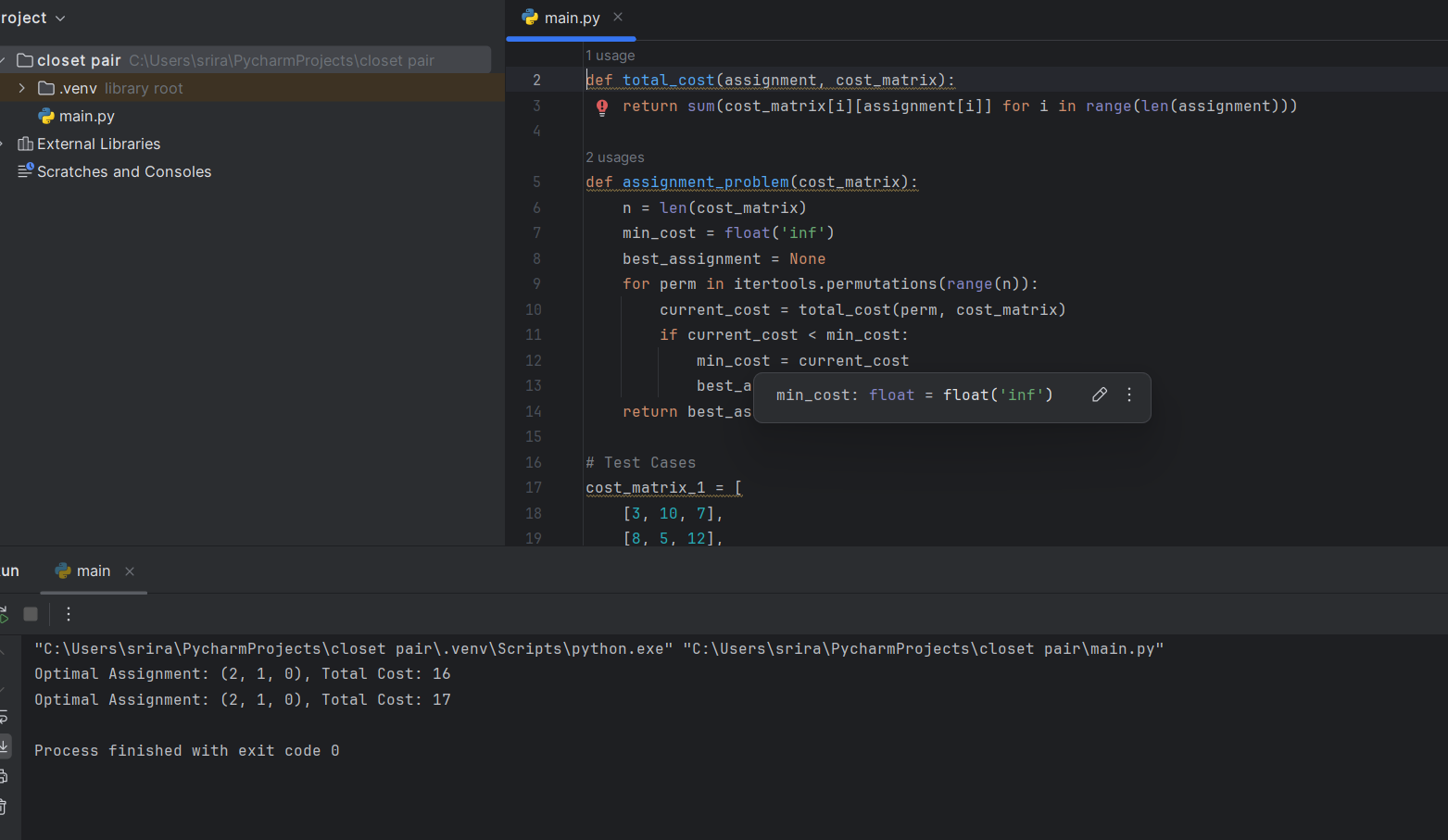
* The list of points that form the convex hull in counter-clockwise order.

Convex Hull: [(0, 0), (1, 1), (8, 1), (4, 6)]

12.You are given a list of cities represented by their coordinates. Develop a program that utilizes exhaustive search to solve the TSP. The program should:



13.You are given a cost matrix where each element cost[i][j] represents the cost of assigning worker i to task j. Develop a program that utilizes exhaustive search to solve the assignment problem. The program should Define a function total\_cost(assignment, cost\_matrix) that takes an assignment (list representing worker-task pairings) and the cost matrix as input. It iterates through the assignment and calculates the total cost by summing the corresponding costs from the cost matrix Implement a function assignment\_problem(cost\_matrix) that takes the cost matrix as input and performs the following Generate all possible permutations of worker indices (excluding repetitions).



14.You are given a list of items with their weights and values. Develop a program that utilizes exhaustive search to solve the 0-1 Knapsack Problem. The program should:

1. Define a function total\_value(items, values) that takes a list of selected items (represented by their indices) and the value list as input. It iterates through the selected items and calculates the total value by summing the corresponding values from the value list.
2. Define a function is\_feasible(items, weights, capacity) that takes a list of selected items (represented by their indices), the weight list, and the knapsack capacity as input. It checks if the total weight of the selected items exceeds the capacity.

