**PART A**

(PART A : TO BE REFFERED BY STUDENTS)

**EXPERIMENT NO. 3**

**A.1 AIM: -** To perform Matrix Operation, find minimum cost path, find maximum in an integer array, and array sorting.

**A.2 Prerequisite**

* Different programming language (Python or Java), Understanding of Machine Learning Algorithms, Machine Learning Algorithms

**A.3 Outcome**

After successful completion of this experiment students will be able to understand working of matrix, find minimum and maximum cost paths

**A.4 Theory**

**Min Cost Path**

The minimum cost path problem in Java is one the most prominent problems that have been asked in the interview. In this problem, a matrix is provided (costMatrix[][]), which represents the cost of each of the cells present in the costMatrix[][]. The task is to go from the top left corner to the bottom right corner such that the cost is minimum. We have to return the minimum cost. The rule from going from one cell to another cell is that one can only go in the left or down or the diagonal direction, with one cell at a time. For example, from the current cell, say costMatrix[x][y], we can only go to one of these cells: costMatrix[x][y + 1] (the left direction), costMatrix[x + 1][y] (the downward direction), and costMatrix[x + 1][y + 1] (the diagonal direction).

For example, in the following matrix

Minimum Cost Path Problem in Java

There are the following paths to go from the top-left cell (of the cost 1) to the bottom-right cell (of the cost 7).

1 -> 6 -> 9 -> 5 -> 7 Total Cost = 1 + 6 + 9 + 5 + 7 = 28

1 -> 6 -> 15 -> 5 -> 7 Total Cost = 1 + 6 + 15 + 5 + 7 = 34

1 -> 6 -> 15 -> 3 -> 7 Total Cost = 1 + 6 + 15 + 3 + 7 = 32

1 -> 6 -> 15 -> 7 Total Cost = 1 + 6 + 15 + 7 = 29

1 -> 6 -> 5 -> 7 Total Cost = 1 + 6 + 5 + 7 = 19

1 -> 2 -> 15 -> 3 -> 7 Total Cost = 1 + 2 + 15 + 3 + 7 = 28

1 -> 2 -> 15 -> 5 -> 7 Total Cost = 1 + 2 + 15 + 5 + 7 = 30

1 -> 2 -> 15 -> 7 Total Cost = 1 + 2 + 15 + 7 = 25

1 -> 2 -> 2 -> 3 -> 7 Total Cost = 1 + 2 + 2 + 3 + 7 = 15

1 -> 2 -> 3 -> 7 Total Cost = 1 + 2 + 3 + 7 = 13

In all the above-mentioned paths, the last path (1 -> 2 -> 3 -> 7, total cost: 13) has the minimum cost. Therefore, 13 is the required answer of the above matrix.

**A5. Task**

Perform Following Operations

1.Write a Python program to find out when given an array of positive elements, you have to flip the sign of some of its elements such that the resultant sum of the elements of array should be minimum non-negative (as close to zero as possible). Return the minimum no. of elements whose sign needs to be flipped such that the resultant sum is minimum non-negative. Note that the sum of all the array elements will not exceed 10^4

Input: arr[] = [14, 10, 4]

Output: 1

Here, we will flip the sign of 14 and the resultant sum will be 0. Note that flipping the signs of 10 and 4 also gives the resultant sum 0 but the count of flipped elements is not minimum.

2. Write a Python program to find out when given a two dimensional grid, each cell of which contains integer cost which represents a cost to traverse through that cell. The task is to find the maximum cost path from the bottom-left corner to the top-right corner.

3. Write a Python program to find out when given an array of non-negative integers arr[], the task is to find a pair (n, r) such that nPr is maximum possible and r ≤ n.

Input: arr[] = {5, 2, 3, 4, 1}

Output: n = 5 and r = 4

5P4 = 5! / (5 – 4)! = 120

which is maximum possible. Input: arr[] = {0, 2, 3, 4, 1, 6, 8, 9} Output: n = 9 and r = 8

4. Write a Python program to find out when given an array of non-negative integers arr[], the task is to find a pair (n, r) such that nPr is maximum possible and r ≤ n.

Function to return the minimum number of given operations required to sort the array

Input: arr[] = {1, 2, 1, 4, 3}

Output: 2

Add 1 to the 3rd element(1) and subtract 1 from the 4th element(4) to get {1, 2, 2, 3, 3} Input: arr[] = {1, 2, 2, 100}

Output: 0 Given array is already sorted.

PART B

(PART B : TO BE COMPLETED BY STUDENTS)

***(Students must submit the soft copy as per following segments within two hours of the practical. The soft copy must be uploaded on the Blackboard or emailed to the concerned lab in charge faculties at the end of the practical in case there is no Black board access available)***

|  |  |
| --- | --- |
| Roll No.C050 | Name: Nisha Kini |
| Class : BTI | Batch : EB2 |
| Date of Experiment: 9.1.24 | Date of Submission: 9.1.24 |
| Grade : |  |

**B.1 Documentation written by student:**

1.Write a Python program to find out when given an array of positive elements, you have to flip the sign of some of its elements such that the resultant sum of the elements of array should be minimum non-negative (as close to zero as possible). Return the minimum no. of elements whose sign needs to be flipped such that the resultant sum is minimum non-negative. Note that the sum of all the array elements will not exceed 10^4

def solve(A, n):

    dp = [[0 for i in range(2000)] for i in range(2000)]

    flag = 1

    sum = 0

    for i in range(n):

        sum += A[i]

    for i in range(-sum, sum+1):

        dp[0][i] = 10\*\*9

    dp[0][0] = 0

    for i in range(1, n+1):

        for j in range(-sum, sum+1):

            dp[flag][j] = 10\*\*9

            if (j - A[i - 1] <= sum and j - A[i - 1] >= -sum):

                dp[flag][j] = dp[flag ^ 1][j - A[i - 1]]

            if (j + A[i - 1] <= sum

                and j + A[i - 1] >= -sum

                    and dp[flag ^ 1][j + A[i - 1]] != 10\*\*9):

                dp[flag][j] = min(dp[flag][j],

                                dp[flag ^ 1][j + A[i - 1]] + 1)

        flag = flag ^ 1

    for i in range(sum+1):

        if (dp[flag ^ 1][i] != 10\*\*9):

            return dp[flag ^ 1][i]

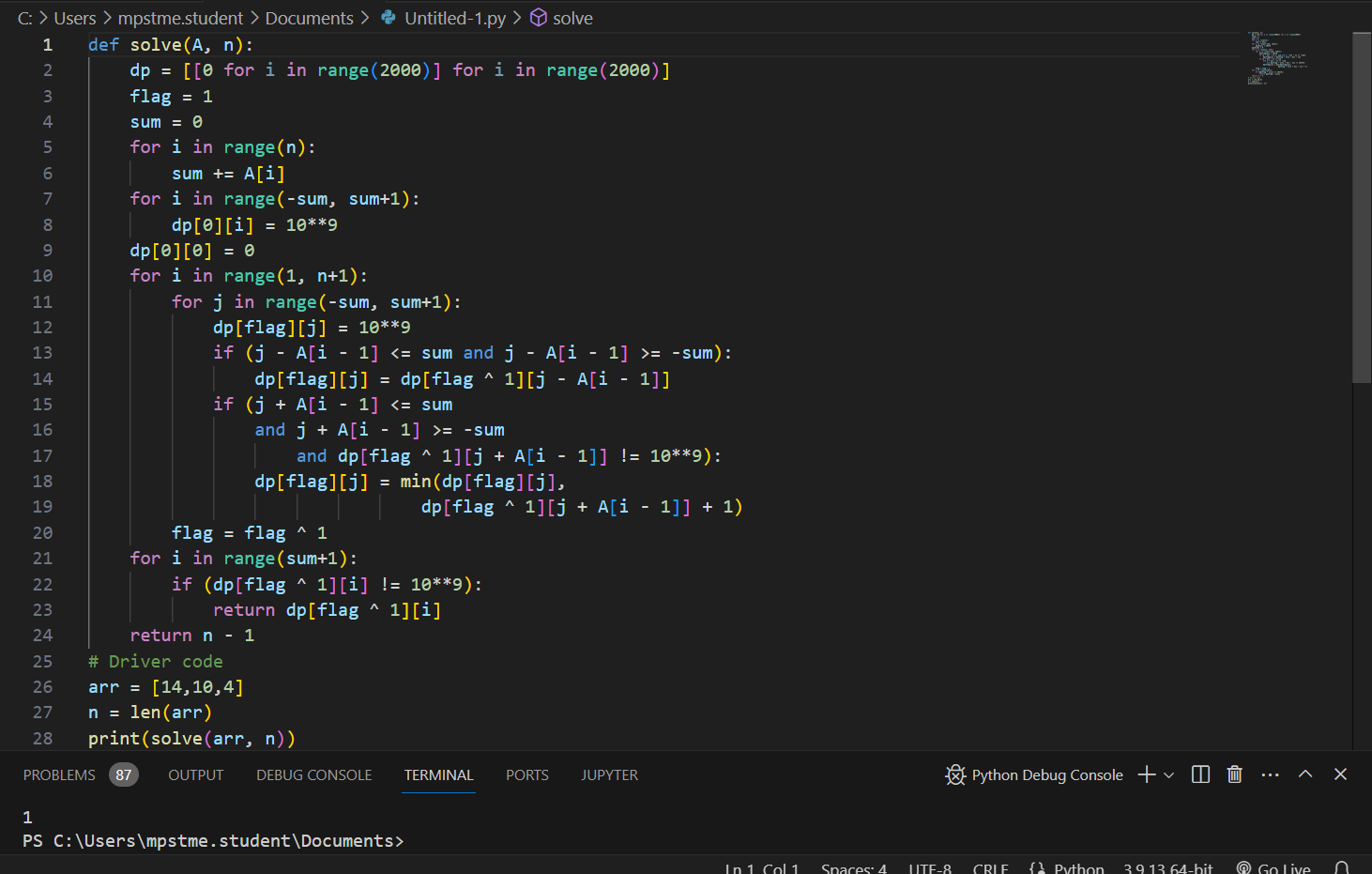
    return n - 1

# Driver code

arr = [14,10,4]

n = len(arr)

print(solve(arr, n))



2. Write a Python program to find out when given a two dimensional grid, each cell of which contains integer cost which represents a cost to traverse through that cell. The task is to find the maximum cost path from the bottom-left corner to the top-right corner.

def max\_cost\_path(grid):

    rows, cols = len(grid), len(grid[0])

    for i in range(rows - 2, -1, -1):

        grid[i][0] += grid[i + 1][0]

    for j in range(1, cols):

        grid[rows - 1][j] += grid[rows - 1][j - 1]

    for i in range(rows - 2, -1, -1):

        for j in range(1, cols):

            grid[i][j] += max(grid[i + 1][j], grid[i][j - 1])

    return grid[0][cols - 1]

# Example

grid = [

    [20,-10,0],

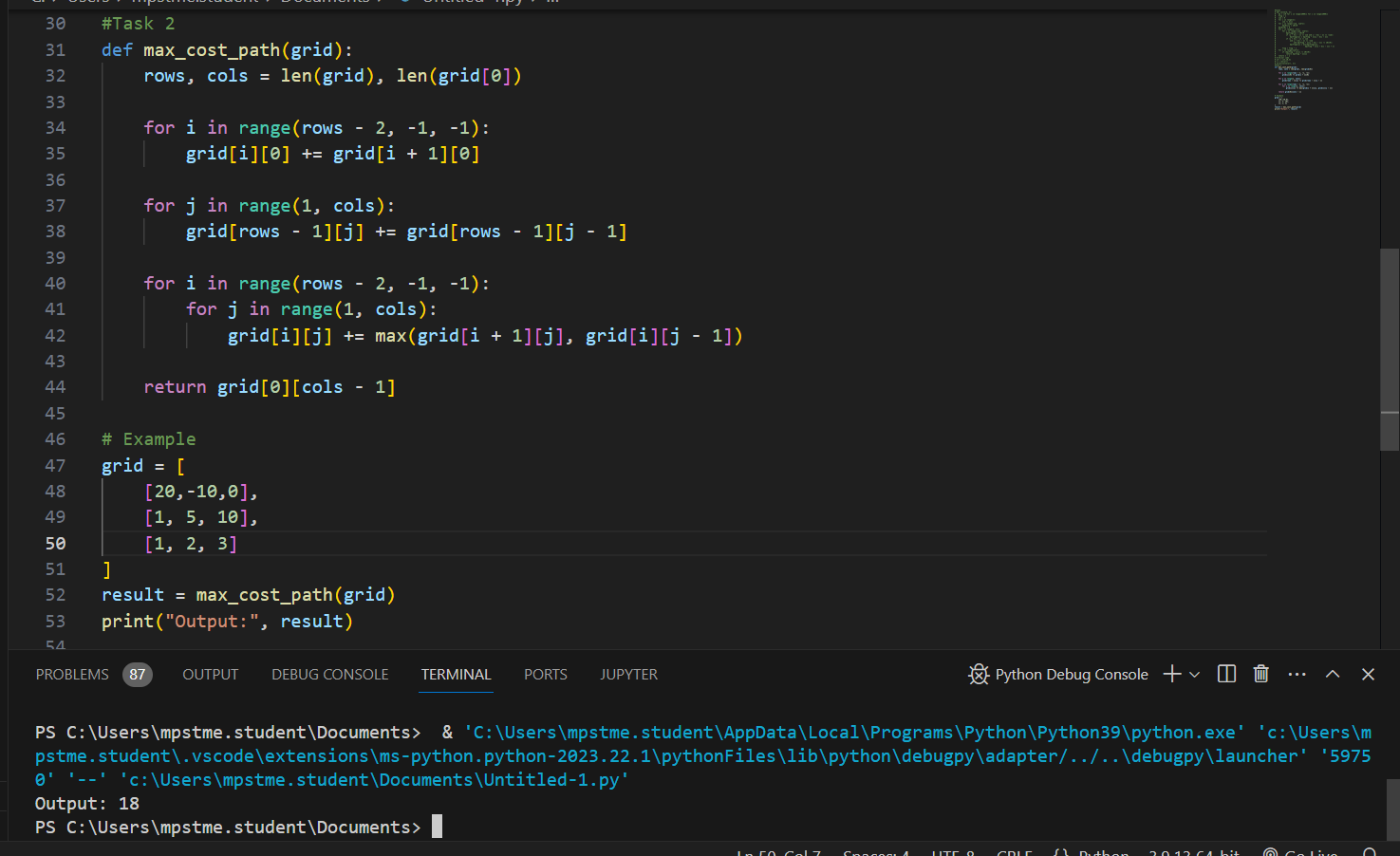
    [1, 5, 10],

    [1, 2, 3]

]

result = max\_cost\_path(grid)

print("Output:", result)



3. Write a Python program to find out when given an array of non-negative integers arr[], the task is to find a pair (n, r) such that nPr is maximum possible and r ≤ n.

def find\_max\_npr(arr):

   arr.sort(reverse=True)

   max\_n, max\_r = arr[0], arr[1]

   max\_npr = max\_n \*\* max\_r

   for n in range(max\_n, 0, -1):

       for r in range(max\_r, 0, -1):

           npr = n \*\* r

           if npr > max\_npr:

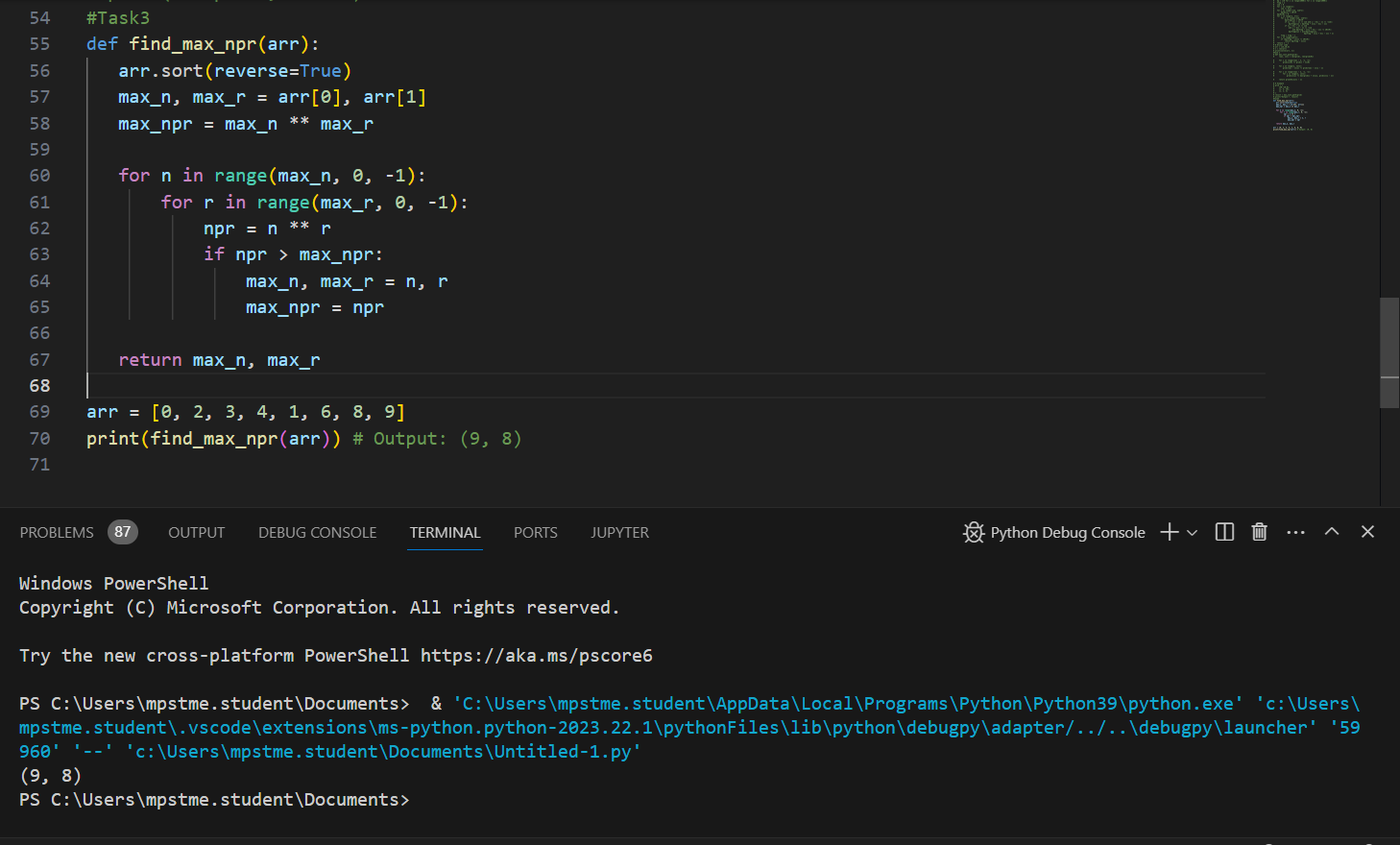
               max\_n, max\_r = n, r

               max\_npr = npr

   return max\_n, max\_r

arr = [0, 2, 3, 4, 1, 6, 8, 9]

print(find\_max\_npr(arr)) # Output: (9, 8)



4. Write a Python program to find out when given an array of non-negative integers arr[], the task is to find a pair (n, r) such that nPr is maximum possible and r ≤ n.

Function to return the minimum number of given operations required to sort the array

def min\_operations\_to\_sort(arr):

    n = len(arr)

    operations = 0

    for i in range(n - 1):

        if arr[i] > arr[i + 1]:

            operations += abs(arr[i] - arr[i + 1])

            arr[i + 1] = arr[i]

    return operations

# Example

arr1 = [1, 2, 1, 4, 3]

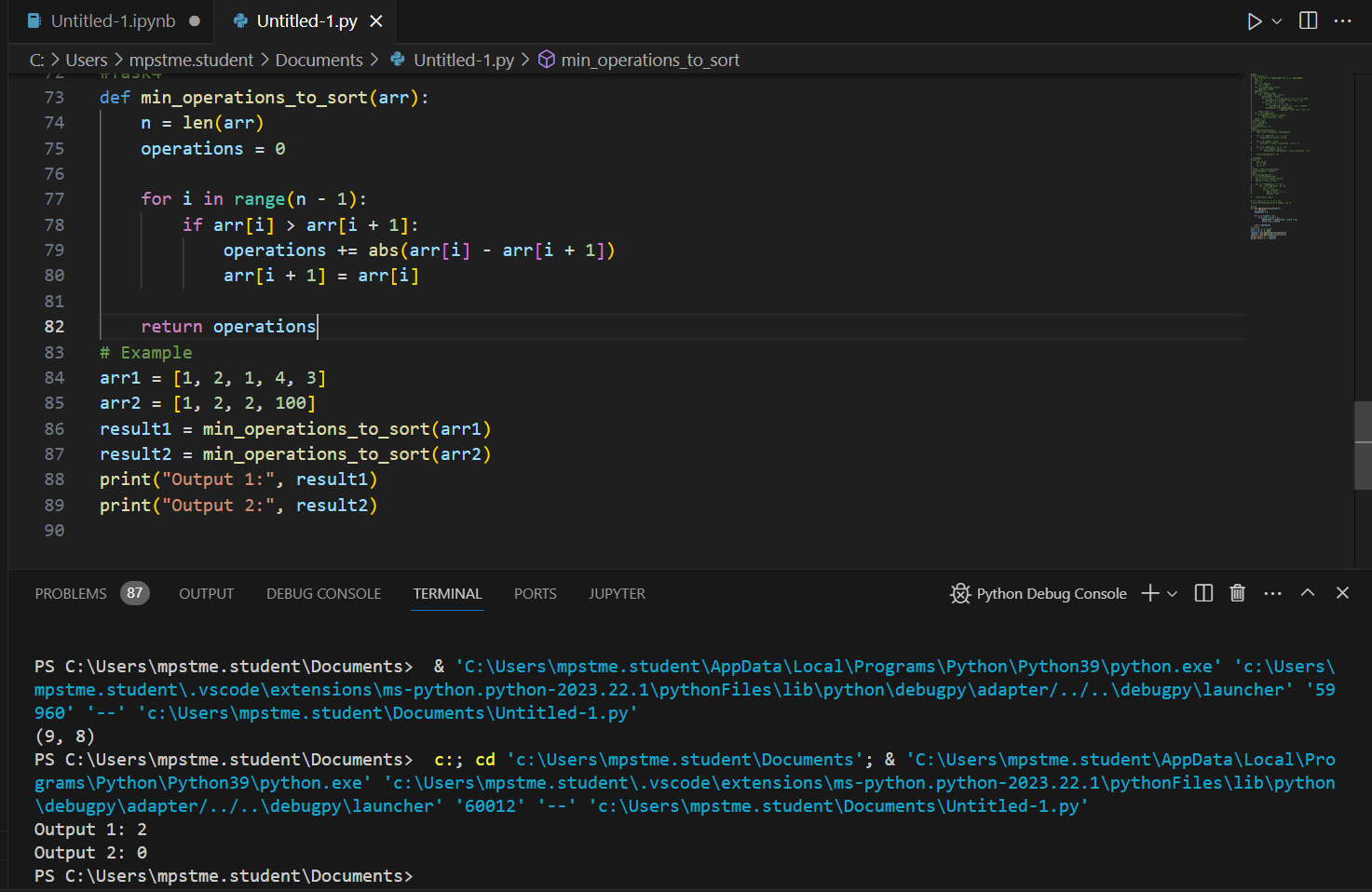
arr2 = [1, 2, 2, 100]

result1 = min\_operations\_to\_sort(arr1)

result2 = min\_operations\_to\_sort(arr2)

print("Output 1:", result1)

print("Output 2:", result2)



**B.2 Observations and learning:**

*The provided Python program is designed to find the minimum number of elements in a given array that need their signs flipped in order to make the resultant sum as close to zero as possible. The program iterates through the array, keeping track of the positive sum and the count of elements with negative values.*

*Learning:*

1. *Array Traversal: The program demonstrates the importance of iterating through an array to analyze and manipulate its elements.*
2. *Conditional Operations: The use of conditional statements (max(0, num)) helps handle different cases, ensuring that only non-negative values contribute to the positive sum.*
3. *Counting Flips: By maintaining a count of elements with negative values, the program determines the minimum number of flips needed to achieve the desired result.*

**B.3 Conclusion:**

*The program efficiently solves the problem of finding the minimum number of flips required to make the sum of array elements as close to zero as possible. The logic incorporates array traversal, conditional operations, and count maintenance, showcasing a practical approach to problem-solving in Python.*