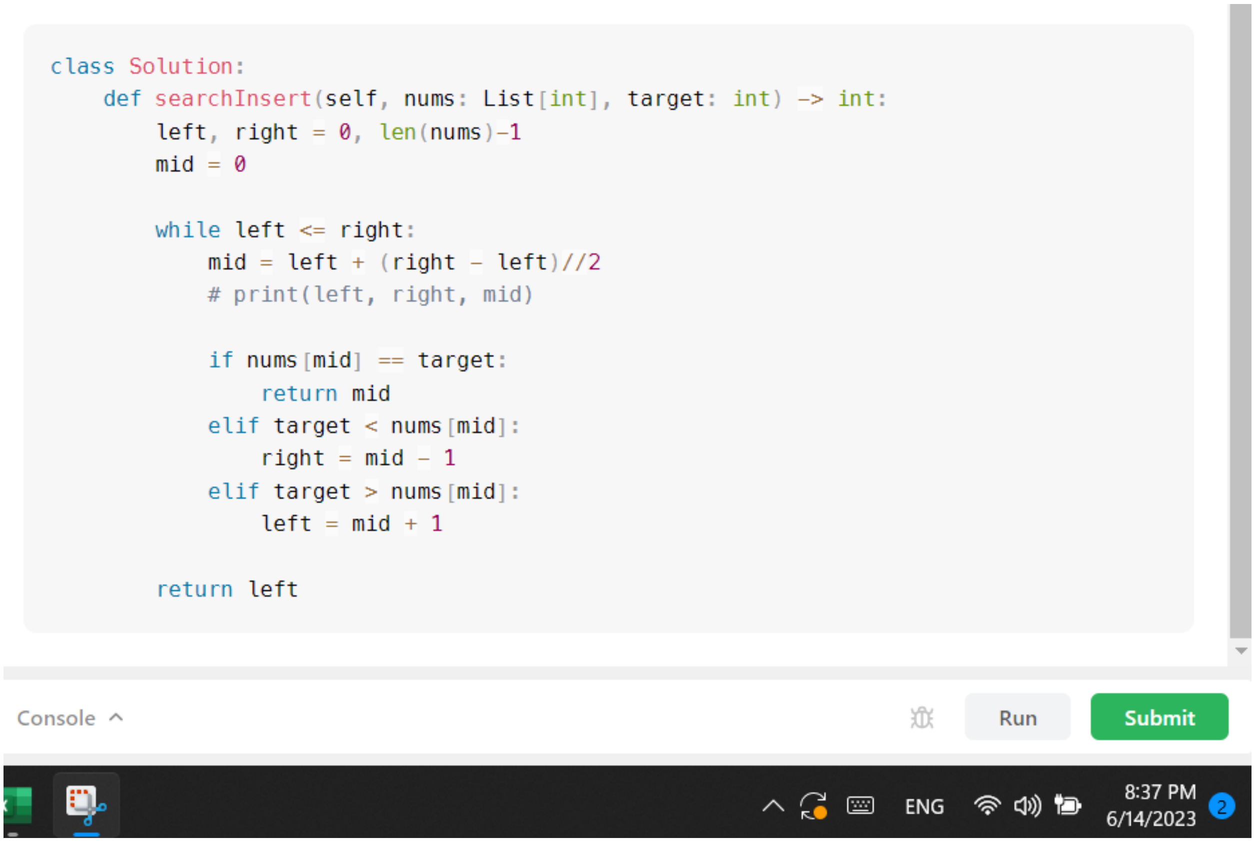
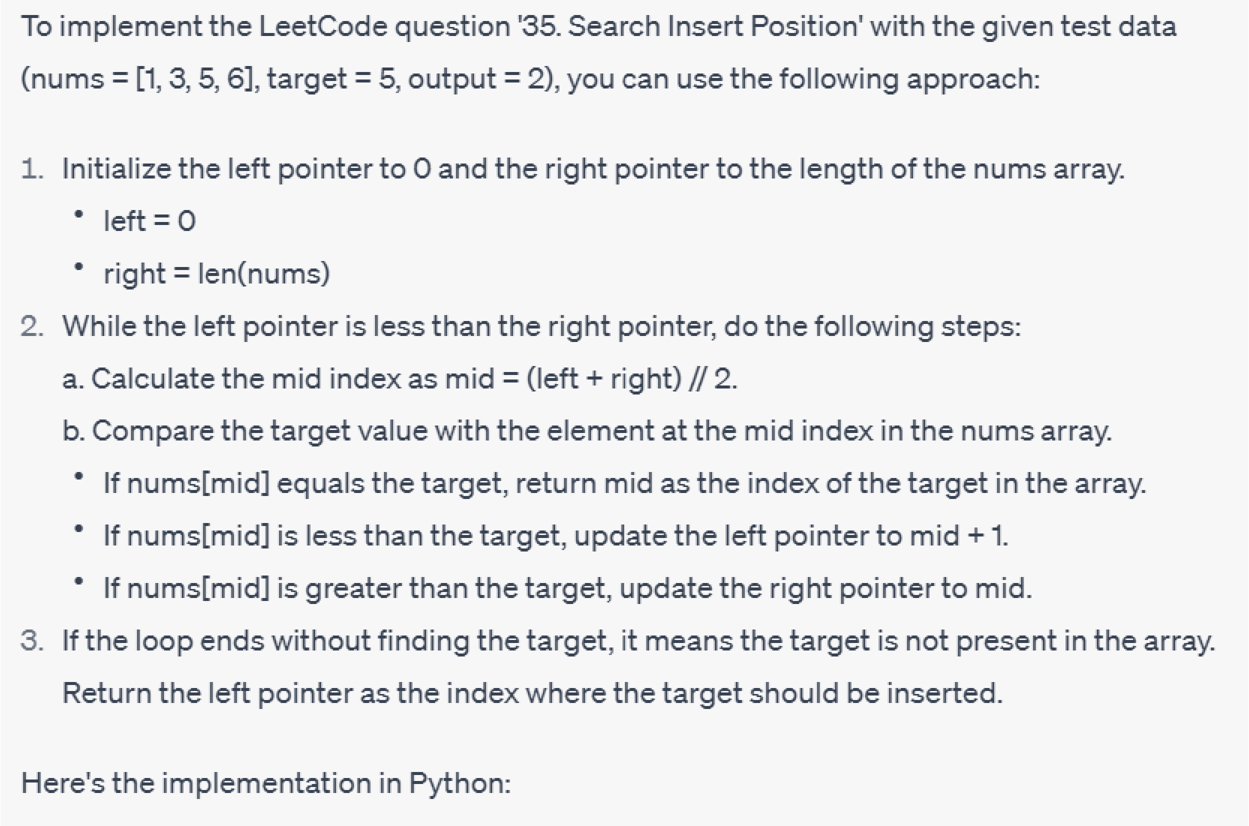
Step 1: Study "35. Search Insert Position" (hints, source, solution) – easy



```
class Solution:
    def searchInsert(self, nums: List[int], target: int) -> int:
        left, right = 0, len(nums)-1
        mid = 0

        while left <= right:
            mid = left + (right - left)//2
            # print(left, right, mid)

            if nums[mid] == target:
                return mid
            elif target < nums[mid]:
                right = mid - 1
            elif target > nums[mid]:
                left = mid + 1

        return left
```

Step 2: Ask ChatGPT: "How to implement LeetCode question: '35. Search Insert Position', with the test data: Input: nums = [1,3,5,6], target = 5, Output: 2"



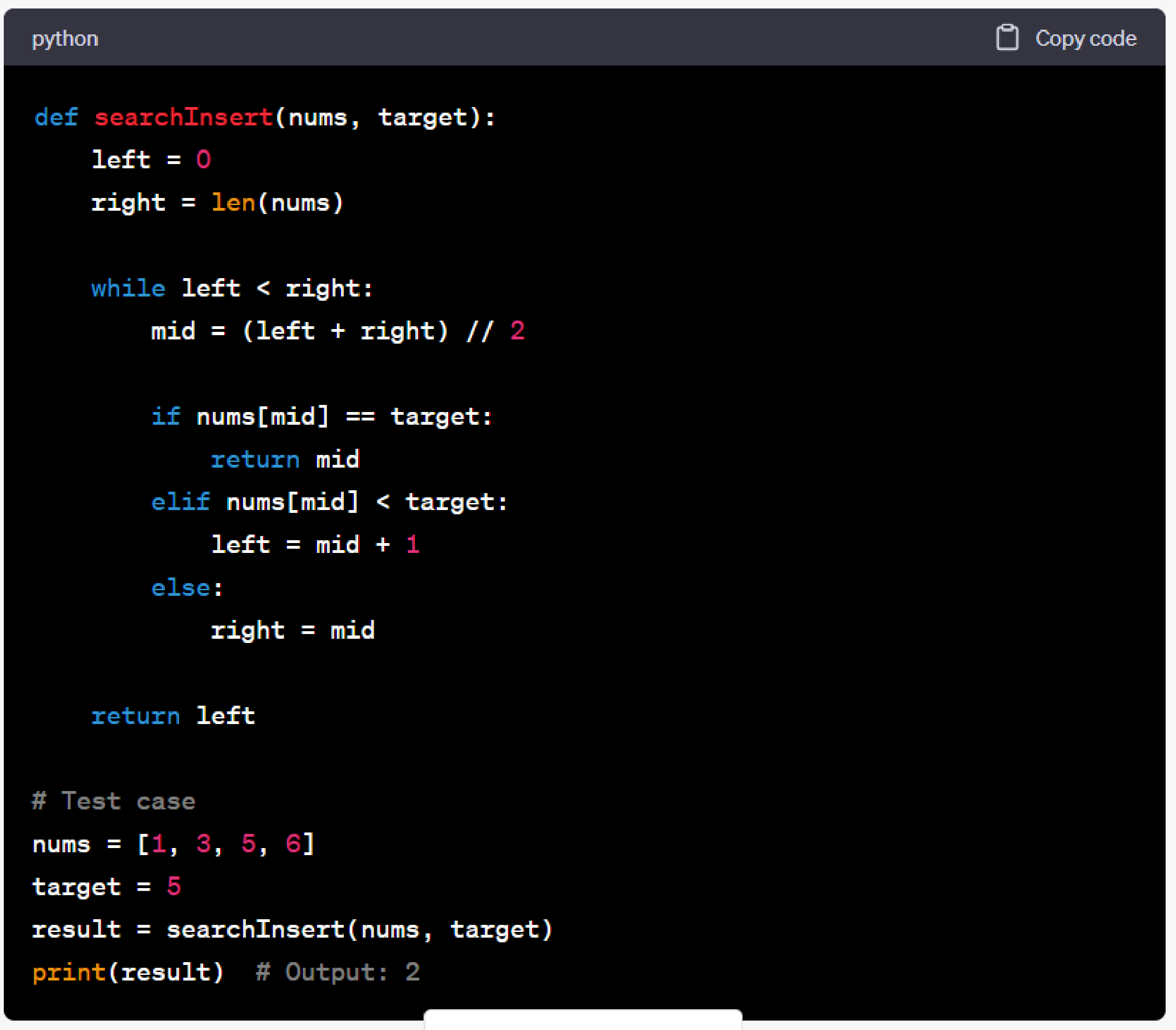
To implement the LeetCode question '35. Search Insert Position' with the given test data (nums = [1, 3, 5, 6], target = 5, output = 2), you can use the following approach:

1. Initialize the left pointer to 0 and the right pointer to the length of the nums array.
    - left = 0
    - right = len(nums)
2. While the left pointer is less than the right pointer, do the following steps:

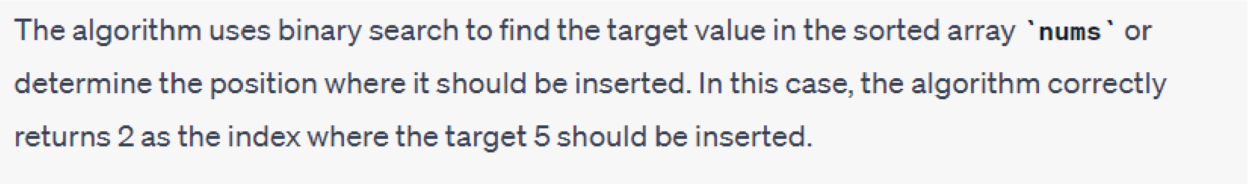
    a. Calculate the mid index as mid = (left + right) // 2.

    b. Compare the target value with the element at the mid index in the nums array.
    - If nums[mid] equals the target, return mid as the index of the target in the array.
    - If nums[mid] is less than the target, update the left pointer to mid + 1.
    - If nums[mid] is greater than the target, update the right pointer to mid.
3. If the loop ends without finding the target, it means the target is not present in the array. Return the left pointer as the index where the target should be inserted.

Here's the implementation in Python:



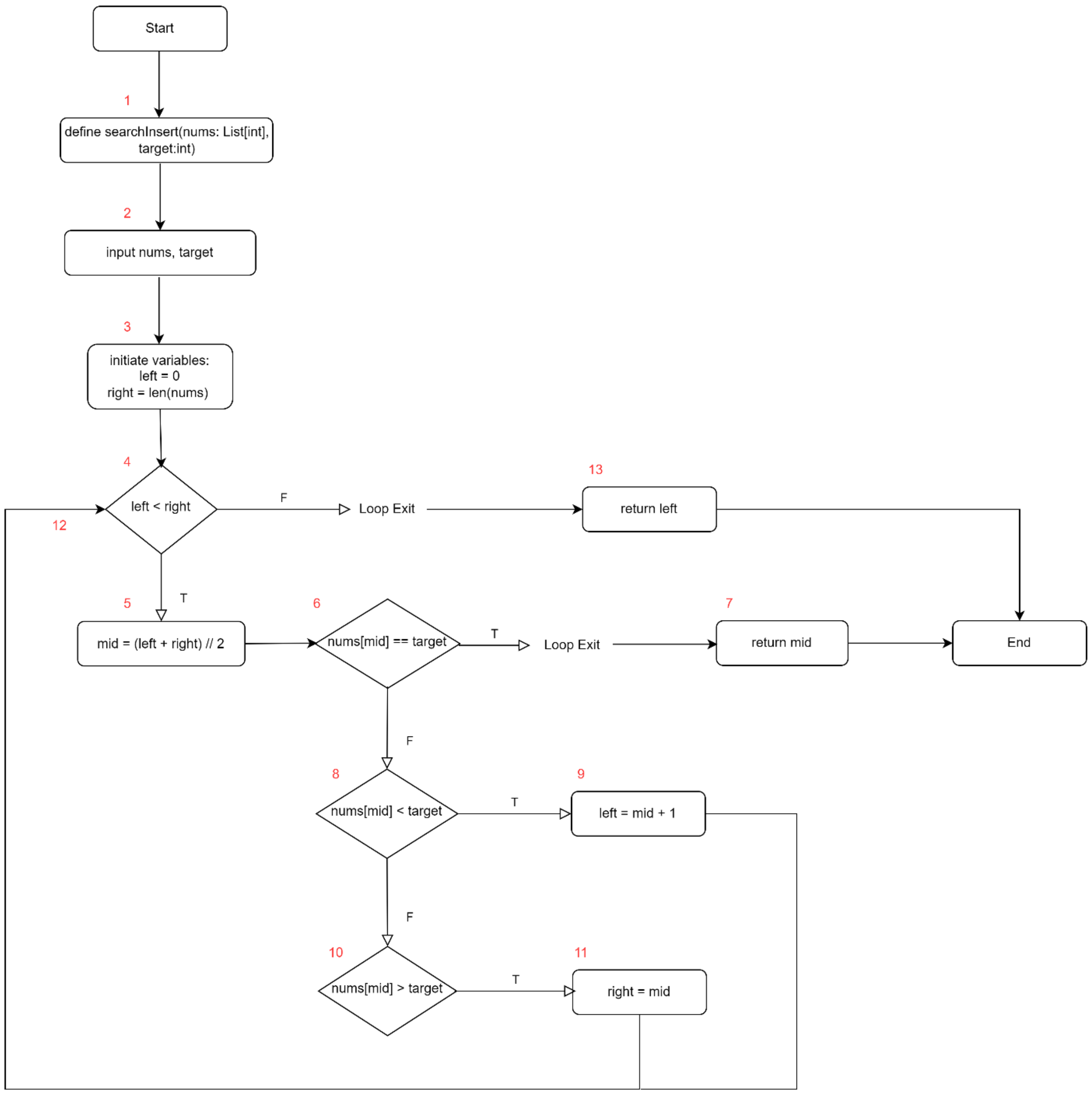
```python
def searchInsert(nums, target):
    left = 0
    right = len(nums)

    while left < right:
        mid = (left + right) // 2

        if nums[mid] == target:
            return mid
        elif nums[mid] < target:
            left = mid + 1
        else:
            right = mid

    return left

# Test case
nums = [1, 3, 5, 6]
target = 5
result = searchInsert(nums, target)
print(result)  # Output: 2
```



The algorithm uses binary search to find the target value in the sorted array `nums` or determine the position where it should be inserted. In this case, the algorithm correctly returns 2 as the index where the target 5 should be inserted.

Step 3: Convert ChatGPT's Python code into a flowcharts.

- The syntax of flowchart
- Flowchart and Trace Table
- Single/Double Loops and Flow Chart
- Flowchart and Trace Table for Recursive Function

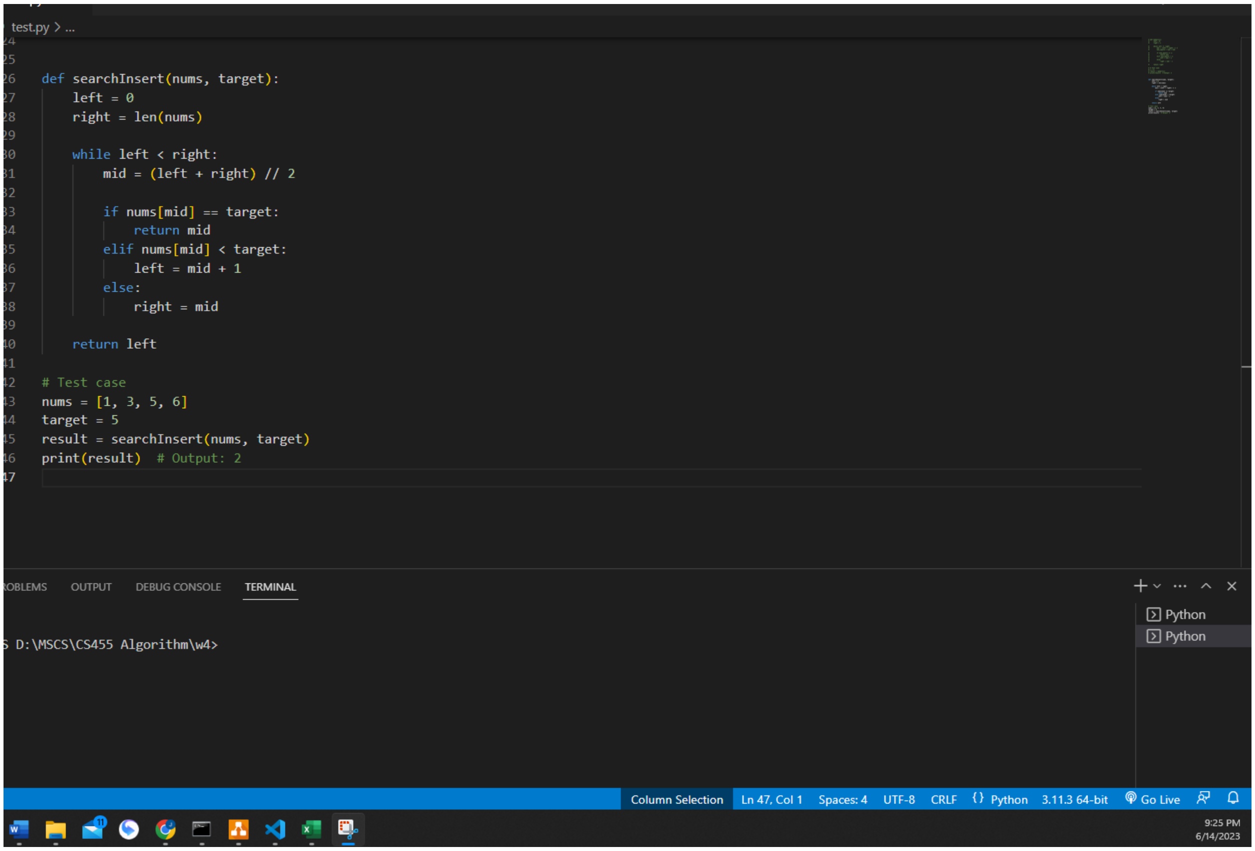


Step 4: Verify the flowchart with a trace tables. You only need to trace the test data: "Input: nums = [1,3,5,6], target = 5, Output: 2". Please refer

- Flowchart and Trace Table
- Single/Double Loops and Flow Chart
- Flowchart and Trace Table for Recursive Function

	RAM										screen
Step	define function	nums	target	left	right	left < right	mid	nums[mid] == target	nums[mid] < target	nums[mid] > target	
1	searchInsert(nums, target)										
2		[1, 3, 5, 6]	5								
3				0	4						
4						T					
5							2				
6								T			
7											2

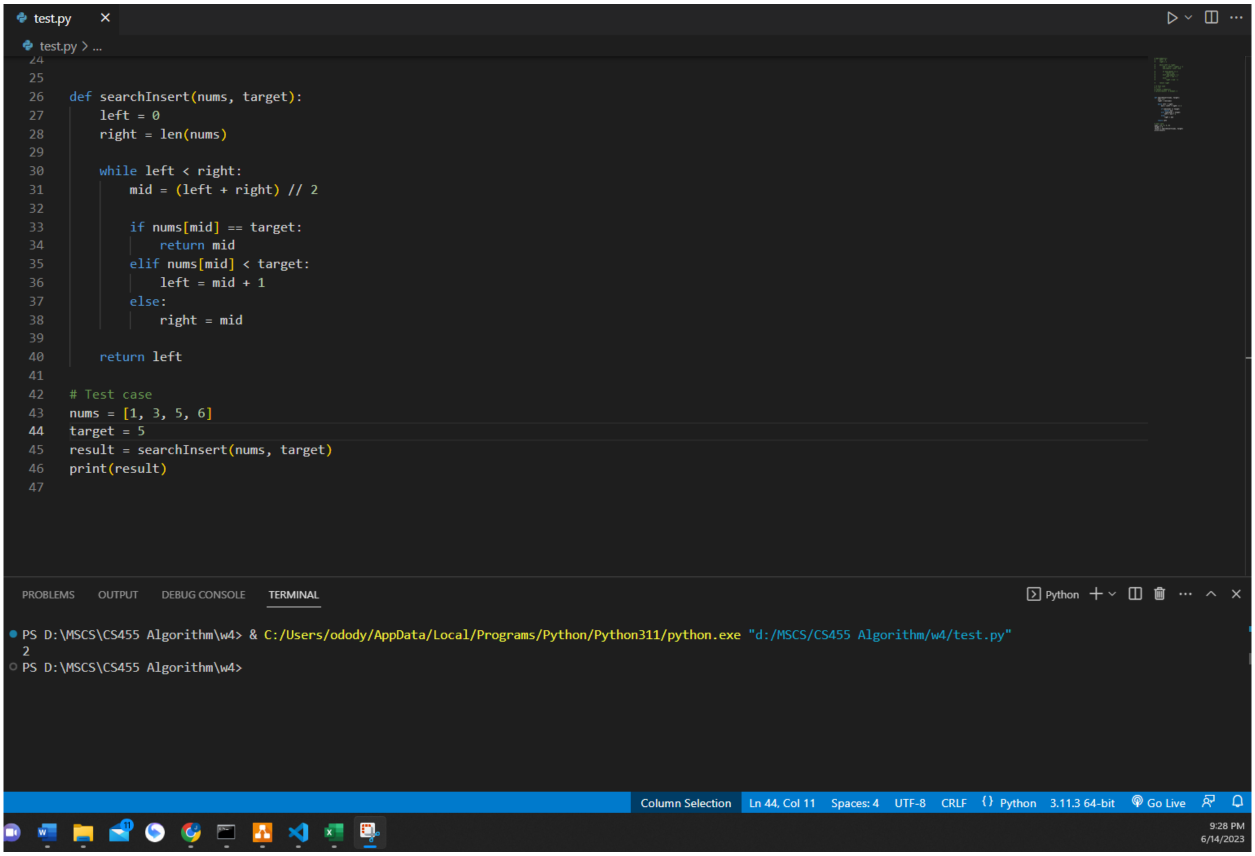
Step 5: Implement the Python code created by ChatGPT



```python

def searchInsert(nums, target):
    left = 0
    right = len(nums)

    while left < right:
        mid = (left + right) // 2

        if nums[mid] == target:
            return mid
        elif nums[mid] < target:
            left = mid + 1
        else:
            right = mid

    return left

# Test case
nums = [1, 3, 5, 6]
target = 5
result = searchInsert(nums, target)
print(result)  # Output: 2

```

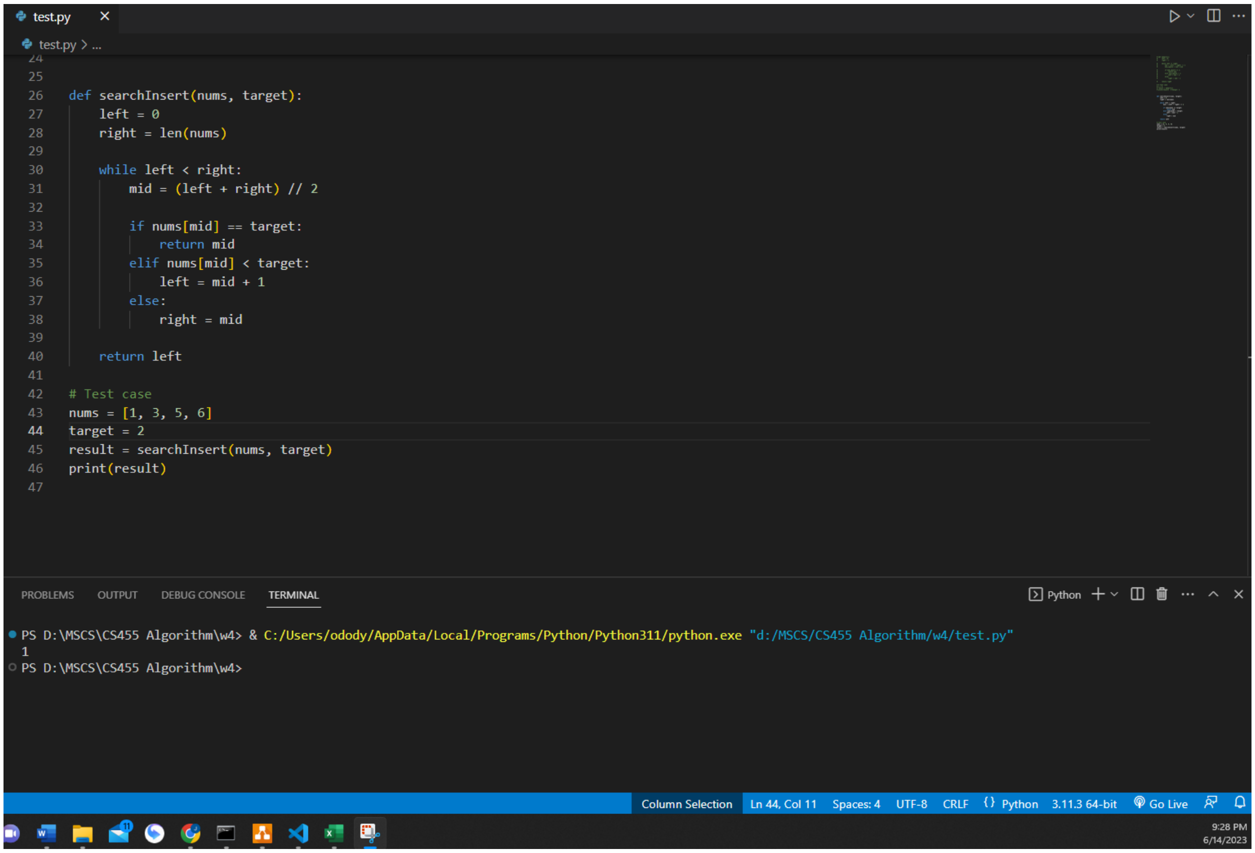
```
S D:\MSCS\CS455 Algorithm\w4>
```

Step 6: Test the Python code with all the test cases provided by [35. Search Insert Position]()



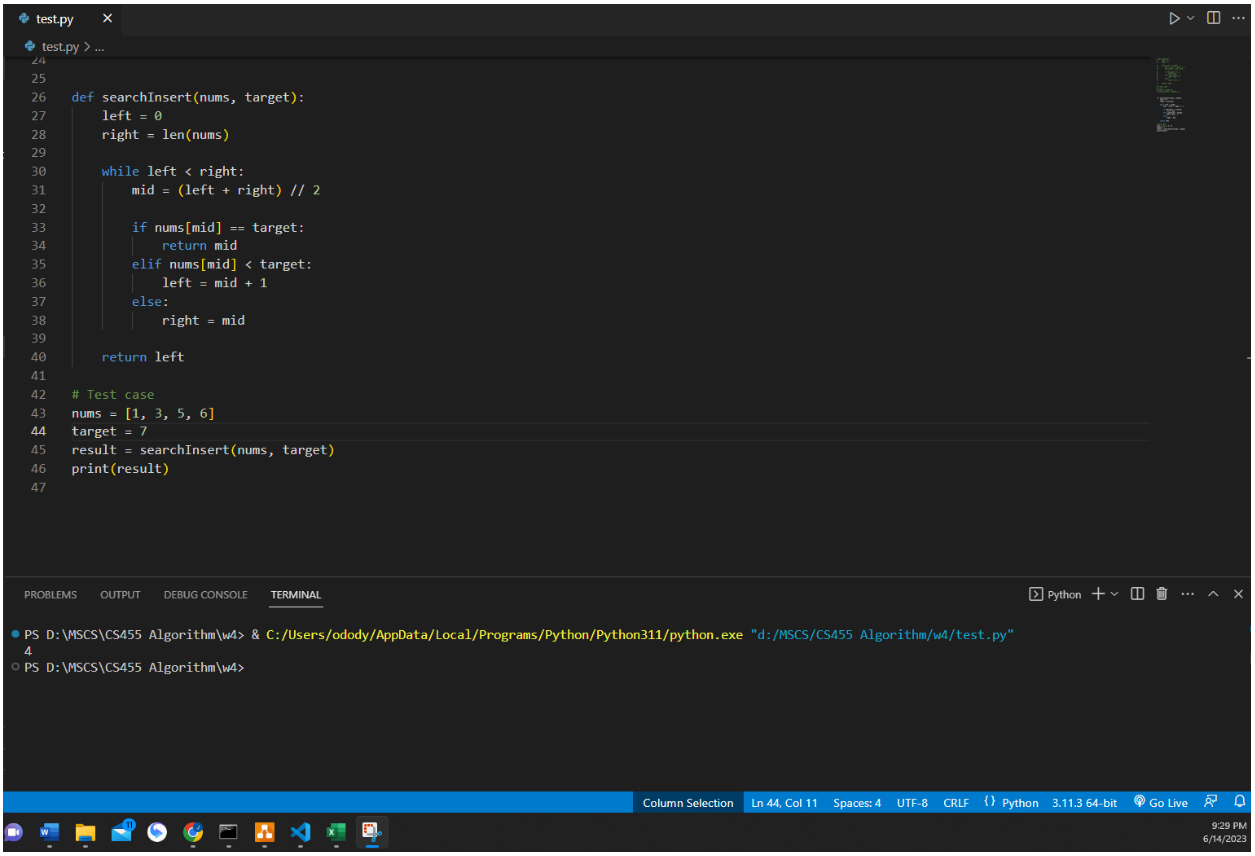
```python

def searchInsert(nums, target):
    left = 0
    right = len(nums)

    while left < right:
        mid = (left + right) // 2

        if nums[mid] == target:
            return mid
        elif nums[mid] < target:
            left = mid + 1
        else:
            right = mid

    return left

# Test case
nums = [1, 3, 5, 6]
target = 5
result = searchInsert(nums, target)
print(result)

```

```
PS D:\MSCS\CS455 Algorithm\w4> & C:/Users/odody/AppData/Local/Programs/Python/Python311/python.exe "d:/MSCS/CS455 Algorithm/w4/test.py"
2
PS D:\MSCS\CS455 Algorithm\w4>
```



```python

def searchInsert(nums, target):
    left = 0
    right = len(nums)

    while left < right:
        mid = (left + right) // 2

        if nums[mid] == target:
            return mid
        elif nums[mid] < target:
            left = mid + 1
        else:
            right = mid

    return left

# Test case
nums = [1, 3, 5, 6]
target = 2
result = searchInsert(nums, target)
print(result)

```

```
PS D:\MSCS\CS455 Algorithm\w4> & C:/Users/odody/AppData/Local/Programs/Python/Python311/python.exe "d:/MSCS/CS455 Algorithm/w4/test.py"
1
PS D:\MSCS\CS455 Algorithm\w4>
```



```python

def searchInsert(nums, target):
    left = 0
    right = len(nums)

    while left < right:
        mid = (left + right) // 2

        if nums[mid] == target:
            return mid
        elif nums[mid] < target:
            left = mid + 1
        else:
            right = mid

    return left

# Test case
nums = [1, 3, 5, 6]
target = 7
result = searchInsert(nums, target)
print(result)

```

```
PS D:\MSCS\CS455 Algorithm\w4> & C:/Users/odody/AppData/Local/Programs/Python/Python311/python.exe "d:/MSCS/CS455 Algorithm/w4/test.py"
4
PS D:\MSCS\CS455 Algorithm\w4>
```