ASSIGNMENT-8.4(AI)

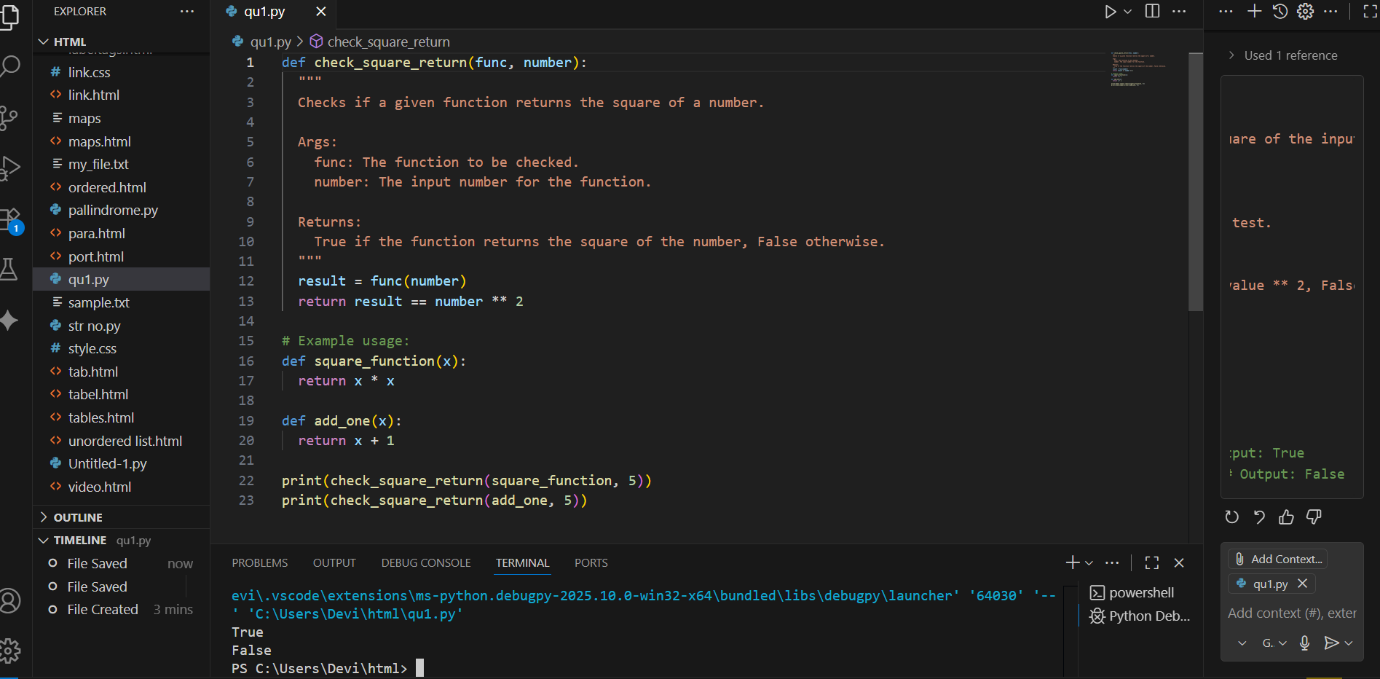
# NAME:DEVI PRIYA.G

2403A52067

BATCH-04

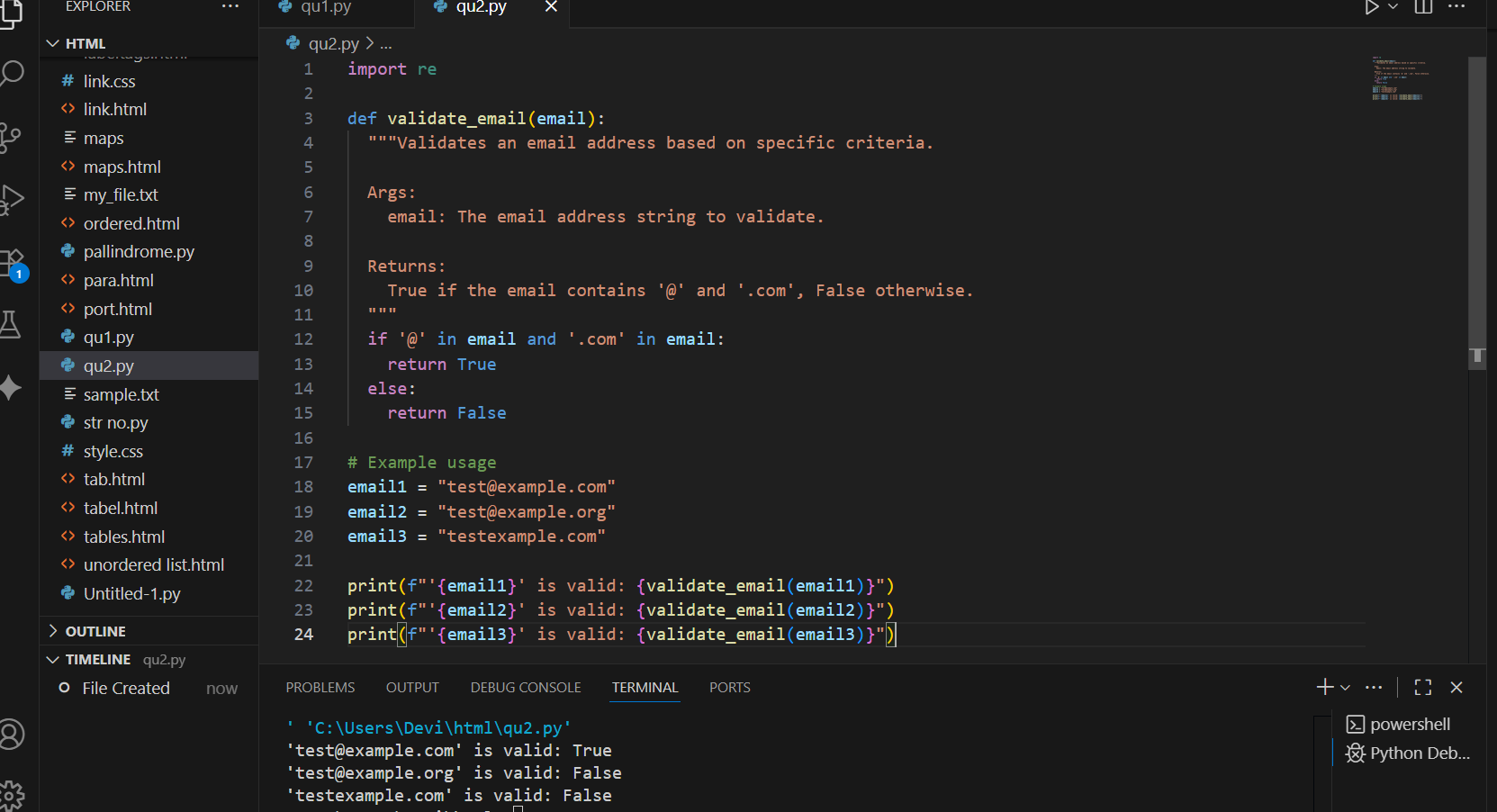
## TASK-1

PROMPT: Write a test case to check if a function returns the square of a number.



* OBSERVATION: **import unittest**: This line imports the unittest module, which provides tools for writing and running tests.
* **class TestPalindromeChecker(unittest.TestCase):**: This defines a test class named TestPalindromeChecker that inherits from unittest.TestCase. Test classes are containers for test methods.
* **if \_\_name\_\_ == '\_\_main\_\_':**: This is a standard Python construct that checks if the script is being run directly.
* **unittest.main(argv=['first-arg-is-ignored'], exit=False)**: This line runs the tests when the script is executed. argv=['first-arg-is-ignored'] and exit=False are often used in environments like Colab to prevent issues with command-line arguments and to allow the script to continue running after the tests.

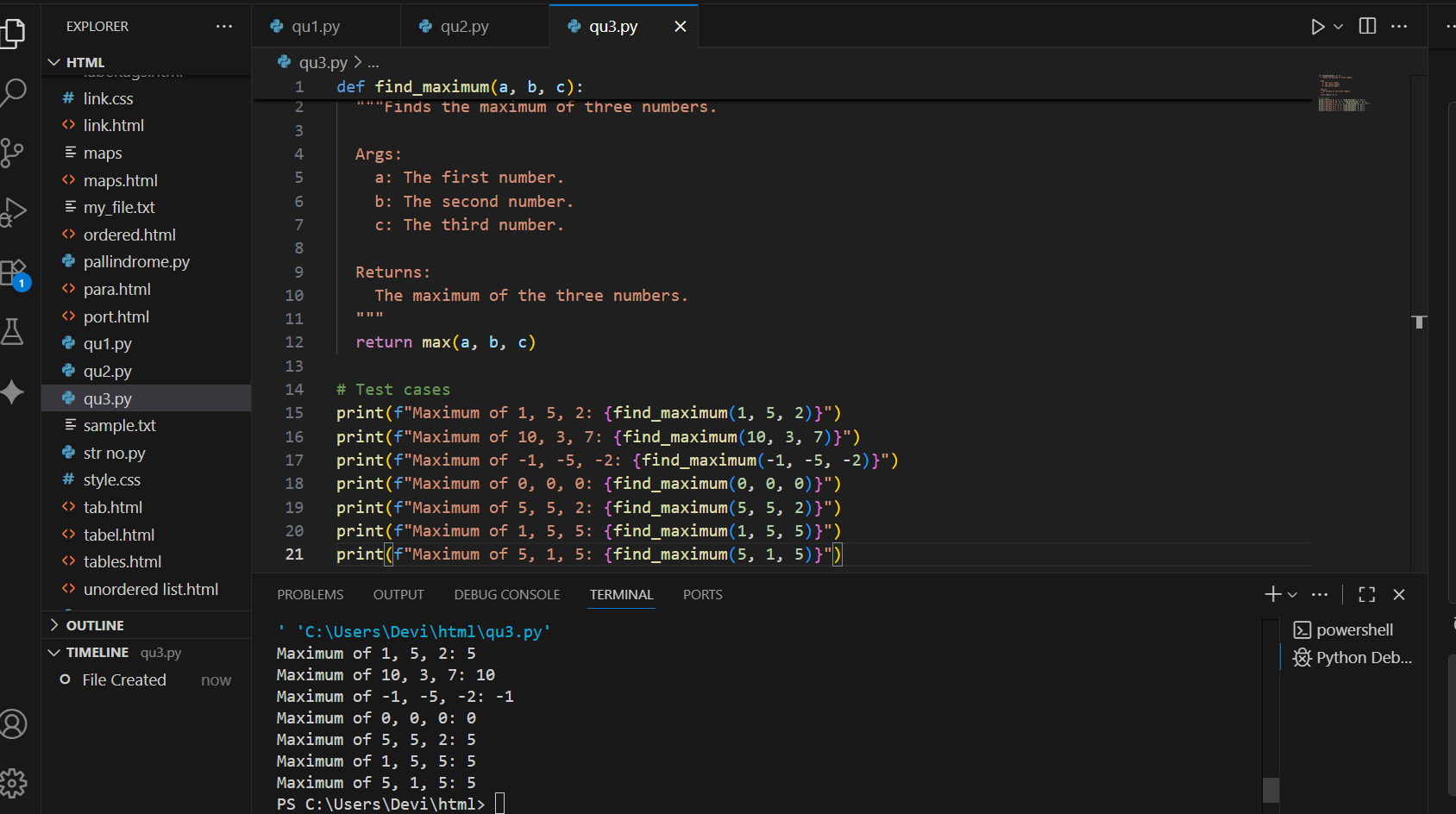
TASK-2

PROMPT: Create test cases to validate an email address (e.g., contains @ and .com).  


* OBSERVATION: **import unittest**: This line imports the unittest module, which is used for writing and running tests.
* **def is\_valid\_email(email):**: This defines a placeholder function is\_valid\_email that will eventually contain the logic to check if an email address is valid. For now, it just has a pass statement.
* **class TestEmailValidation(unittest.TestCase):**: This defines a test class named TestEmailValidation that inherits from unittest.TestCase. This class will contain various test methods to check different email validation scenarios.
* **test\_valid\_email(self)**: This test method checks if the is\_valid\_email function correctly identifies valid email addresses using self.assertTrue().
* **unittest.main(argv=['first-arg-is-ignored'], exit=False)**: This line runs the defined unit tests. The arguments are included to make it compatible with environments like Colab.

TASK-3:

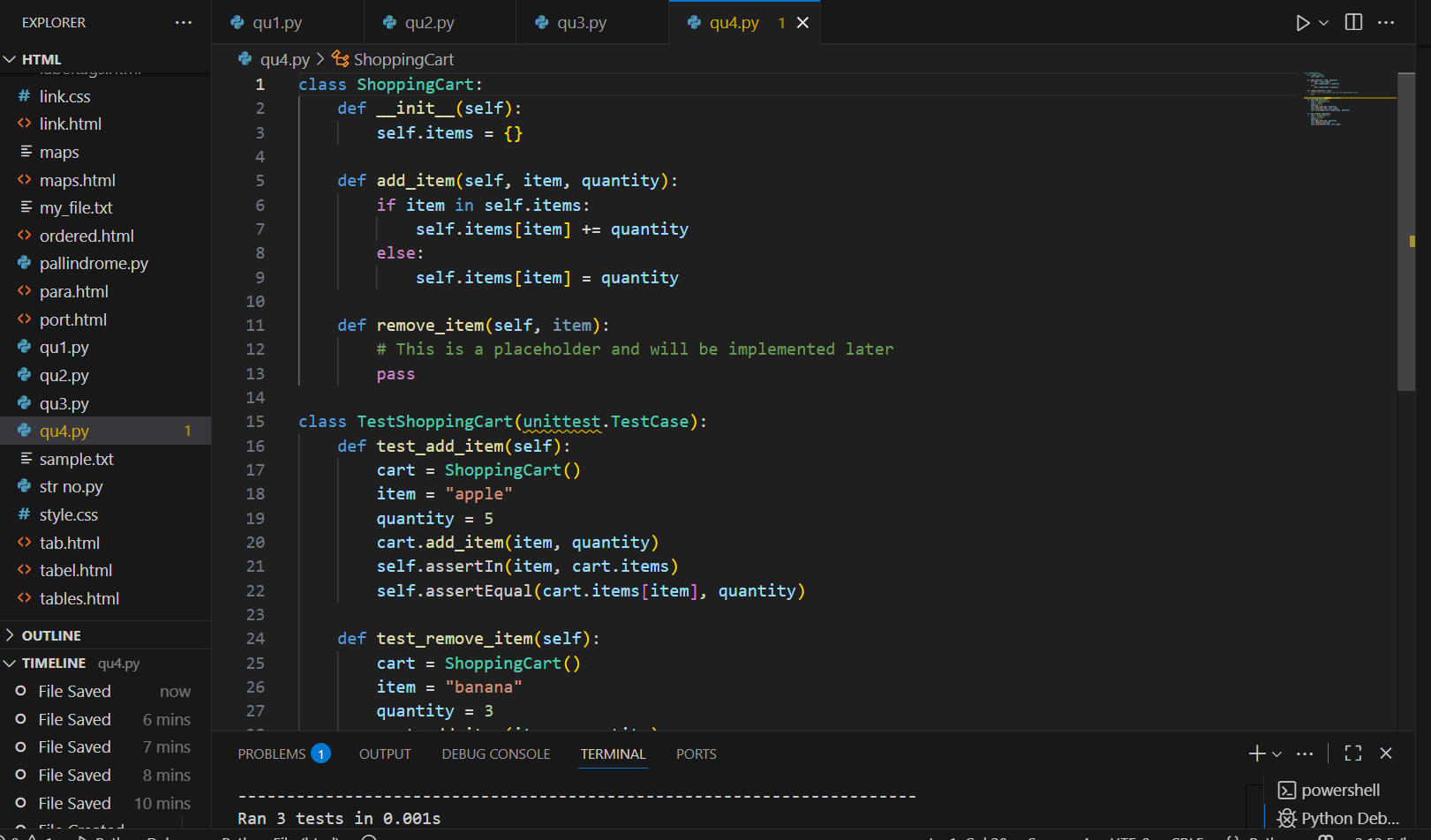
PROMPT: Write test cases for a function that returns the maximum of three numbers.

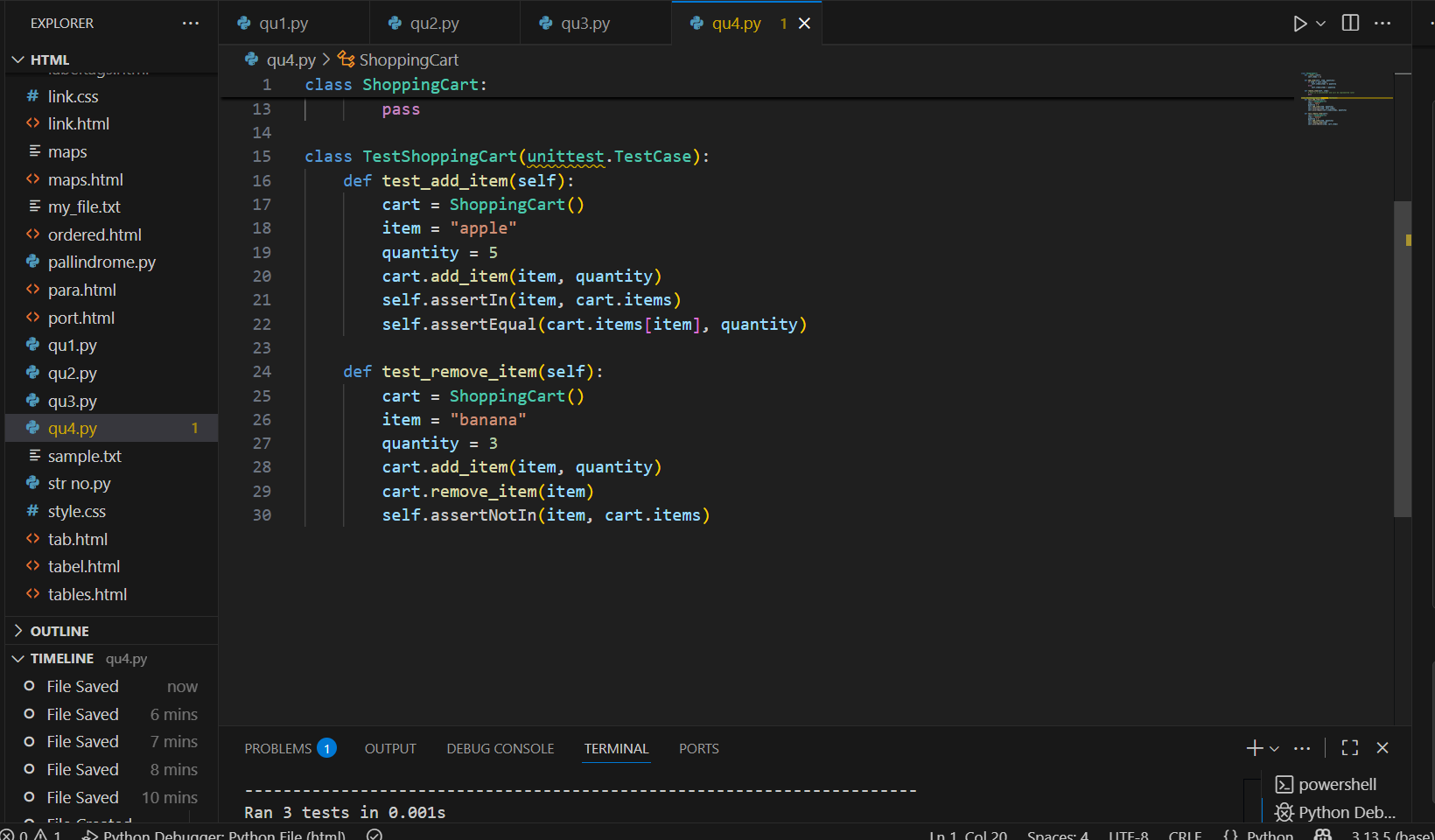


* OBSERVATION: **import unittest**: This line imports the unittest module, which is used for writing and running tests.
* **def find\_maximum(a, b, c):**: This defines a placeholder function find\_maximum that will eventually contain the logic to find the maximum of three numbers. For now, it just has a pass statement.
* **class TestFindMaximum(unittest.TestCase):**: This defines a test class named TestFindMaximum that inherits from unittest.TestCase. This class will contain various test methods to check different scenarios for finding the maximum.
* **test\_with\_mixed\_numbers(self)**: This test includes a mix of positive, negative, and zero to check various cases.
* **if \_\_name\_\_ == '\_\_main\_\_':**: This standard Python construct ensures the code inside this block only runs when the script is executed directly.
* **unittest.main(argv=['first-arg-is-ignored'], exit=False)**: This line runs the defined unit tests. The arguments are included to make it compatible with environments like Colab.

TASK-4:

PROMPT: Use TDD to write a shopping cart class with methods to add, remove, and get total price

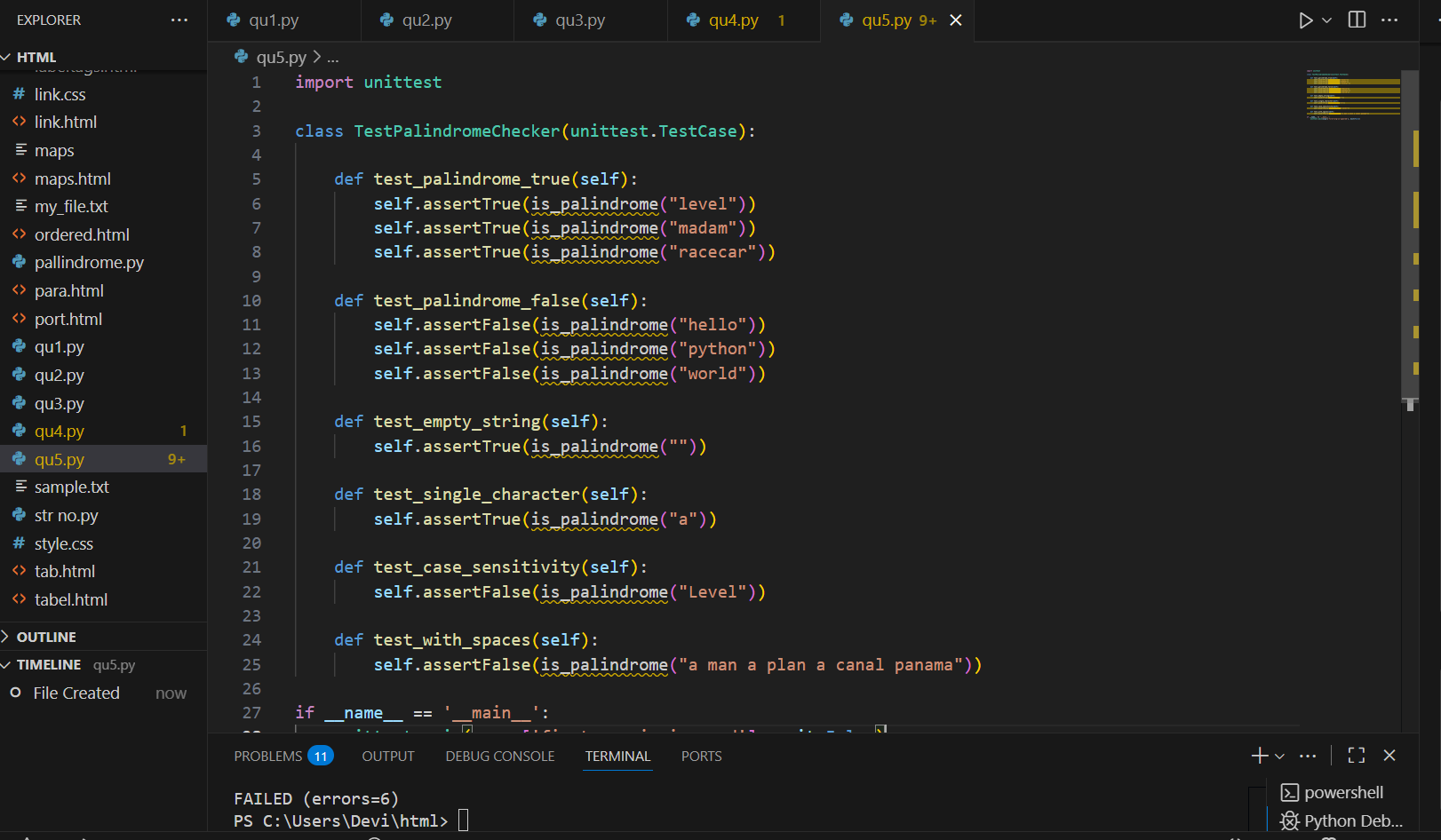


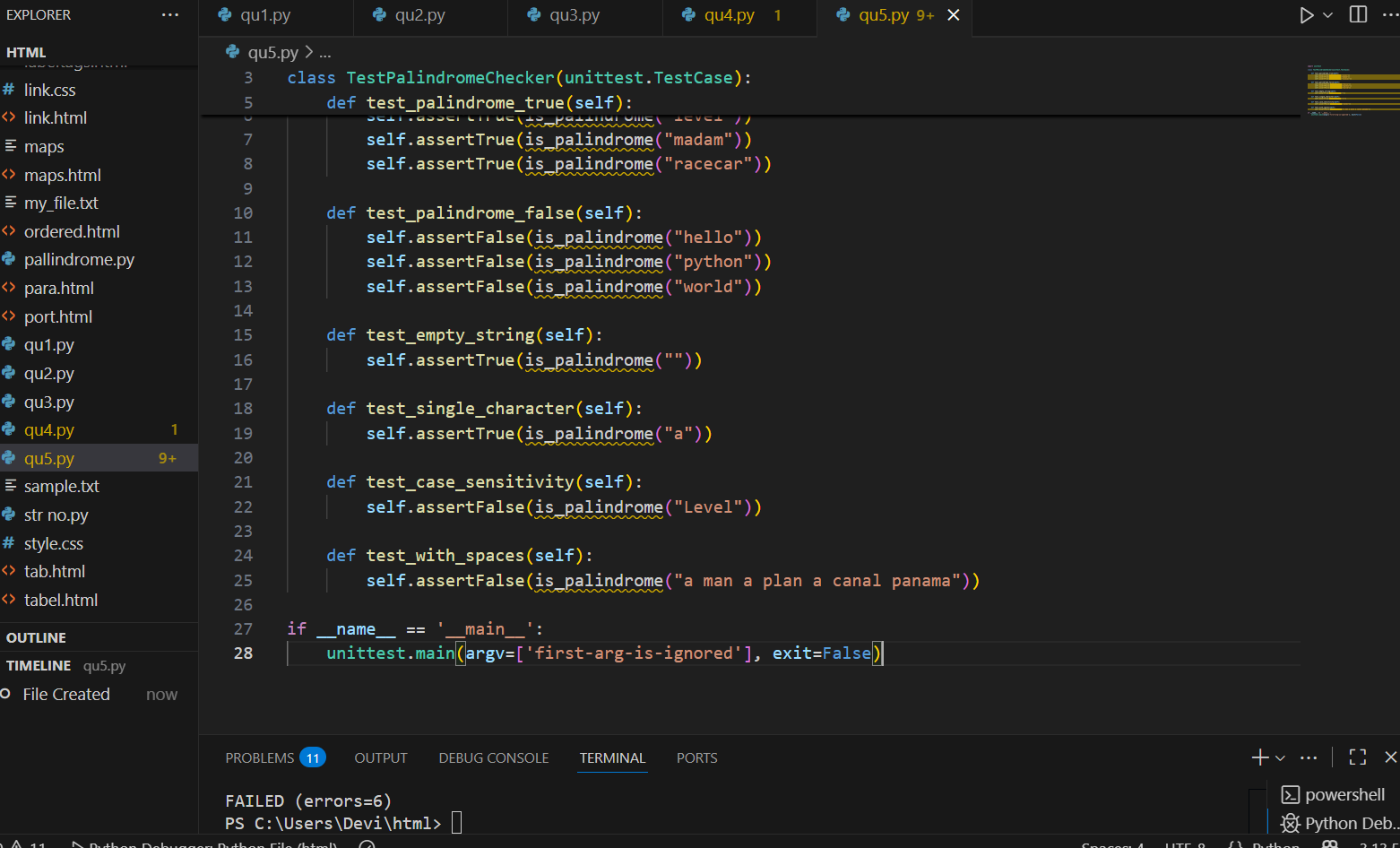


* OBSERVATION: **import unittest**: This line imports the unittest module for creating and running tests.
* **class ShoppingCart:**: This defines the ShoppingCart class.
  + **\_\_init\_\_(self)**: The constructor initializes an empty dictionary self.items to store the items in the cart.
  + **add\_item(self, item\_name, price, quantity=1)**: This method adds an item to the cart. If the item already exists, it increases the quantity; otherwise, it adds the new item with its price and quantity.
  + **remove\_item(self, item\_name)**: This is a placeholder method for removing an item. It currently does nothing (pass).**class TestShoppingCart(unittest.TestCase):**: This defines a test class for the ShoppingCart, inheriting from unittest.TestCase.

TASK-5:

PROMPT: Write tests for a palindrome checker (e.g., is\_palindrome("level") → True).





* OBSERVATION: **import unittest**: This line imports the unittest module, which provides the tools for writing and running tests.
* **def is\_palindrome(text):**: This defines a placeholder function is\_palindrome that is intended to check if a given text is a palindrome. Currently, it just has a pass statement and doesn't contain the actual logic.
* **class TestPalindromeChecker(unittest.TestCase):**: This defines a test class named TestPalindromeChecker that inherits from unittest.TestCase.
* **test\_empty\_string(self)**: This test checks if the function correctly identifies an empty string as a palindrome using self.assertTrue("").
* **test\_single\_character(self)**: This test verifies that a single character string is considered a palindrome using self.assertTrue("a").
* **test\_case\_sensitivity(self)**: This test checks if the function is case-sensitive, expecting False for "Level" which is not a palindrome when considering case.
* **test\_with\_spaces(self)**: This test checks if the function handles strings with spaces, expecting False for "a man a plan a canal panama" as this version with spaces is not a strict palindrome.
* **if \_\_name\_\_ == '\_\_main\_\_':**: This is a standard Python construct that ensures the code inside this block only runs when the script is executed directly.
* **unittest.main(argv=['first-arg-is-ignored'], exit=False)**: This line runs the defined unit tests. The arguments are included to make it compatible with environments like Colab.