LAB ASSIGNMENT-8.4

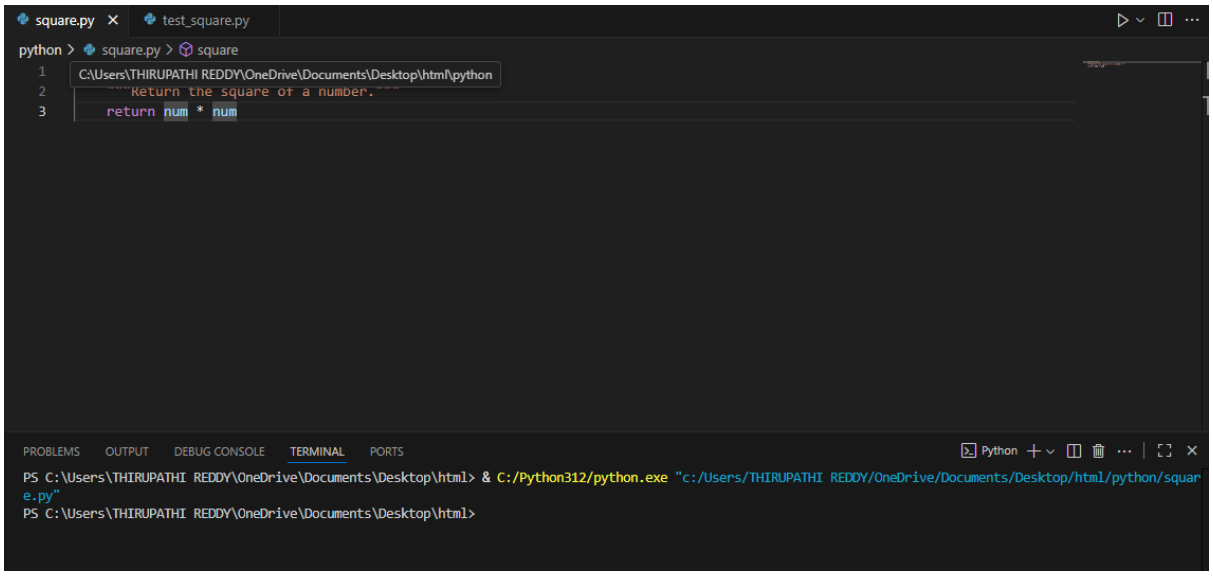
Name: P. Umesh Reddy Hall-Ticket No: 2403a510f9

Batch: 06 Course: AI Assisted Coding

Task 1
Task Description#1
• Write a test case to check if a function returns the square of a number.
• Then write the function with help from GitHub Copilot or Cursor AI.
Expected Outcome#1
• A test file and function file with passing test cases and working logic

#Prompt:
• A test file and function file with passing test cases and working logic

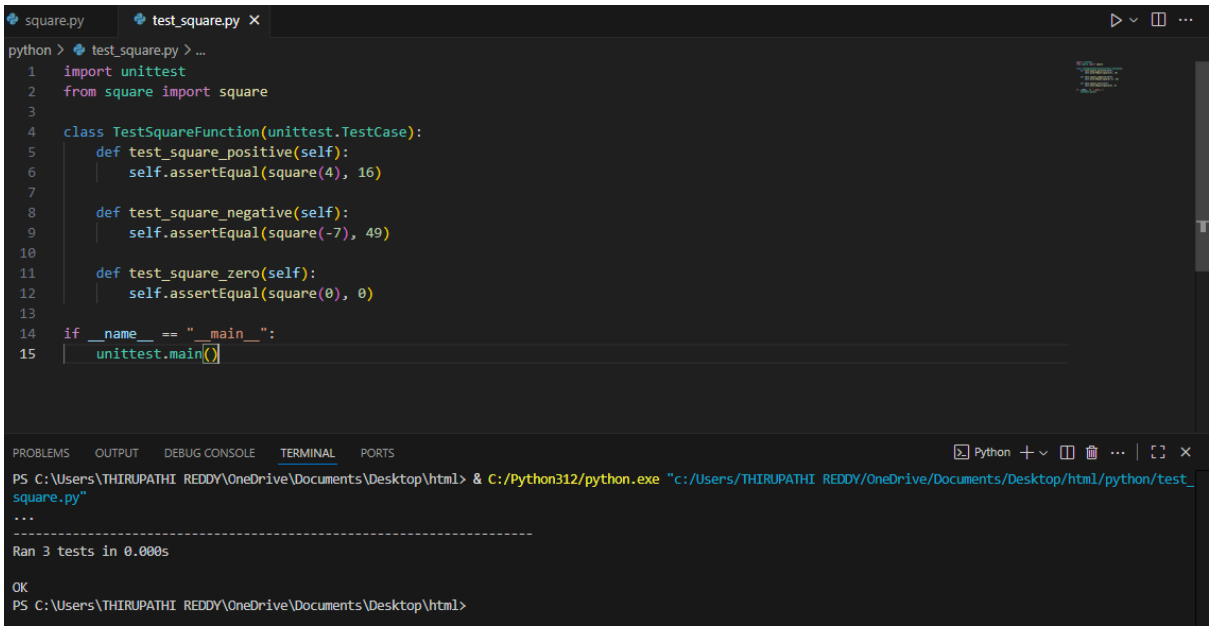
Function File and Output:



Code Explanation:

- The square function takes a number as input and returns its square by multiplying the number by itself.
- The docstring describes the function's purpose.
- This function will pass all the test cases in your test_square.py file.

Test File & Output:



Code Explanation:

- **Imports:**
  import unittest imports Python's built-in unit testing framework.
  from square import square imports the square function from the square.py file.
- **Test Class:**
  class TestSquareFunction(unittest.TestCase): defines a test case class that inherits from unittest.TestCase.
- **Test Methods:**
  - test_square_positive: Checks if square(4) returns 16.
  - test_square_negative: Checks if square(-7) returns 49.
  - test_square_zero: Checks if square(0) returns 0.
- **Assertions:**
  Each test uses self.assertEqual() to compare the function's output to the expected result.
- **Main Block:**
  if __name__ == "__main__": unittest.main() runs all test cases when the script is executed directly.

Comments:

This test file verifies that the square function works correctly for positive numbers, negative numbers, and zero. All tests should pass if the function is implemented properly.

Task 2

Task Description#2

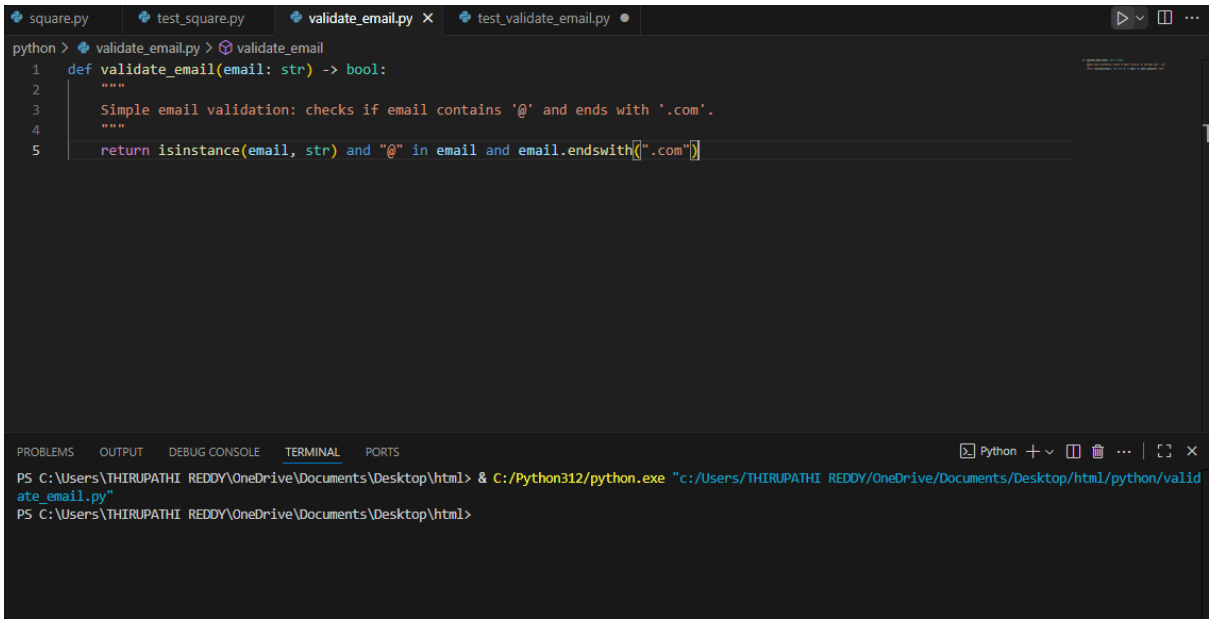
• Create test cases to validate an email address (e.g., contains @ and .com).

• Use AI assistance to implement the validate_email() function.

Expected Outcome#2

• Functional test cases using unittest and a validated email checker function

Code & Output:



Ode Explanation:

- The function validate_email takes an email string as input.
- It returns True only if:
  - The input is a string,
  - It contains the '@' symbol,
  - It ends with '.com'.
- Otherwise, it returns False.
- The docstring describes the function's purpose.

Code:

```python
import unittest
from validate_email import validate_email

class TestValidateEmail(unittest.TestCase):
    def test_valid_email(self):
        self.assertTrue(validate_email("user@example.com"))

    def test_missing_at_symbol(self):
        self.assertFalse(validate_email("userexample.com"))

    def test_missing_dotcom(self):
        self.assertFalse(validate_email("user@example.org"))

    def test_empty_string(self):
        self.assertFalse(validate_email(""))

    def test_non_string_input(self):
        self.assertFalse(validate_email(12345))

    def test_multiple_at_symbols(self):
        self.assertTrue(validate_email("user@domain@company.com"))

if __name__ == "__main__":
    unittest.main()
```

Output:

```
PROBLEMS    OUTPUT    DEBUG CONSOLE    TERMINAL    PORTS

PS C:\Users\THIRUPATHI REDDY\OneDrive\Documents\Desktop\html> & C:/Python312/python.exe "c:/Users/THIRUPATHI REDDY/OneDrive/Documents/Desktop/html/python/test_
validate_email.py"
PS C:\Users\THIRUPATHI REDDY\OneDrive\Documents\Desktop\html> & C:/Python312/python.exe "c:/Users/THIRUPATHI REDDY/OneDrive/Documents/Desktop/html/python/test_
validate_email.py"
validate_email.py"
......
----------------------------------------------------------------------
......
----------------------------------------------------------------------
Ran 6 tests in 0.001s
Ran 6 tests in 0.001s

OK
PS C:\Users\THIRUPATHI REDDY\OneDrive\Documents\Desktop\html>
```

Code Explanation:

- Uses Python's unittest framework to test the validate_email function.
- Each method checks a different scenario:
    - Valid email,
    - Missing '@',
    - Missing '.com',
    - Empty string,
    - Non-string input,
    - Multiple '@' symbols.
- The tests ensure the function works as expected for various inputs.

Comments: