ASSIGNMENT-08

NAME:B.MANISHWAR

HALLTICKET NO:2303A51276

BATCH:05

Task 1: Developing a Utility Function Using TDD

Scenario

You are working on a small utility library for a larger software system. One of the required functions should calculate the square of a given number, and correctness is critical because other modules depend on it.

Task Description

Following the Test Driven Development (TDD) approach:

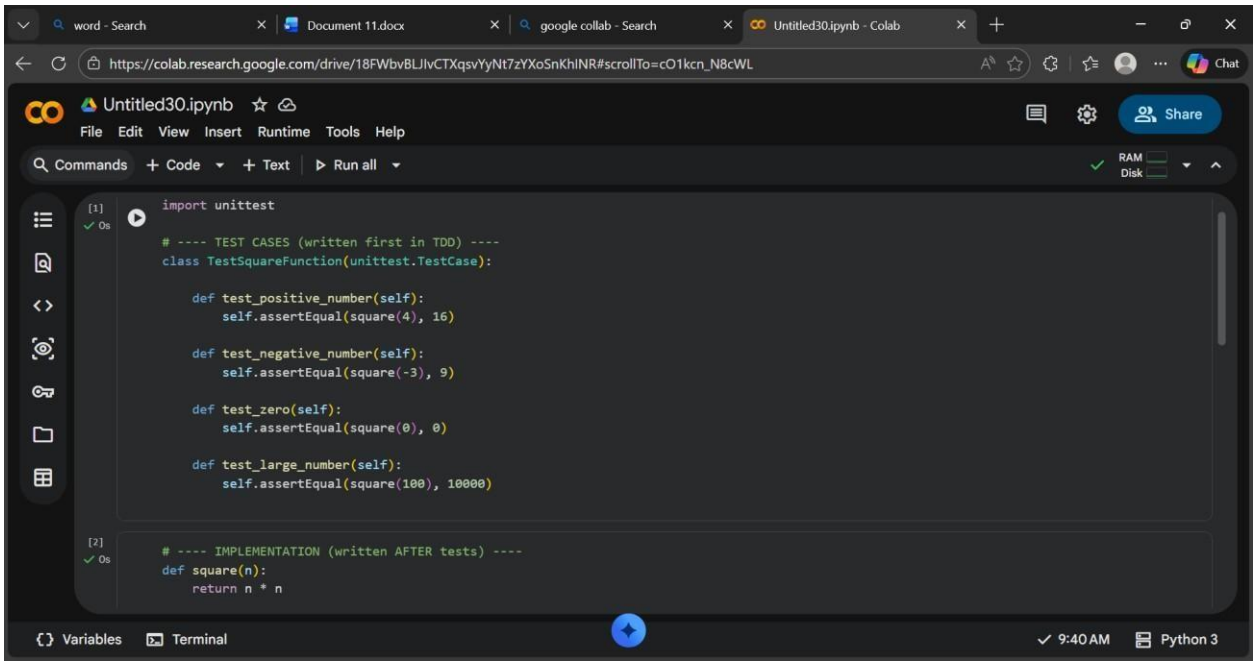
1. **First, write unit test cases to verify that a function correctly returns the square of a number for multiple inputs.**
2. **After defining the test cases, use GitHub Copilot or Cursor AI to generate the function implementation so that all tests pass.**

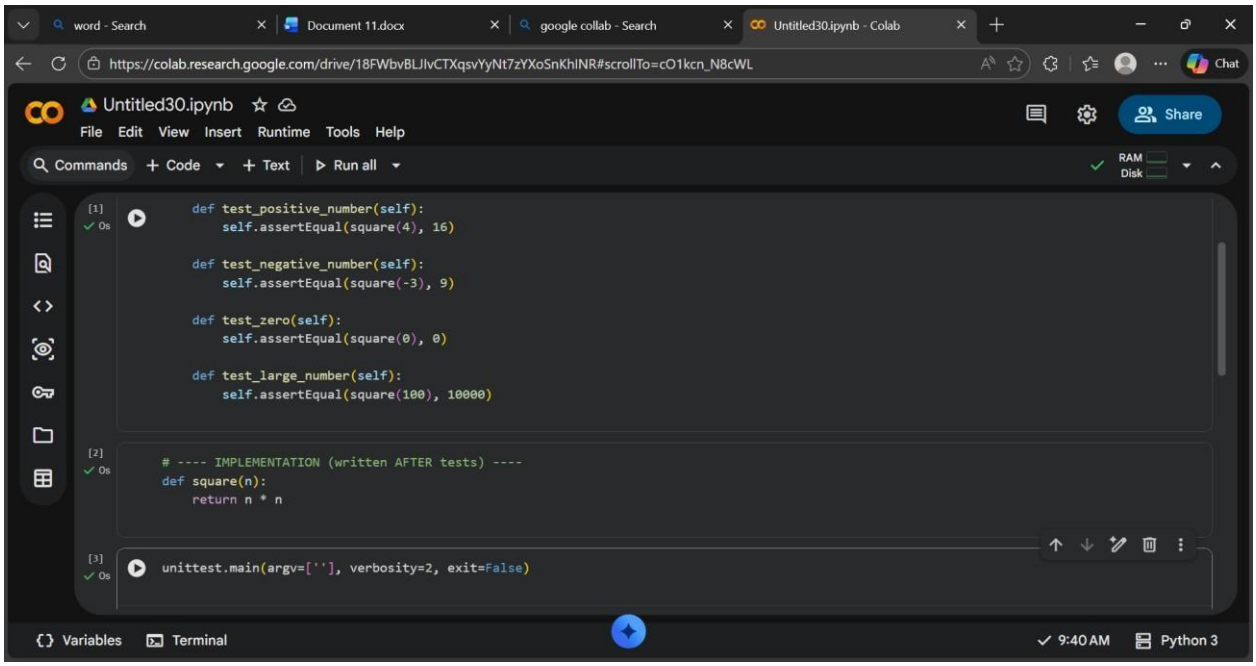
Ensure that the function is written only after the tests are created.

Expected Outcome

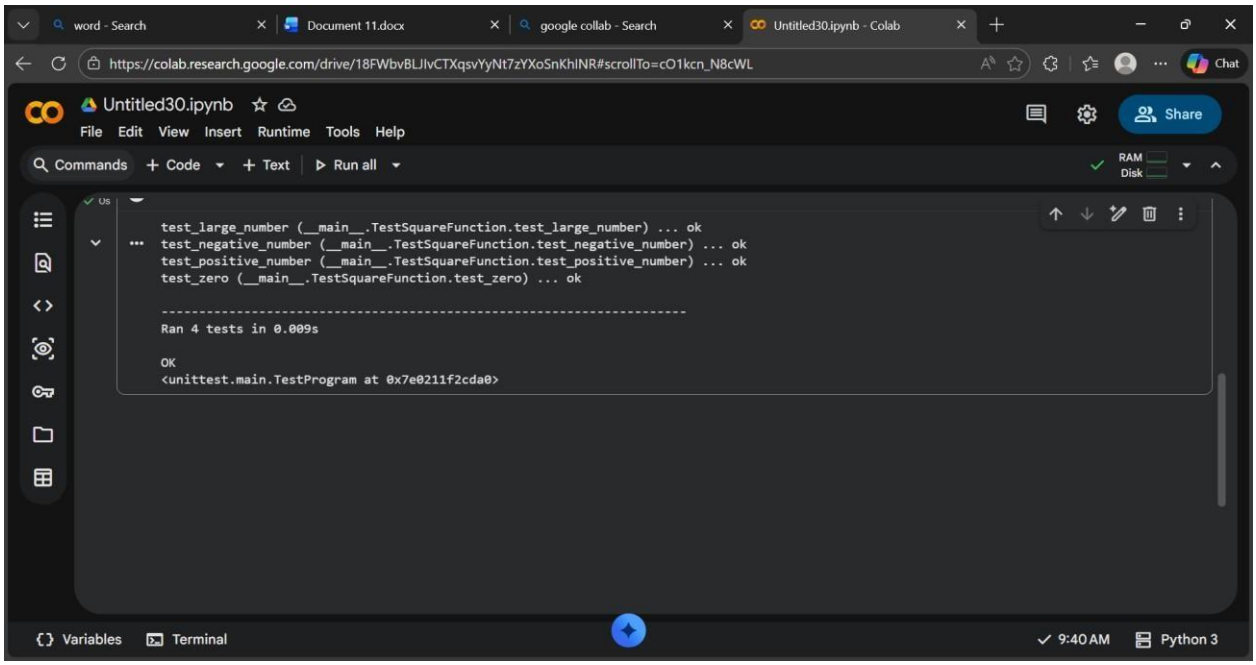
- **A separate test file and implementation file**
- **Clearly written test cases executed before implementation**
- **AI-assisted function implementation that passes all tests**

Demonstration of the TDD cycle: test → fail → implement → pass Code:


word - Search
Document 11.docx
google collab - Search
Untitled30.ipynb - Colab
https://colab.research.google.com/drive/18FWbvBLJlvCTXqsvYyNt7zYXoSnKhlNR#scrollTo=cO1kcn_N8cWL
Chat
Untitled30.ipynb
File Edit View Insert Runtime Tools Help
Share
Commands + Code + Text Run all
RAM Disk
[1] 0s
import unittest
# ---- TEST CASES (written first in TDD) ----
class TestSquareFunction(unittest.TestCase):
def test_positive_number(self):
self.assertEqual(square(4), 16)
def test_negative_number(self):
self.assertEqual(square(-3), 9)
def test_zero(self):
self.assertEqual(square(0), 0)
def test_large_number(self):
self.assertEqual(square(100), 10000)
[2] 0s
# ---- IMPLEMENTATION (written AFTER tests) ----
def square(n):
return n * n
Variables
Terminal
9:40 AM
Python 3


word - Search
Document 11.docx
google collab - Search
Untitled30.ipynb - Colab
https://colab.research.google.com/drive/18FWbvBLJlvCTXqsvYyNt7zYXoSnKhlNR#scrollTo=cO1kcn_N8cWL
Chat
Untitled30.ipynb
File Edit View Insert Runtime Tools Help
Share
Commands + Code + Text Run all
RAM Disk
[1] 0s
def test_positive_number(self):
self.assertEqual(square(4), 16)
def test_negative_number(self):
self.assertEqual(square(-3), 9)
def test_zero(self):
self.assertEqual(square(0), 0)
def test_large_number(self):
self.assertEqual(square(100), 10000)
[2] 0s
# ---- IMPLEMENTATION (written AFTER tests) ----
def square(n):
return n * n
[3] 0s
unittest.main(argv=[''], verbosity=2, exit=False)
Variables
Terminal
9:40 AM
Python 3

Output:



Task 2: Email Validation for a User Registration System

Scenario

You are developing the backend of a user registration system. One requirement is to validate user email addresses before storing them in the database.

Task Description

Apply Test Driven Development by:

1. **Writing unit test cases that define valid and invalid email formats (e.g., missing @, missing domain, incorrect structure).**
2. **Using AI assistance to implement the validate_email() function based strictly on the behavior described by the test cases.**

The implementation should be driven entirely by the test expectations.

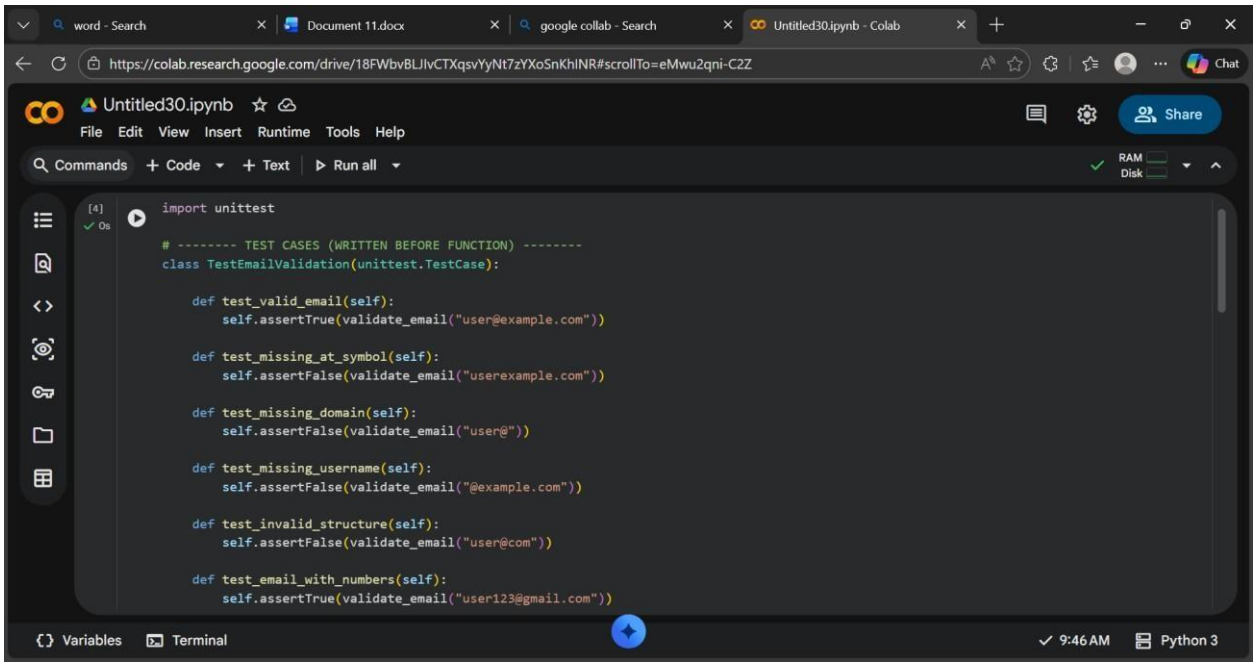
Expected Outcome

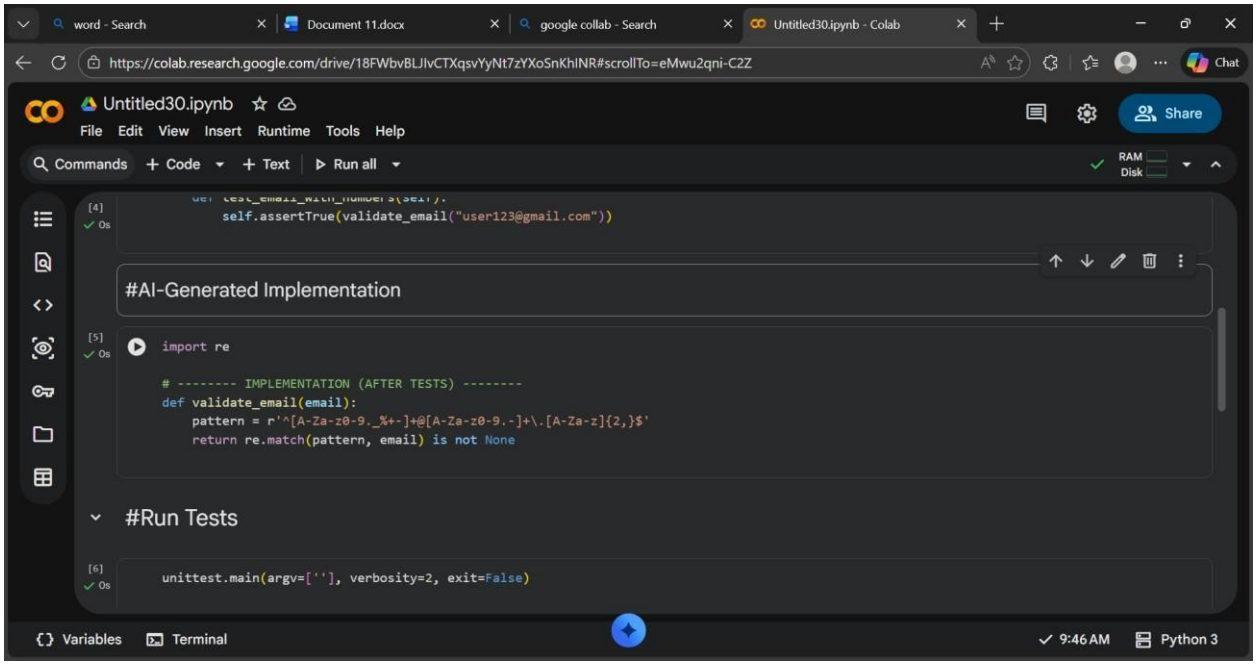
- **Well-defined unit tests using unittest or pytest**
- **An AI-generated email validation function**

- **All test cases passing successfully**

Clear alignment between test cases and function behavior

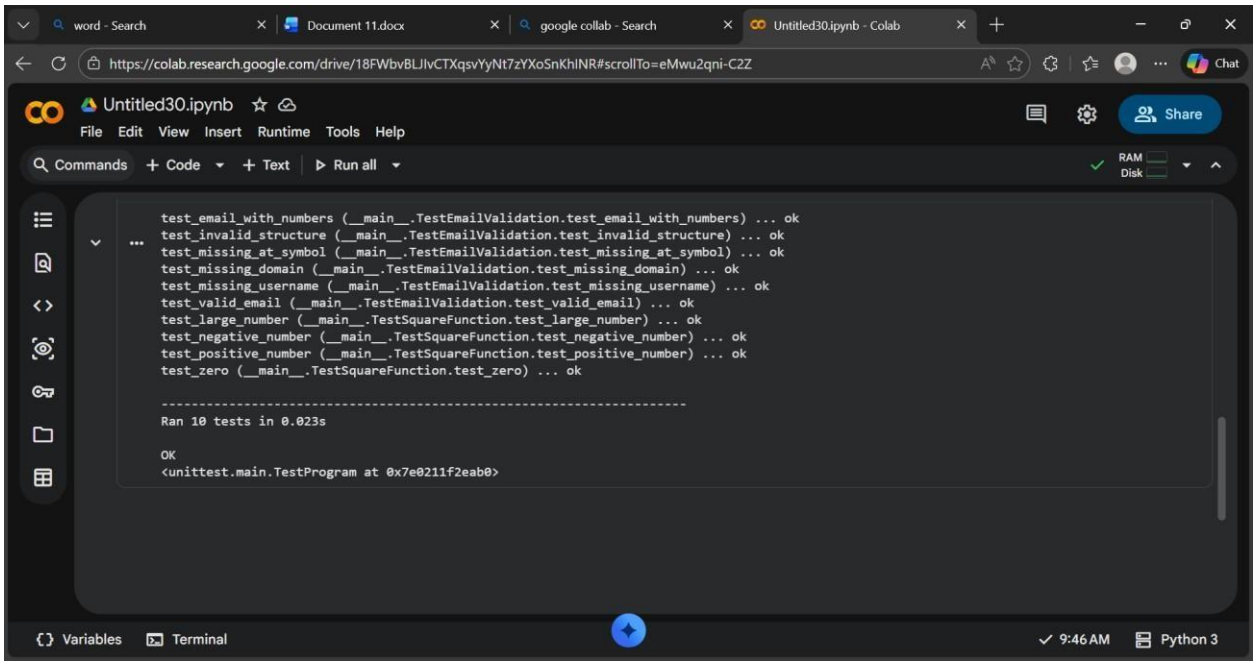
Code:





Output:

•



Task 3: Decision Logic Development Using TDD

Scenario

In a grading or evaluation module, a function is required to determine the maximum value among three inputs. Accuracy is essential, as incorrect results could affect downstream decision logic.

Task Description

Using the TDD methodology:

1. **Write test cases that describe the expected output for different combinations of three numbers.**
2. **Prompt GitHub Copilot or Cursor AI to implement the function logic based on the written tests.**

Avoid writing any logic before test cases are completed.

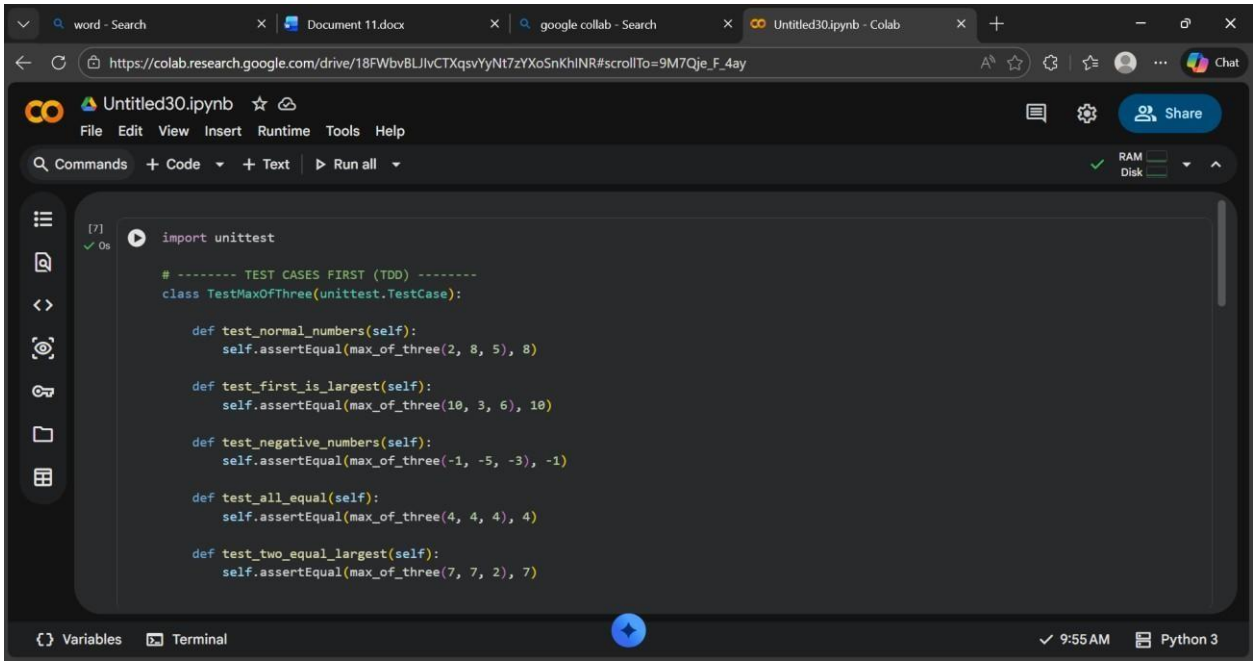
Expected Outcome

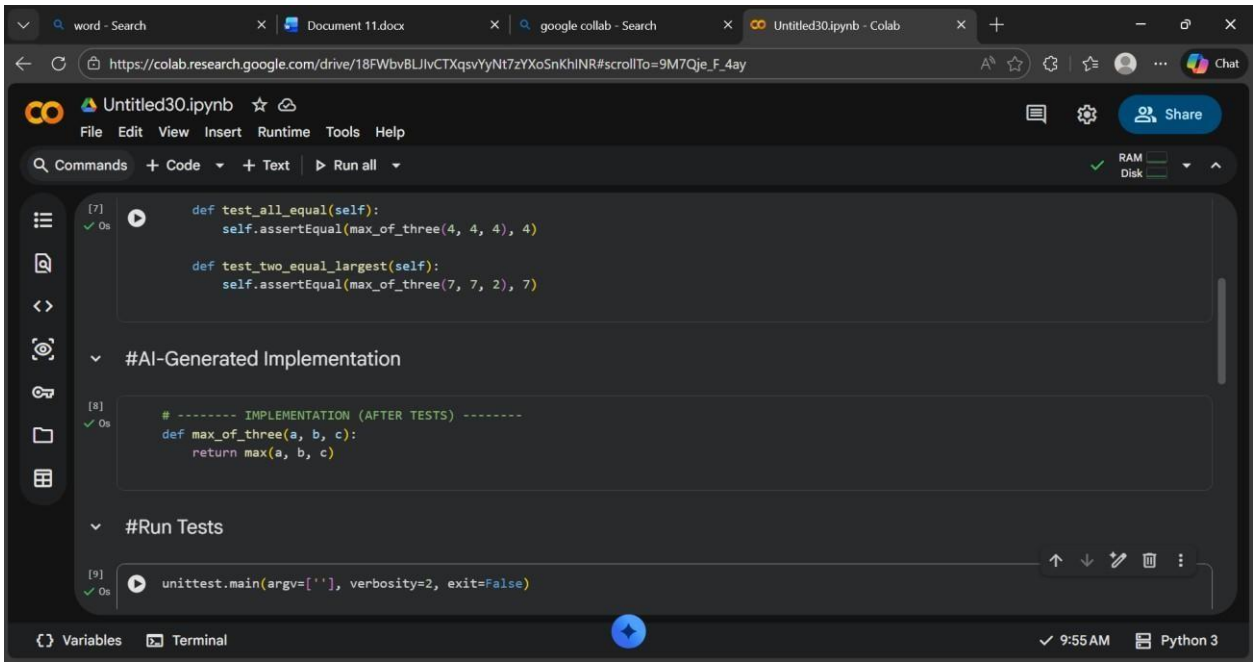
- **Comprehensive test cases covering normal and edge cases**
- **AI-generated function implementation**

- **Passing test results demonstrating correctness**

  Evidence that logic was derived from tests, not assumptions

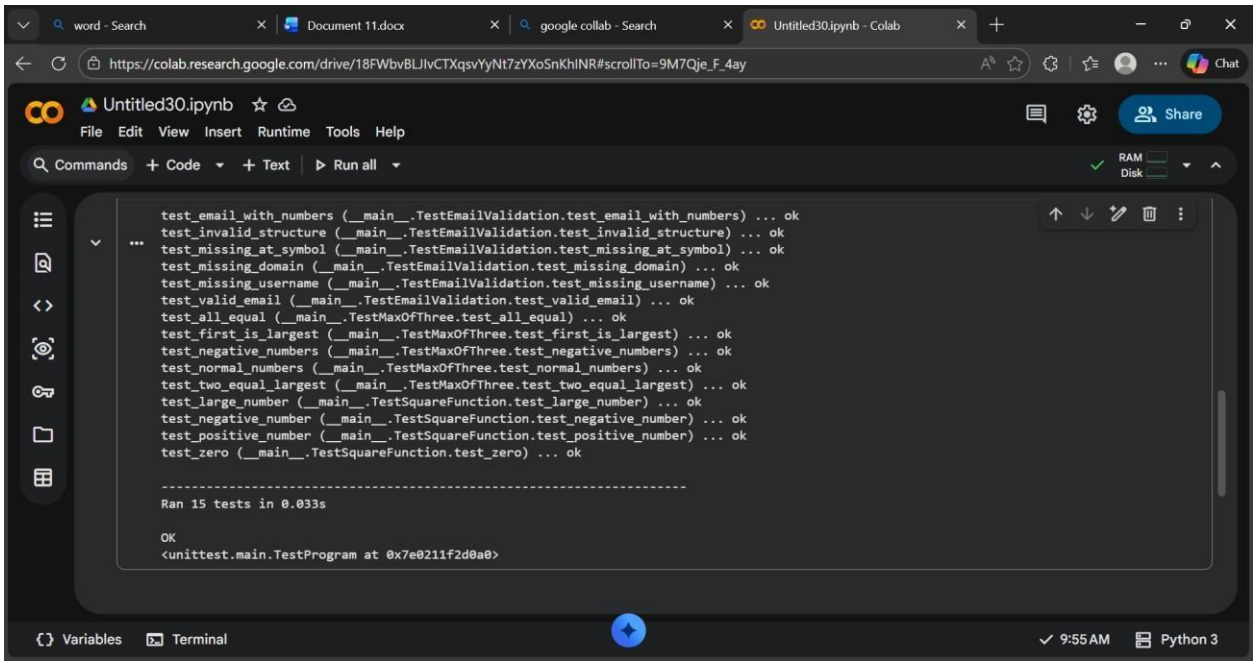
Code:





Output:

•



Task 4: Shopping Cart Development with AI-Assisted TDD

Scenario

You are building a simple shopping cart module for an e-commerce application.

The cart must support adding items, removing items, and calculating the total price accurately.

Task Description

Follow a test-driven approach:

1. Write unit tests for each required behavior:

o Adding an item o Removing an item o Calculating the total price

2. After defining all tests, use AI tools to generate the ShoppingCart class and its methods so that the tests pass.

Focus on behavior-driven testing rather than implementation details.

Expected Outcome

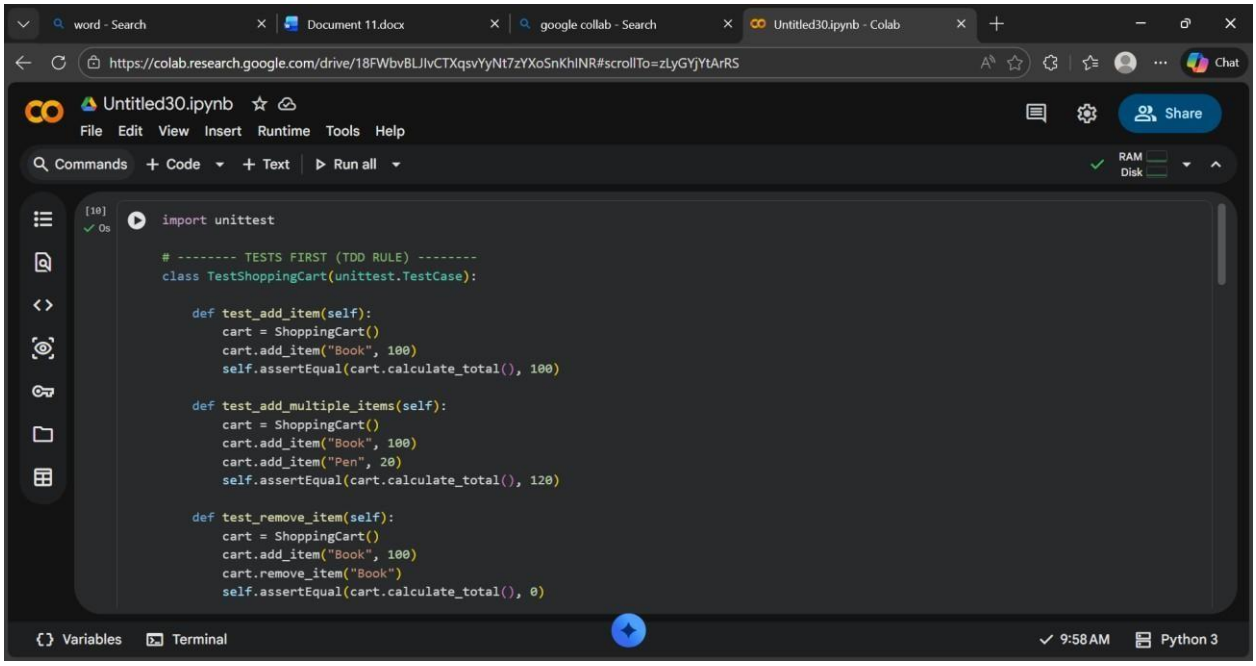
- **Unit tests defining expected shopping cart behavior**

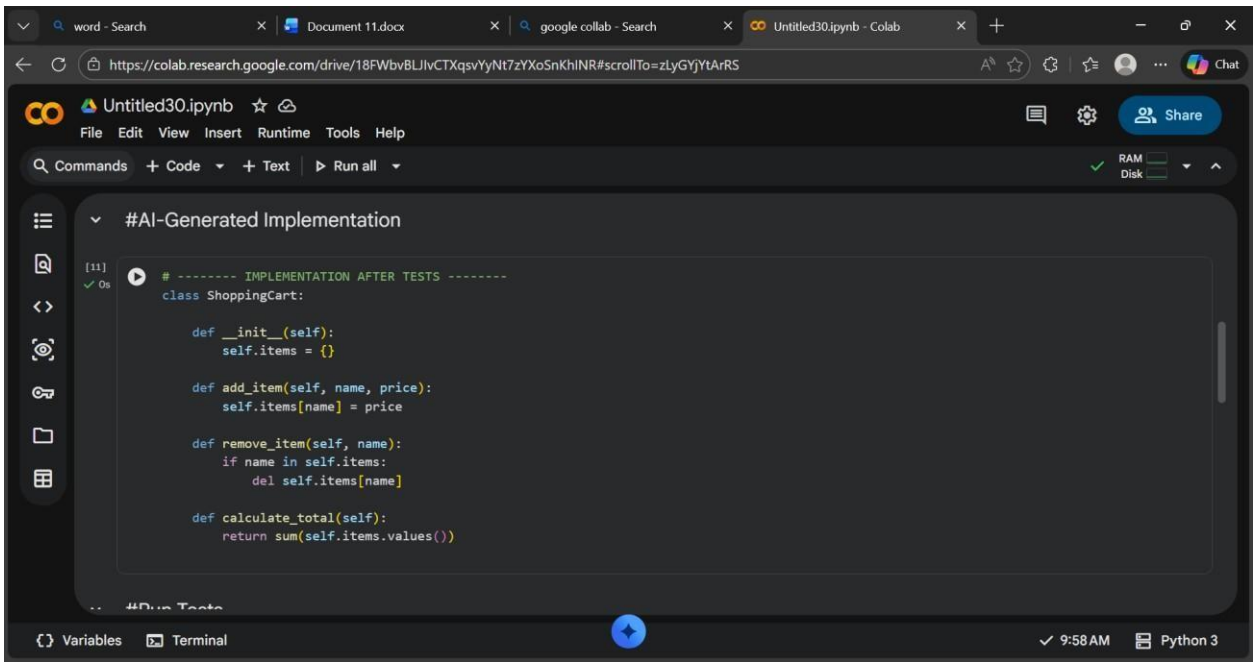
  AI-generated class implementation

- **All tests passing successfully**

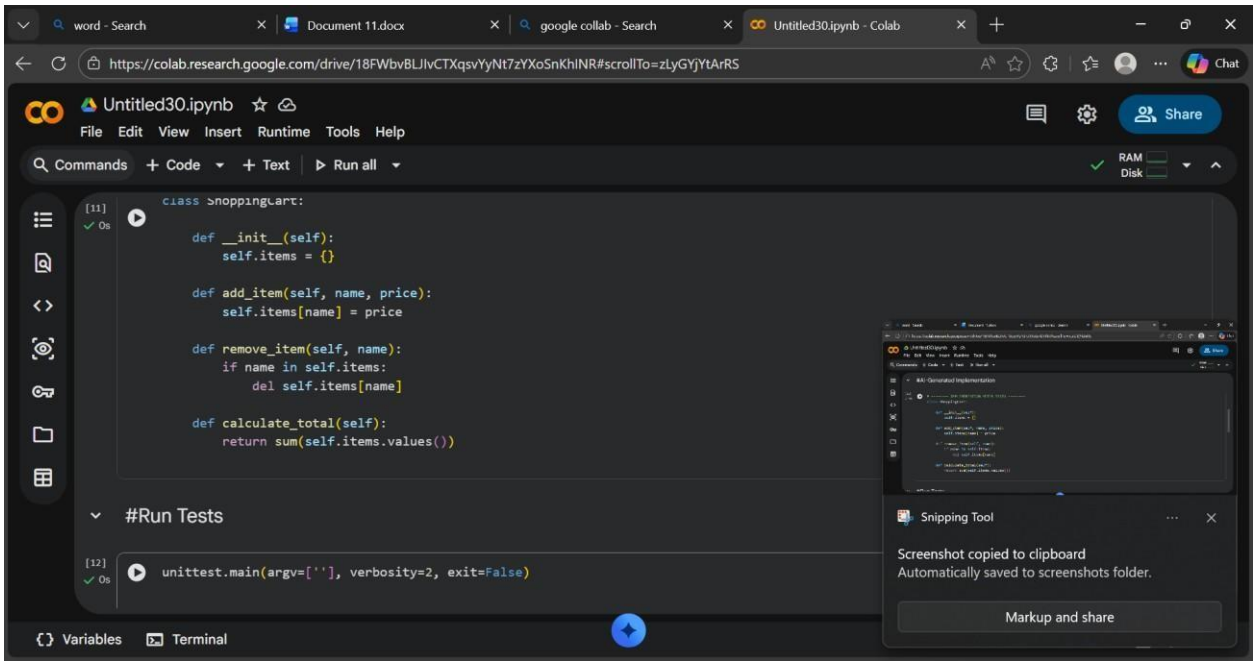
- **Clear demonstration of TDD applied to a class-based design**

CODE:

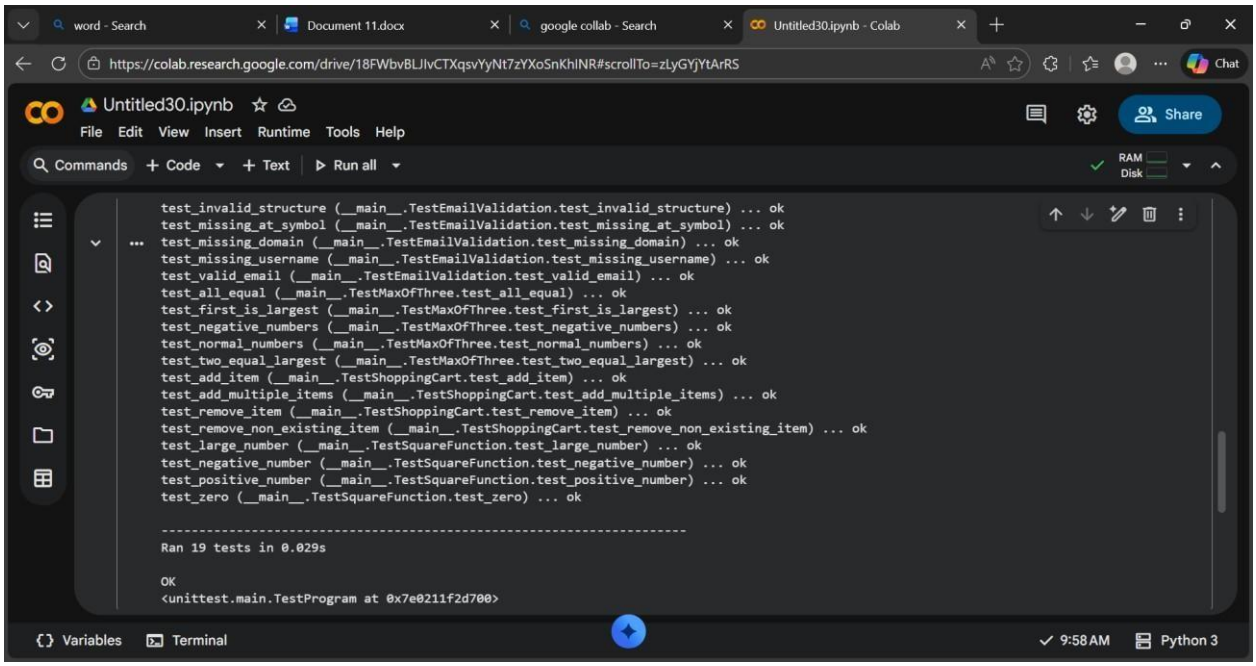




•



Output:



Task 5: String Validation Module Using TDD

Scenario

You are working on a text-processing module where a function is required to identify whether a given string is a palindrome. The function must handle different cases and inputs reliably.

Task Description

Using Test Driven Development:

1. Write test cases for a palindrome checker covering:

o Simple palindromes

o Non-palindromes o

Case variations

2. Use GitHub Copilot or Cursor AI to generate the is_palindrome() function based on the test case expectations.

The function should be implemented only after tests are written.

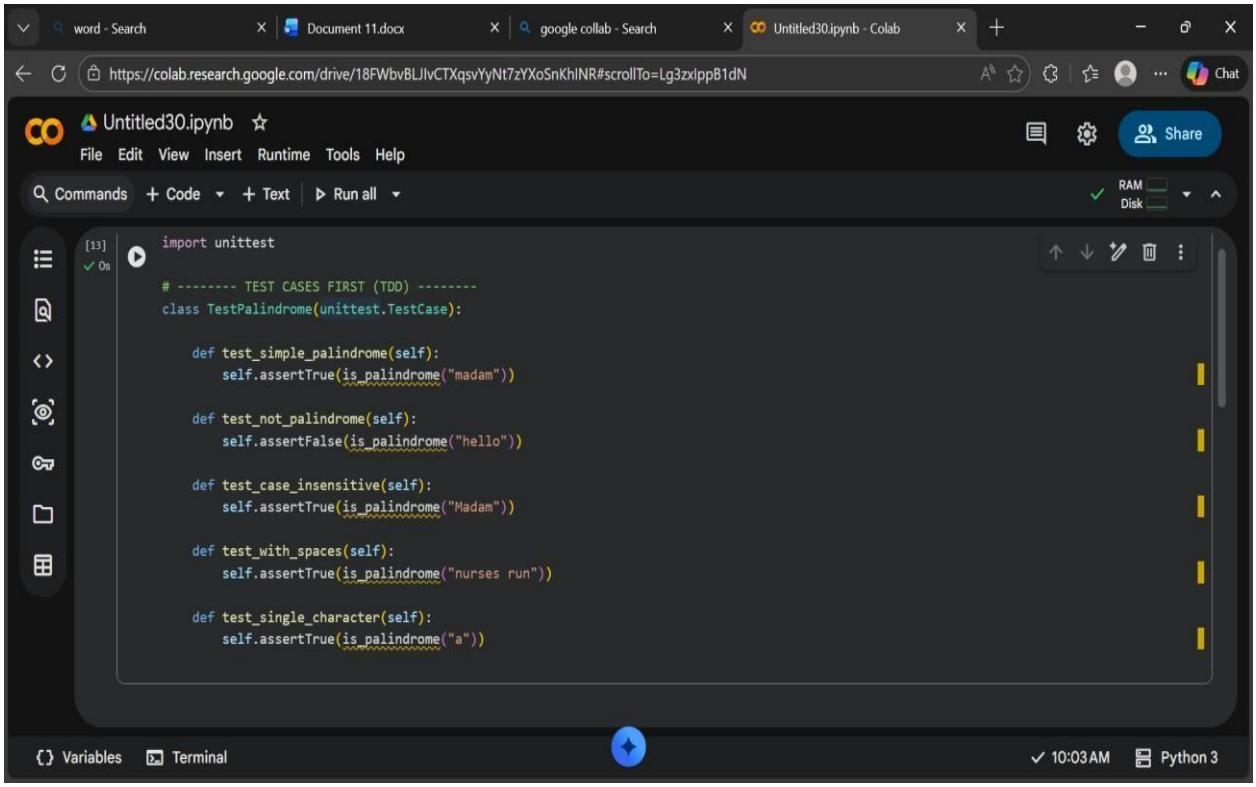
Expected Outcome

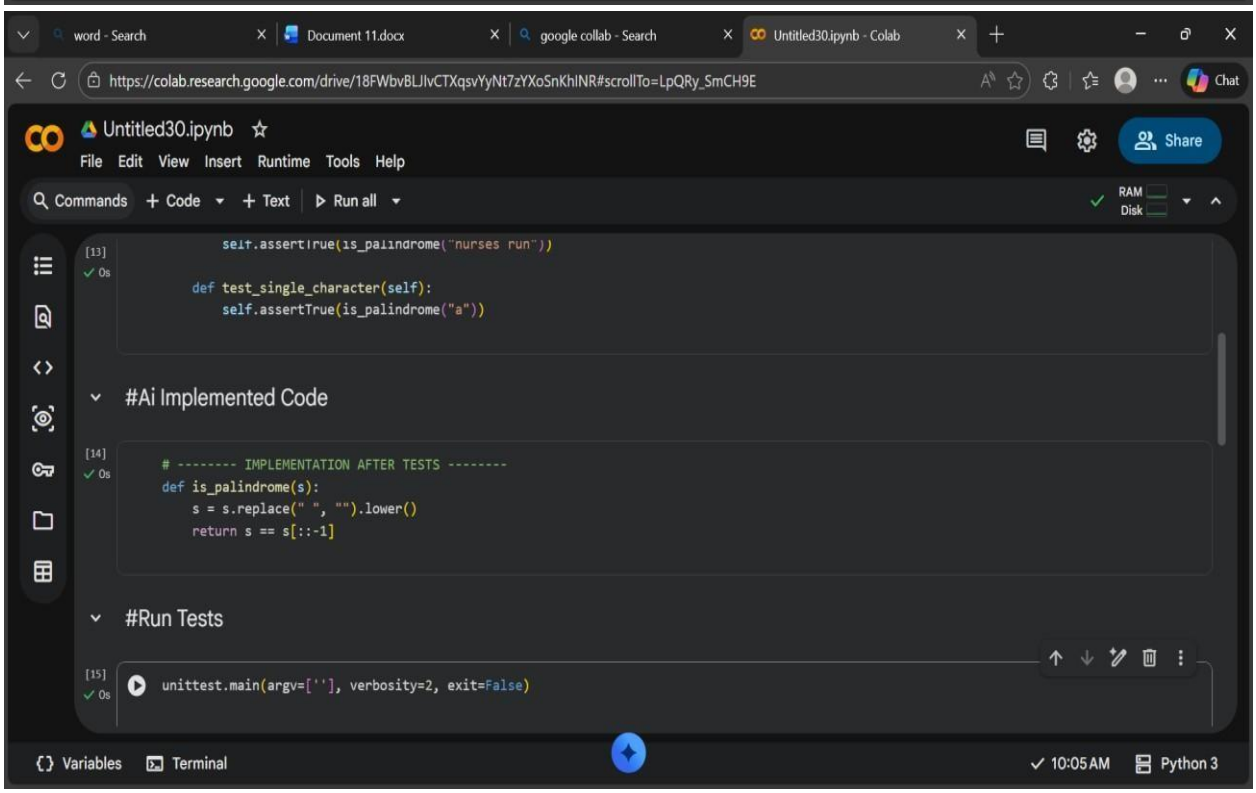
• Clearly written test cases defining expected behavior

• AI-assisted implementation of the palindrome checker

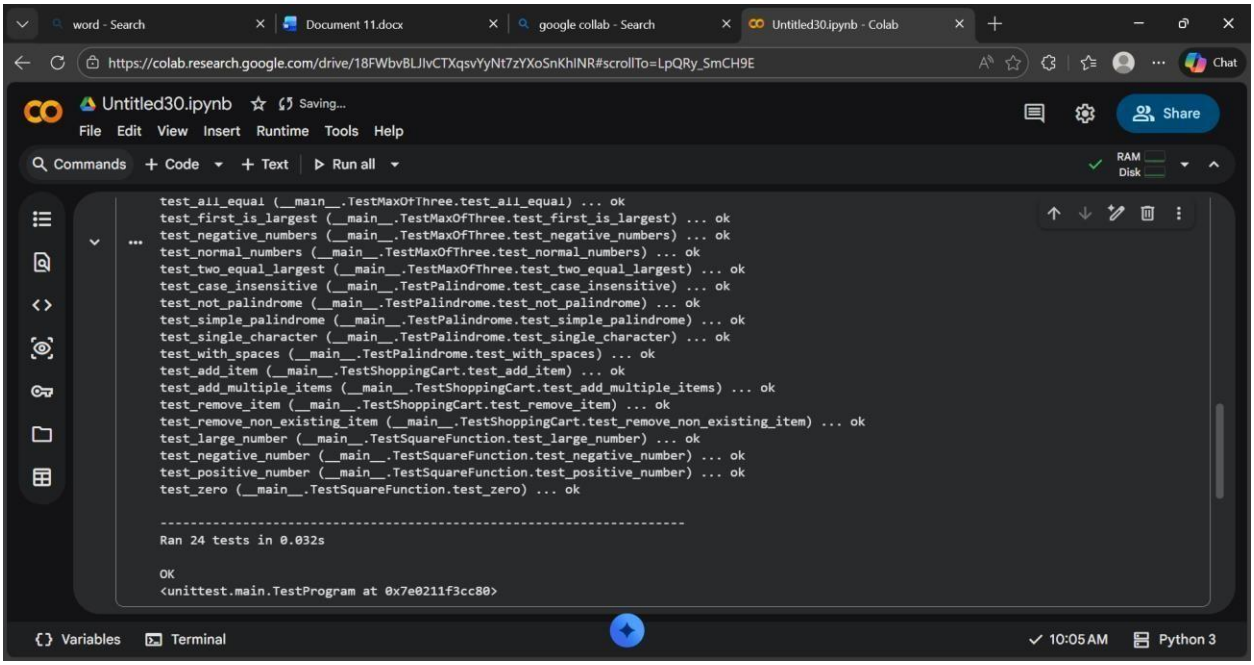
All test cases passing successfully • Evidence of TDD methodology applied correctly

CODE:


word - Search
Document 11.docx
google collab - Search
Untitled30.ipynb - Colab
https://colab.research.google.com/drive/18FWbvBLJIvCTXqsvYyNt7zYXoSnKhINR#scrollTo=Lg3zxIppB1dN
Chat
Untitled30.ipynb
File Edit View Insert Runtime Tools Help
Share
Commands + Code + Text Run all
RAM
Disk
[13]
0s
import unittest
# -------- TEST CASES FIRST (TDD) --------
class TestPalindrome(unittest.TestCase):
def test_simple_palindrome(self):
self.assertTrue(is_palindrome("madam"))
def test_not_palindrome(self):
self.assertFalse(is_palindrome("hello"))
def test_case_insensitive(self):
self.assertTrue(is_palindrome("Madam"))
def test_with_spaces(self):
self.assertTrue(is_palindrome("nurses run"))
def test_single_character(self):
self.assertTrue(is_palindrome("a"))
Variables
Terminal
10:03 AM
Python 3


word - Search
Document 11.docx
google collab - Search
Untitled30.ipynb - Colab
https://colab.research.google.com/drive/18FWbvBLJIvCTXqsvYyNt7zYXoSnKhINR#scrollTo=LpQRy_SmCH9E
Chat
Untitled30.ipynb
File Edit View Insert Runtime Tools Help
Share
Commands + Code + Text Run all
RAM
Disk
[13]
0s
self.assertTrue(is_palindrome("nurses run"))
def test_single_character(self):
self.assertTrue(is_palindrome("a"))
#Ai Implemented Code
[14]
0s
# -------- IMPLEMENTATION AFTER TESTS --------
def is_palindrome(s):
s = s.replace(" ", "").lower()
return s == s[::-1]
#Run Tests
[15]
0s
unittest.main(argv=[''], verbosity=2, exit=False)
Variables
Terminal
10:05 AM
Python 3

Output:


word - Search
Document 11.docx
google collab - Search
Untitled30.ipynb - Colab
https://colab.research.google.com/drive/18FWbvBLJlvCTXqsvYyNt7zYXoSnKhlNR#scrollTo=LpQRy_SmCH9E
Chat
Untitled30.ipynb
Saving...
File Edit View Insert Runtime Tools Help
Share
Commands + Code + Text Run all
RAM
Disk
test_all_equal (__main__.TestMaxOfThree.test_all_equal) ... ok
test_first_is_largest (__main__.TestMaxOfThree.test_first_is_largest) ... ok
test_negative_numbers (__main__.TestMaxOfThree.test_negative_numbers) ... ok
test_normal_numbers (__main__.TestMaxOfThree.test_normal_numbers) ... ok
test_two_equal_largest (__main__.TestMaxOfThree.test_two_equal_largest) ... ok
test_case_insensitive (__main__.TestPalindrome.test_case_insensitive) ... ok
test_not_palindrome (__main__.TestPalindrome.test_not_palindrome) ... ok
test_simple_palindrome (__main__.TestPalindrome.test_simple_palindrome) ... ok
test_single_character (__main__.TestPalindrome.test_single_character) ... ok
test_with_spaces (__main__.TestPalindrome.test_with_spaces) ... ok
test_add_item (__main__.TestShoppingCart.test_add_item) ... ok
test_add_multiple_items (__main__.TestShoppingCart.test_add_multiple_items) ... ok
test_remove_item (__main__.TestShoppingCart.test_remove_item) ... ok
test_remove_non_existing_item (__main__.TestShoppingCart.test_remove_non_existing_item) ... ok
test_large_number (__main__.TestSquareFunction.test_large_number) ... ok
test_negative_number (__main__.TestSquareFunction.test_negative_number) ... ok
test_positive_number (__main__.TestSquareFunction.test_positive_number) ... ok
test_zero (__main__.TestSquareFunction.test_zero) ... ok
Ran 24 tests in 0.032s
OK
<unittest.main.TestProgram at 0x7e0211f3cc80>
Variables
Terminal
10:05 AM
Python 3