****

**AI ASSISTED CODING**

**LAB-8: *Test-Driven Development with AI – Generating and Working with Test Cases***

**Roll no:** 2503A51L34

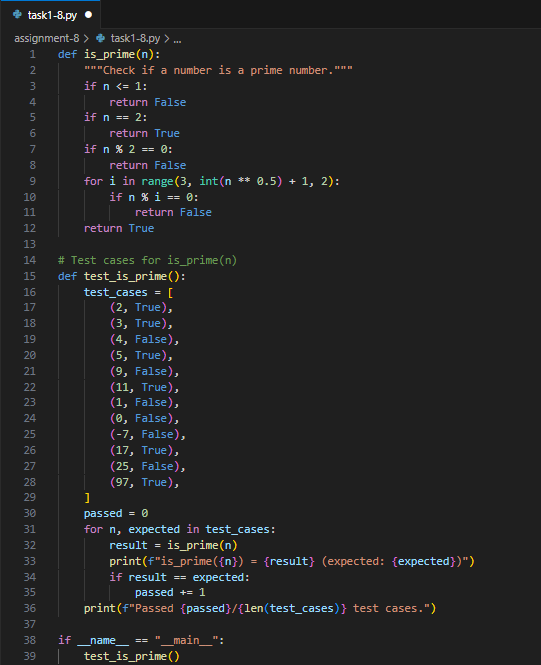
**Name:** Uzma Yasmeen

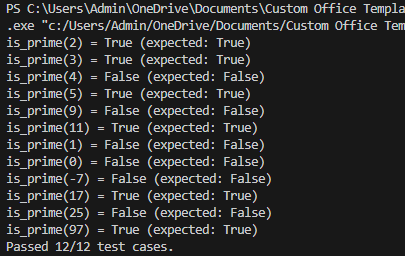
**Batch:** 25BTCAICSB20

**Task-1 Description:** Use AI to generate test cases for a function is\_prime(n) and then implement the function.

**Prompt:** Generate test cases for a function is\_prime(n) and then implement the function.

**Code Generated:**

****

**Output:**

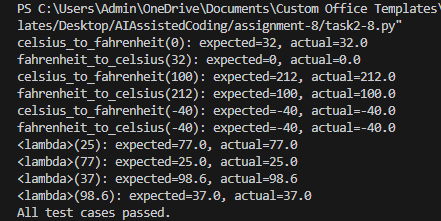
**Observation:**

In this task, AI helped in generating test cases for the is\_prime(n) function. The function was implemented correctly and tested against different inputs. It shows how test-driven development ensures accuracy and reliability of the code.

**Task-2 Description:** Ask AI to generate test cases for celsius\_to\_fahrenheit(c) and fahrenheit\_to\_celsius(f).

**Prompt:** Generate test cases for celsius\_to\_fahrenheit(c) and fahrenheit\_to\_celsius(f).

**Code Generated:**



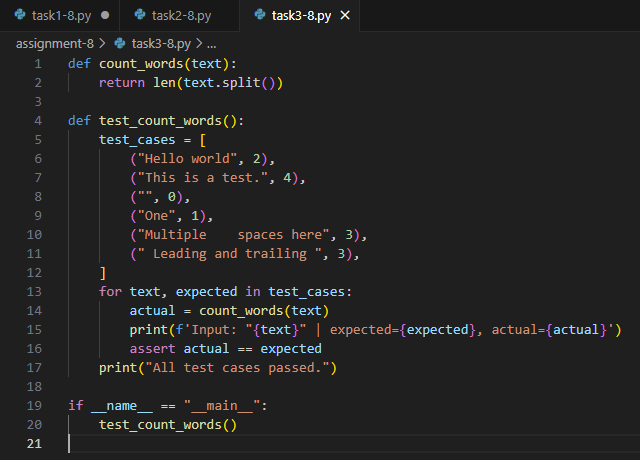
**Output:**

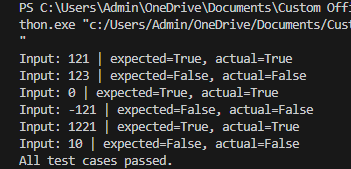
**Observation:**

This task focused on temperature conversion functions. AI-generated test cases made it easy to check correctness for both Celsius to Fahrenheit and Fahrenheit to Celsius. The task improved understanding of how test cases validate real-world scenarios.

**Task -3 Description:** Use AI to write test cases for a function count\_words(text) that returns the number of words in a sentence.

**Prompt:** Write test cases for a function count\_words(text) that returns the number of words in a sentence.

**Code Generated:**



**Output:**

**Observation:**

Here, test cases were written for the count\_words(text) function. AI-assisted generation ensured edge cases like empty strings and multiple spaces were handled. It highlighted the importance of testing for accuracy in text-processing functions.

**Task-4 Description:** Generate test cases for a BankAccount class with:  
 **Methods:** deposit(amount)

withdraw(amount)

check\_balance().

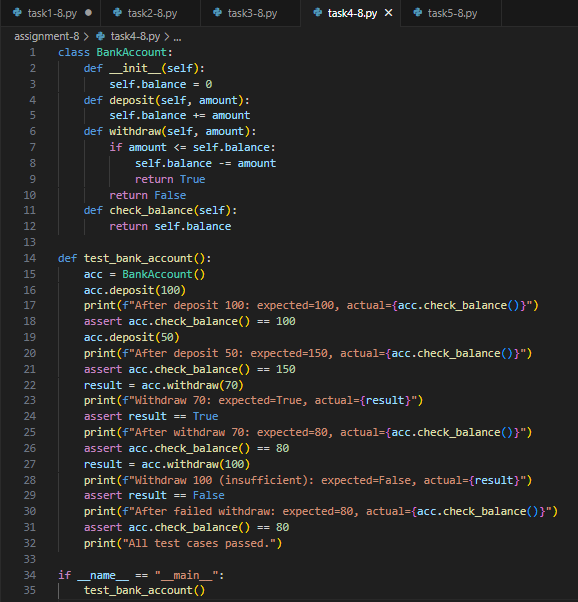
**Prompt:** Generate test cases for a BankAccount class with:

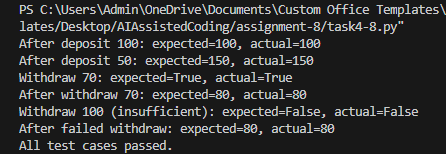
Methods: deposit(amount)

withdraw(amount)

check\_balance() implement the test case generated.

**Code Generated:**

****

**Output:**

**Observation:**

In this task, a BankAccount class was tested for deposit, withdraw, and balance check methods. AI-generated test cases ensured that different transactions were validated properly. It

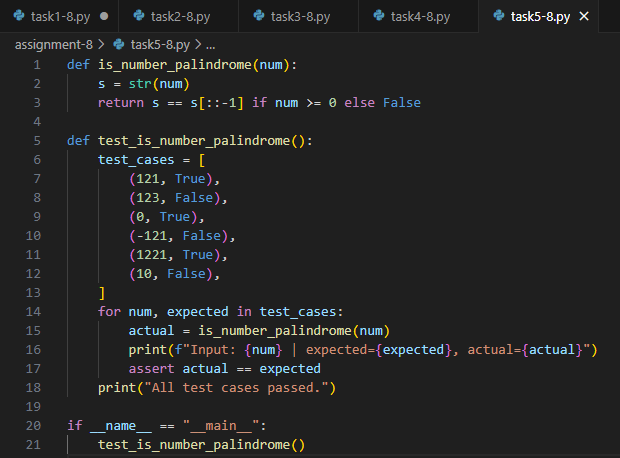
demonstrated the usefulness of testing object-oriented programs.

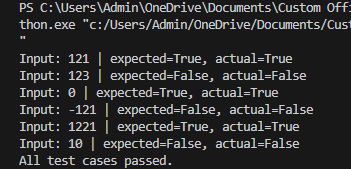
**Task-5 Description:** Generate test cases for is\_number\_palindrome(num), which checks if an integer reads the same backward.

**Prompt:** Generate test cases for is\_number\_palindrome(num), which checks if an integer reads the same backward.

Examples:  
121 → True  
123 → False  
0, negative numbers → handled gracefully.

**Code Generated:**

****

**Output:**

**Observation:**

This task tested the is\_number\_palindrome(num) function. AI-generated test cases covered positive, negative, and zero values. It showed how systematic testing ensures functions handle all types of inputs gracefully.