

**SR UNIVERSITY**

**Assignment-8.2**

Name : Eldi Arun Kumar

Enrollment No : 2503A51L27

Batch No : 19

**TASK DESCRIPTION 1:**

Use AI to generate test cases for a function is\_prime(n) and then implement the function.

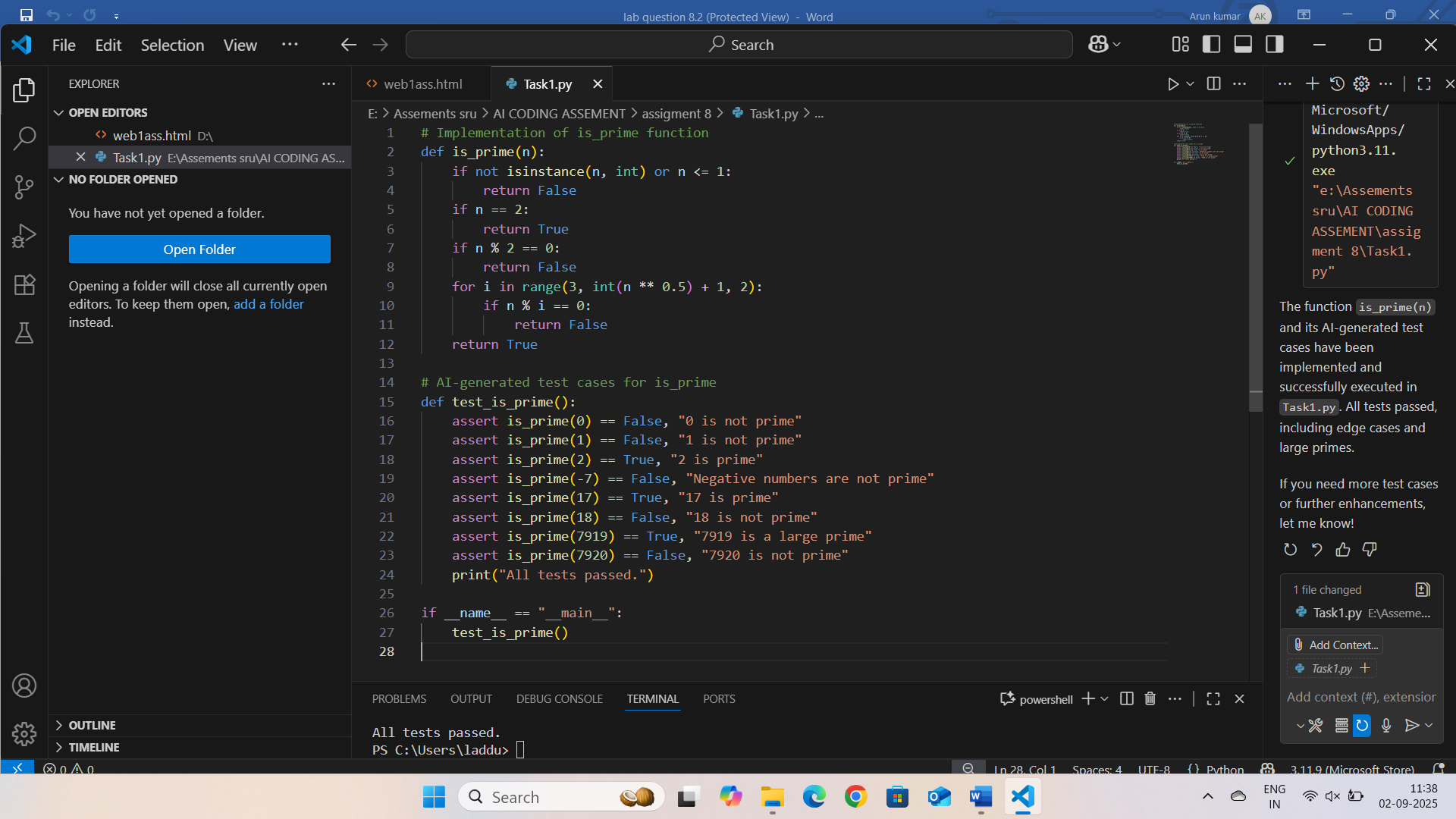
Requirements:

\*Only integers > 1 can be prime.

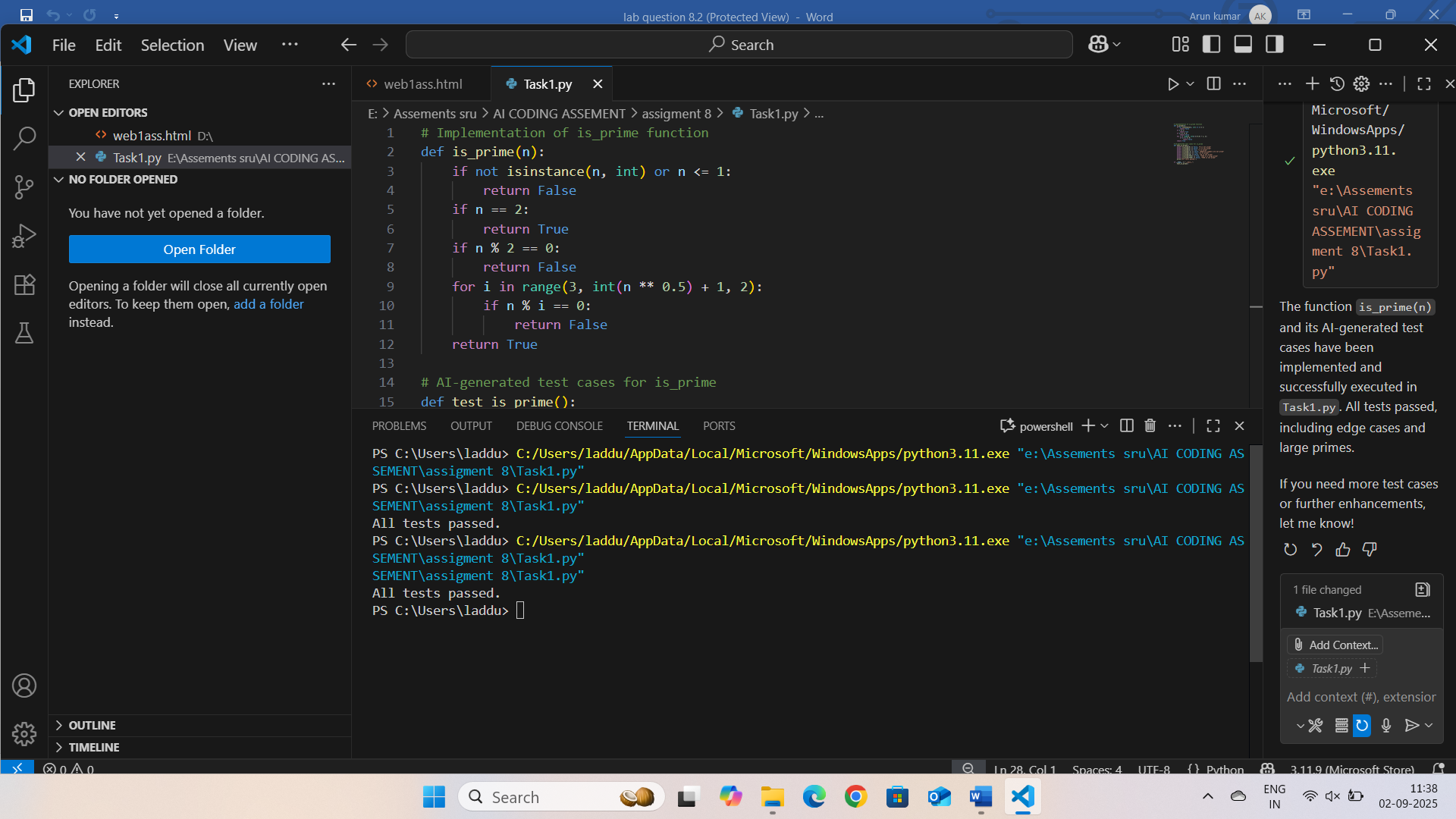
\*Check edge cases: 0, 1, 2, negative numbers, and large primes.

**PROMPT:** Generate at least 10 test cases (including edge cases like 0, 1, negative numbers, 2, and a large prime) for a Python function is\_prime(n) that checks if a number is prime. Then implement the function efficiently and write a test runner to compare expected vs actual results.

**CODE :**

****

**OUTPUT:**

****

**OBSERVATION:** The is\_prime(n) function must correctly identify whether an integer is prime, considering that only numbers greater than 1 can be prime. Edge cases such as 0, 1, negative numbers, and very large primes need to be tested to ensure accuracy and efficiency. The function should avoid brute force for large inputs and produce correct results for all provided test cases.

**TASK DESCRIPTION 2:**

Ask AI to generate test cases for celsius\_to\_fahrenheit(c) and fahrenheit\_to\_celsius(f).

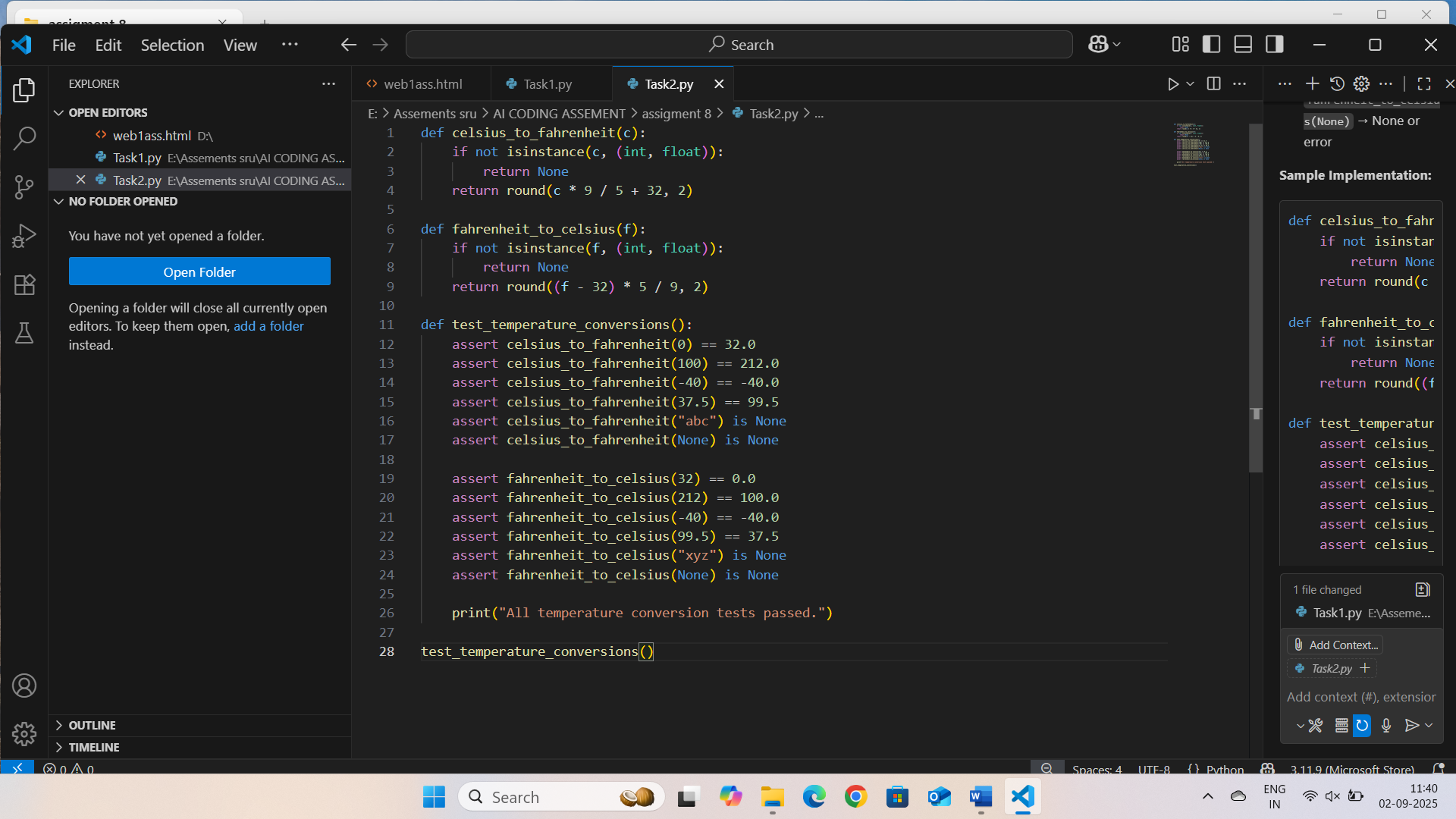
Requirements:

\*Validate known pairs: 0°C = 32°F, 100°C = 212°F.

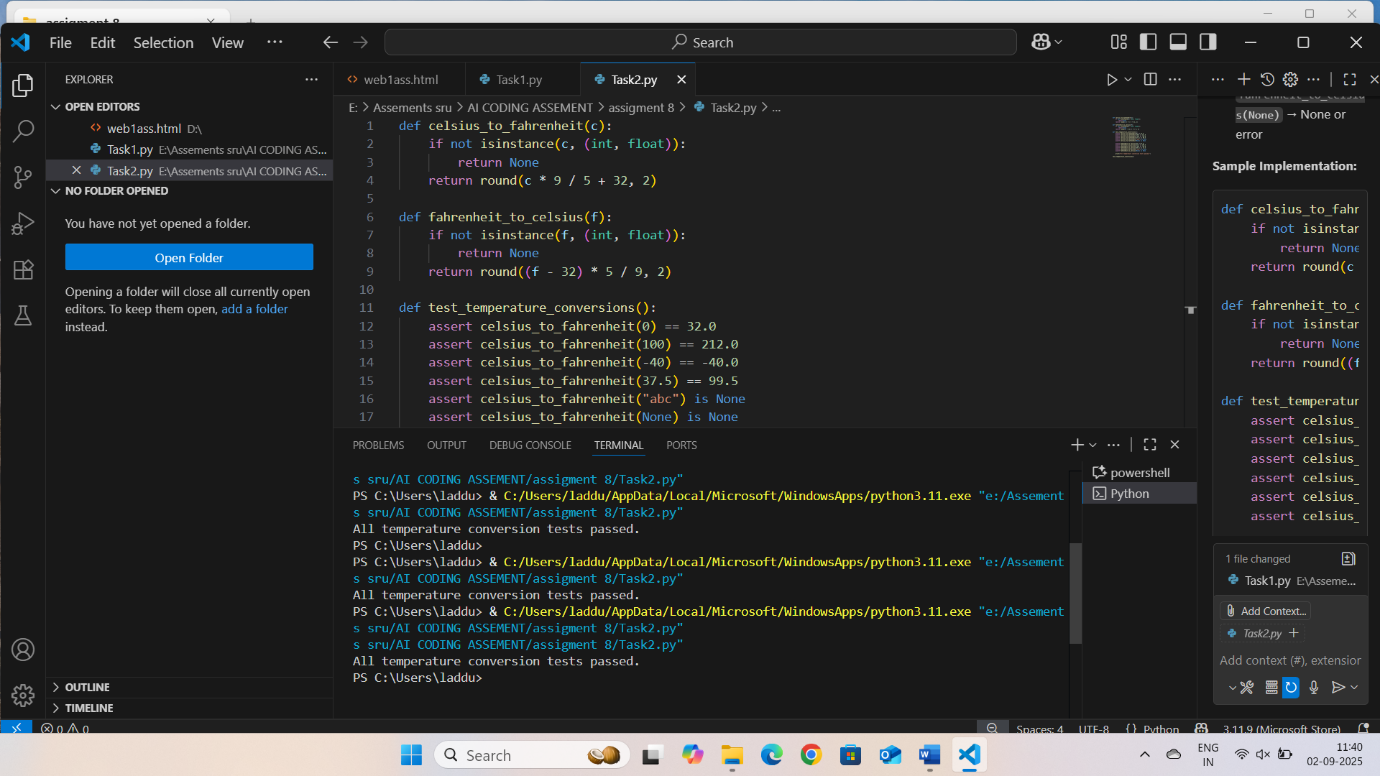
\*Include decimals and invalid inputs like strings or None.

**PROMPT:** Generate at least 10 test cases for celsius\_to\_fahrenheit(c) and fahrenheit\_to\_celsius(f), including known pairs (0°C = 32°F, 100°C = 212°F), decimals, and invalid inputs like strings or None. Then implement both functions and add a test runner to check expected vs actual results.

**CODE:**

****

**OUTPUT:**

****

**OBSERVATION:** The functions celsius\_to\_fahrenheit(c) and fahrenheit\_to\_celsius(f) should accurately handle standard conversions (e.g., 0°C = 32°F), decimal values, and gracefully manage invalid inputs like strings or None without errors.

**TASK DESCRIPTION 3:**

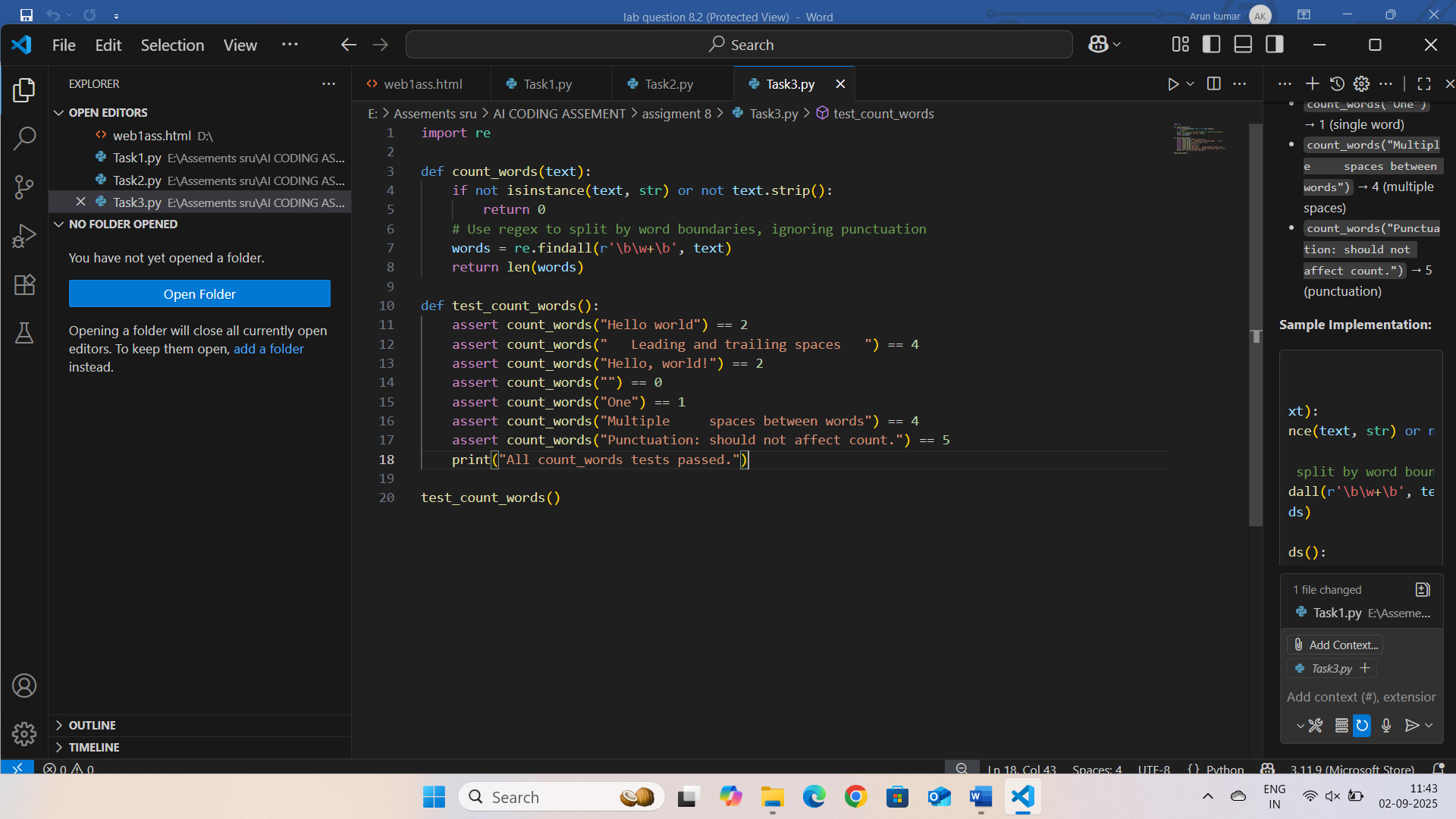
Use AI to write test cases for a function count\_words(text) that returns the number of words in a sentence.

Requirement:

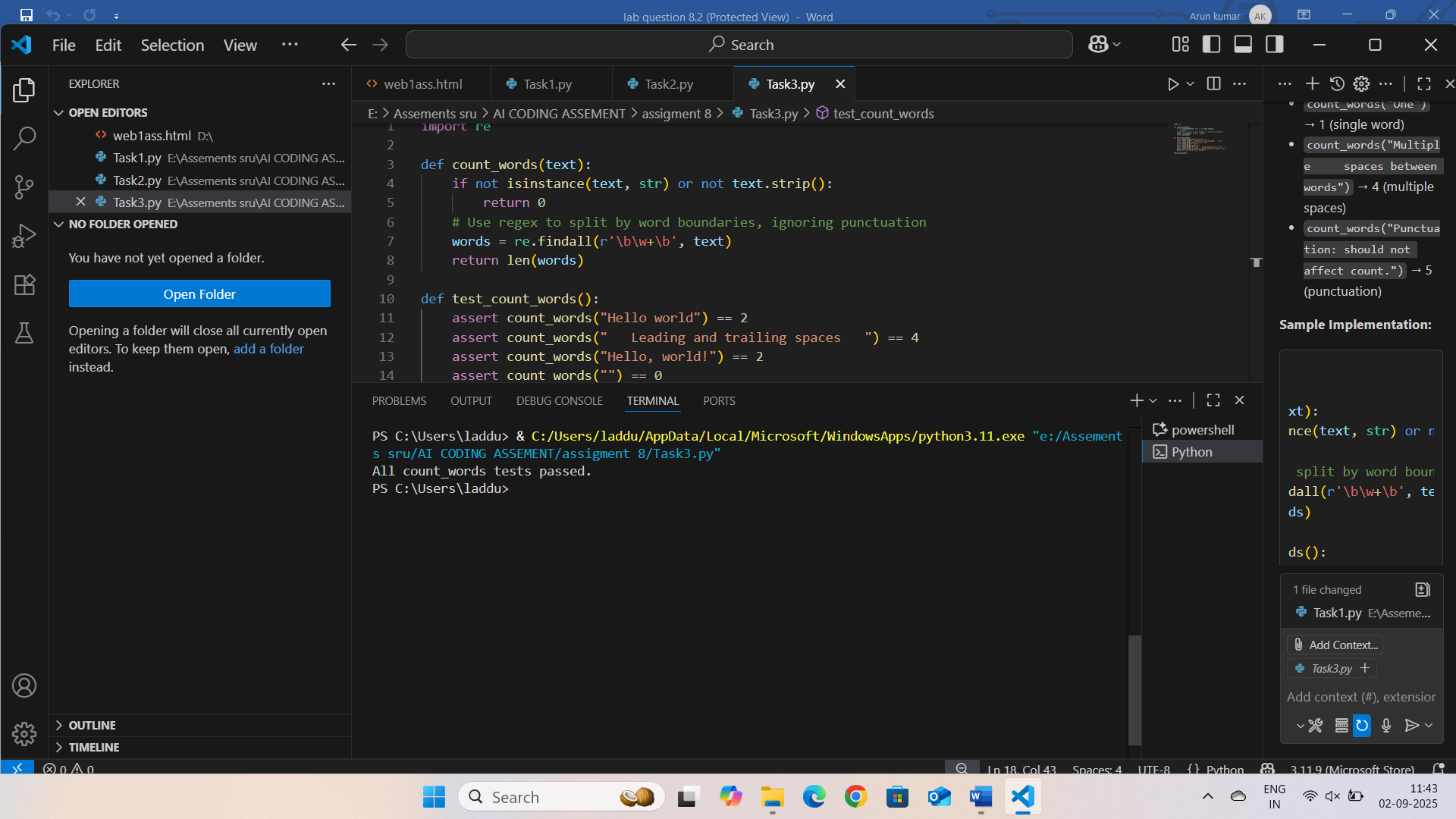
Handle normal text, multiple spaces, punctuation, and empty strings.

**PROMPT:** Generate at least 10 test cases for a Python function count\_words(text) that returns the number of words in a sentence. Include normal text, multiple spaces, punctuation, empty strings, and edge cases like only spaces or special characters.

**CODE:**

****

**OUTPUT:**

****

**OBSERVATION:**

The count\_words(text) function should correctly count words in sentences, handling normal text, multiple spaces, punctuation, empty strings, and edge cases like strings with only spaces or special characters.

**TASK DESCRIPTION 4:**

Generate test cases for a BankAccount class with:

Methods:

deposit(amount)

withdraw(amount)

check\_balance()

Requirements:

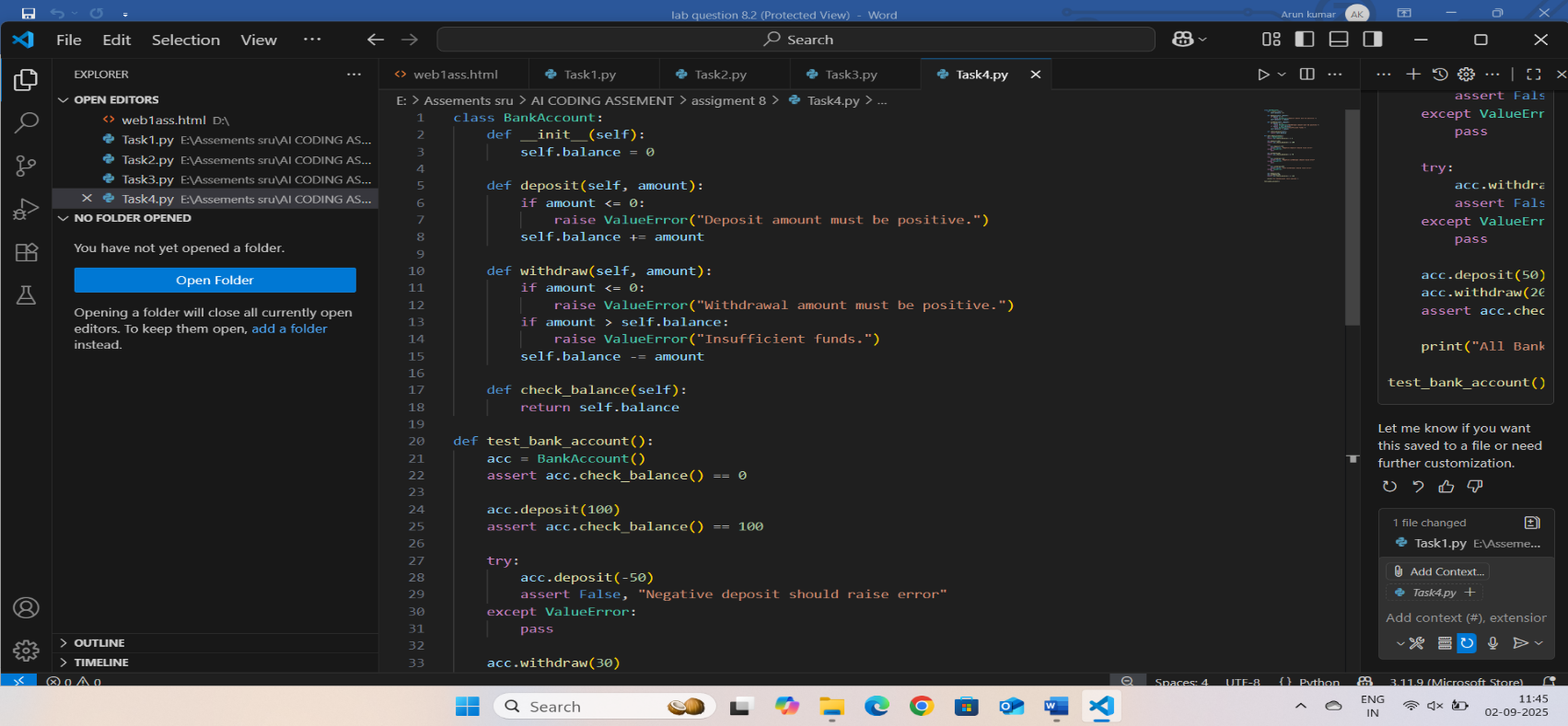
\*Negative deposits/withdrawals should raise an error.

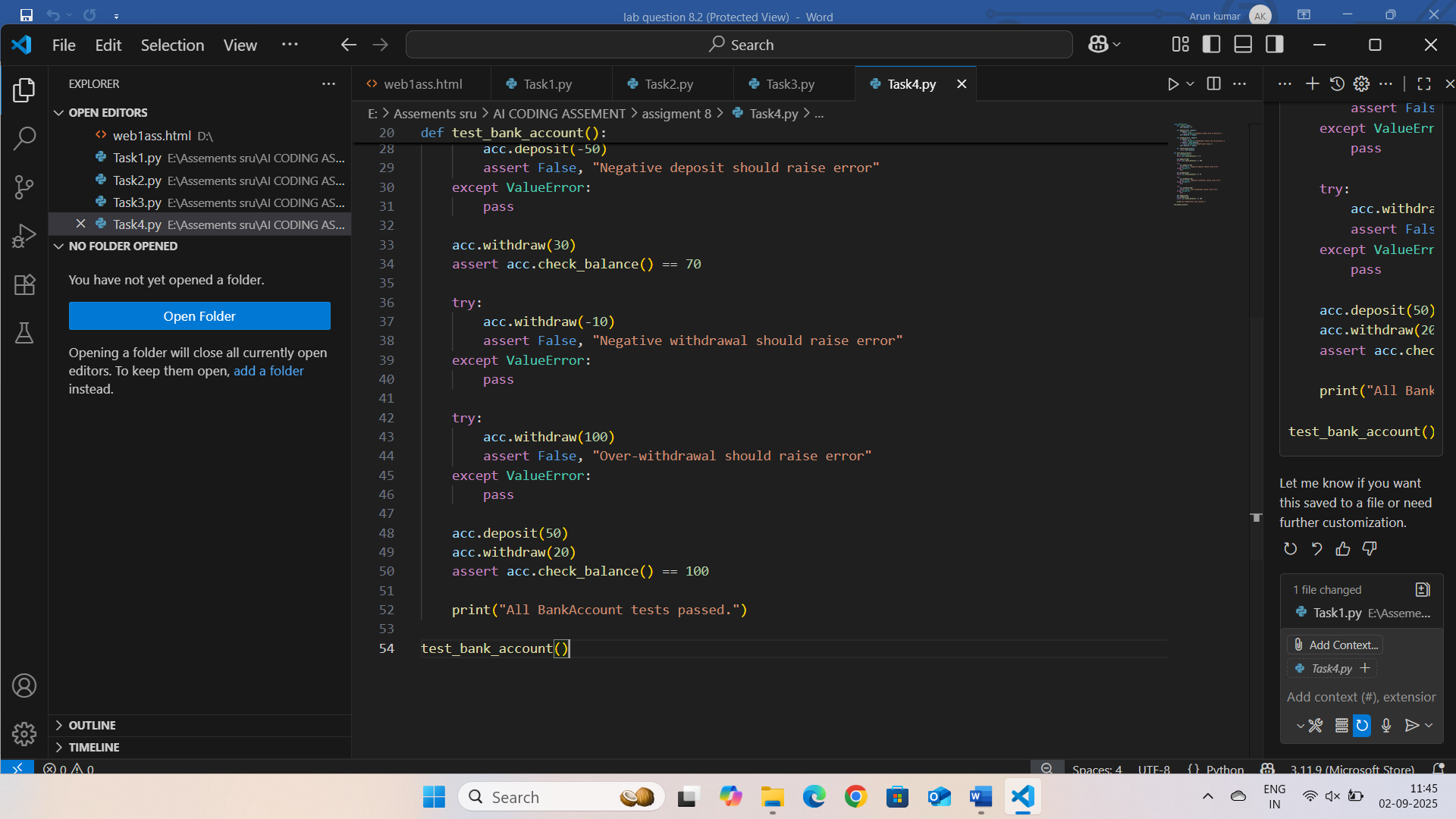
\*Cannot withdraw more than balance.

**PROMPT:**

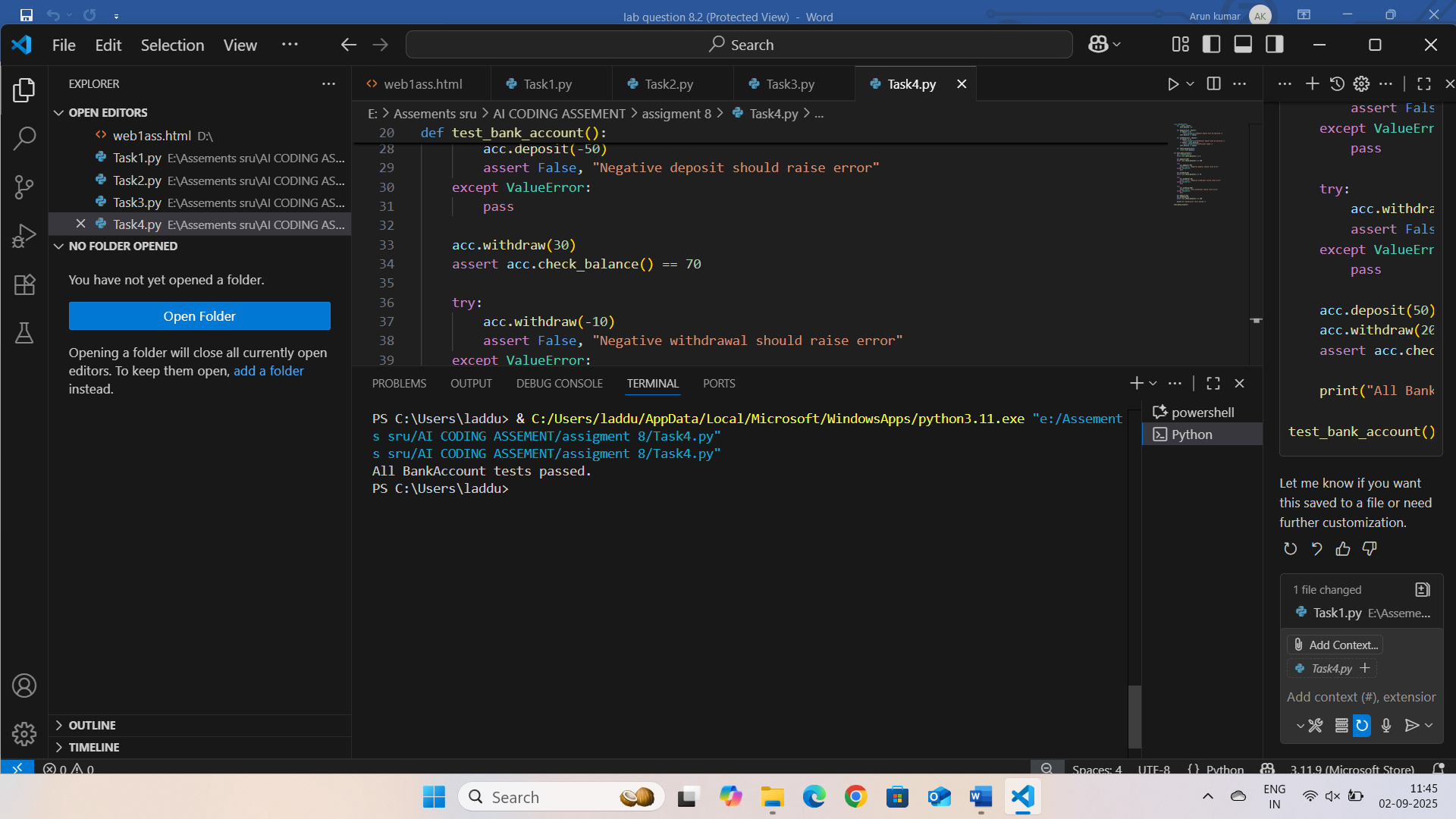
Generate test cases for a BankAccount class with deposit(amount), withdraw(amount), and check\_balance(). Ensure negative amounts raise an error and withdrawals cannot exceed the balance. Include valid, invalid, and edge cases.

**CODE:**

****

****

**OUTPUT:**

****

OBSERVATION:

The BankAccount class must validate inputs and maintain correct balance handling. Negative deposits or withdrawals should trigger errors, and withdrawals cannot exceed available balance. Test cases should confirm proper error handling and correct balance updates.

**TASK DESCRIPTION 5:**

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

Examples:

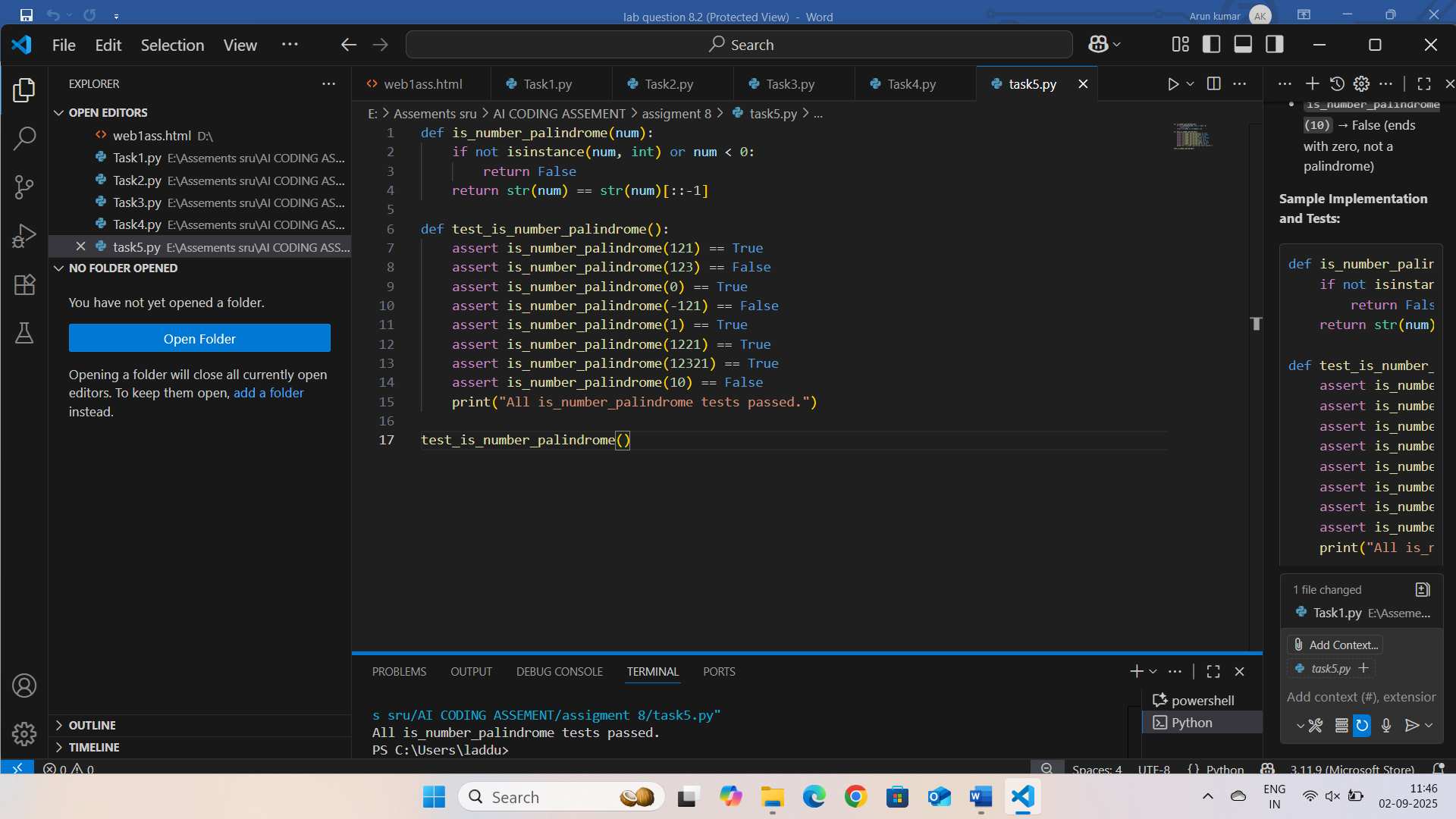
121 → True

123 → False

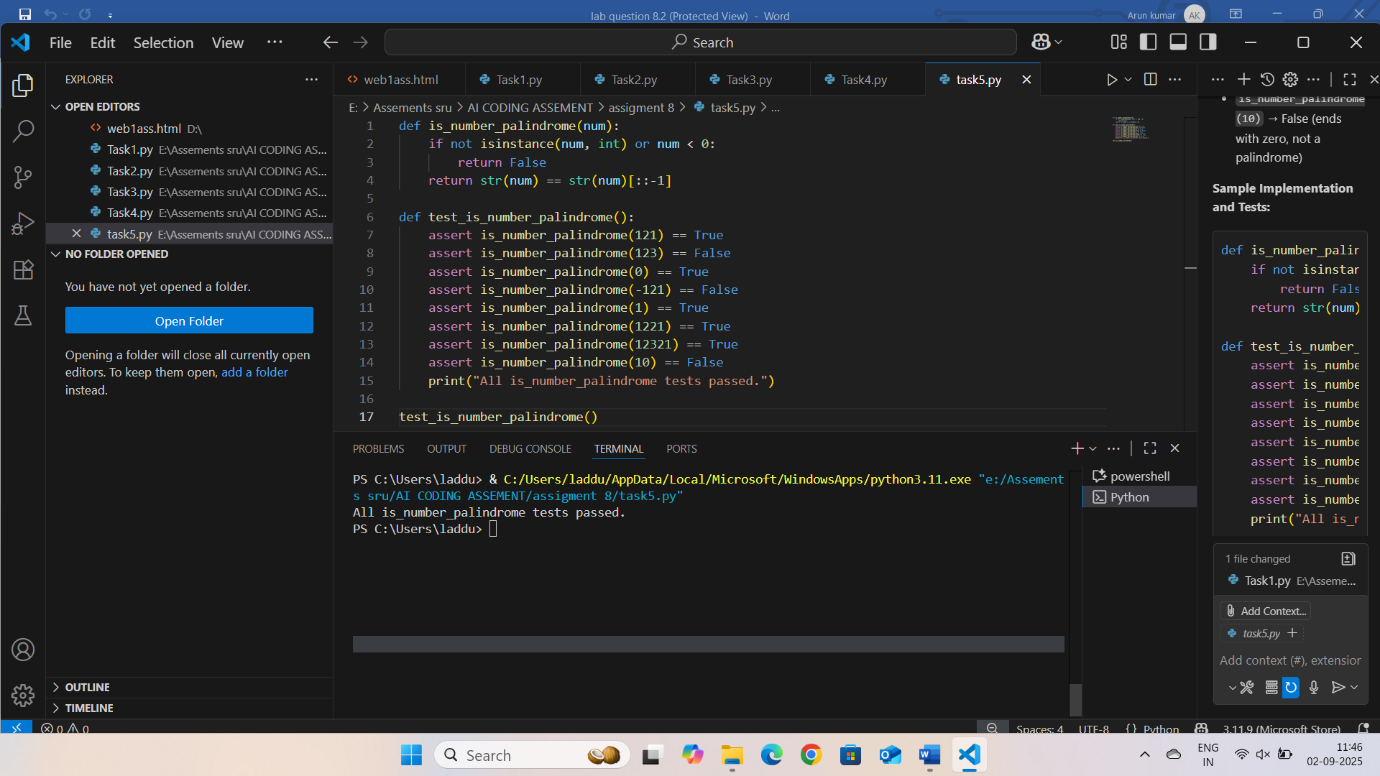
0, negative numbers → handled gracefully

**PROMPT:** Generate test cases for is\_number\_palindrome(num) that checks if an integer reads the same backward. Cover positive palindromes (e.g., 121 → True), non-palindromes (e.g., 123 → False), zero, negative numbers, and large integers.

**CODE:**

****

**OUTPUT:**

****

**OBSERVATION:** The function should correctly identify palindromes for positive integers, return False for non-palindromes, and handle special cases like 0 and negative numbers without errors. Edge cases such as single-digit numbers and large integers must also be validated.