# AI ASSISTED CODING

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

**Roll no:** 2503A51L36

**Name:** Suhana Rehan

**Batch:** 25BTCAICSB20

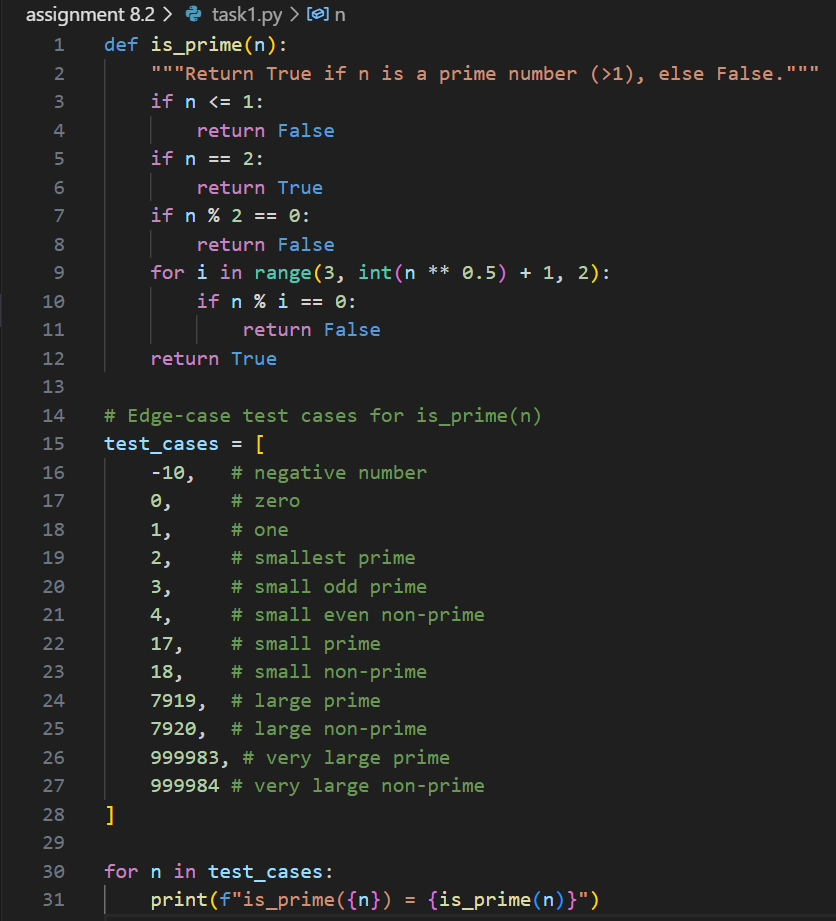
**Task 1:**

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

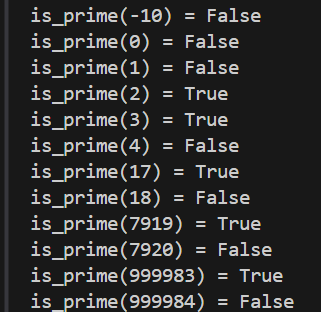
**Prompt:**

Generate and implement an is\_prime(n) function and test it with edge cases (0, 1, 2, negatives, large primes)

**Code:**

****

**Output:**

****

**Observation:**

* The is\_prime(n) function correctly identifies prime numbers greater than 1.
* It returns False for negative numbers, 0, and 1, as these are not prime.
* It returns True for 2 (the smallest prime) and other valid primes, including large primes.
* Even numbers greater than 2 and composite numbers are correctly identified as non-prime.

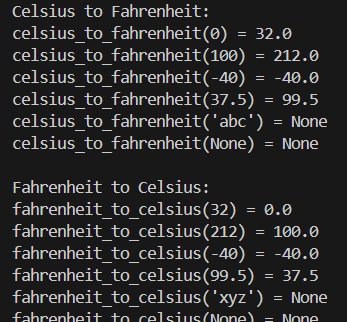
**Task 2:** Ask AI to generate test cases for celsius\_to\_fahrenheit(c) and fahrenheit\_to\_celsius(f).

**Prompt:** Generate test cases for celsius\_to\_fahrenheit(c) and fahrenheit\_to\_celsius(f) covering known pairs (0°C=32°F, 100°C=212°F), decimals, and invalid inputs like strings or None, ensuring safe type handling

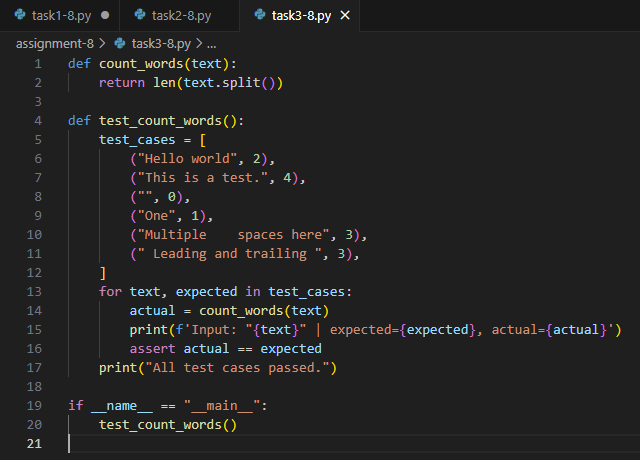
**Code Generated:**



**Output:**

****

**Observation:**

* Both conversion functions handle known values (e.g., 0°C=32°F, 100°C=212°F) and decimals accurately.
* Invalid inputs such as strings and None are safely managed, returning None instead of raising errors.
* The functions are robust against type errors and provide correct results for edge cases, including negative temperatures and decimal values

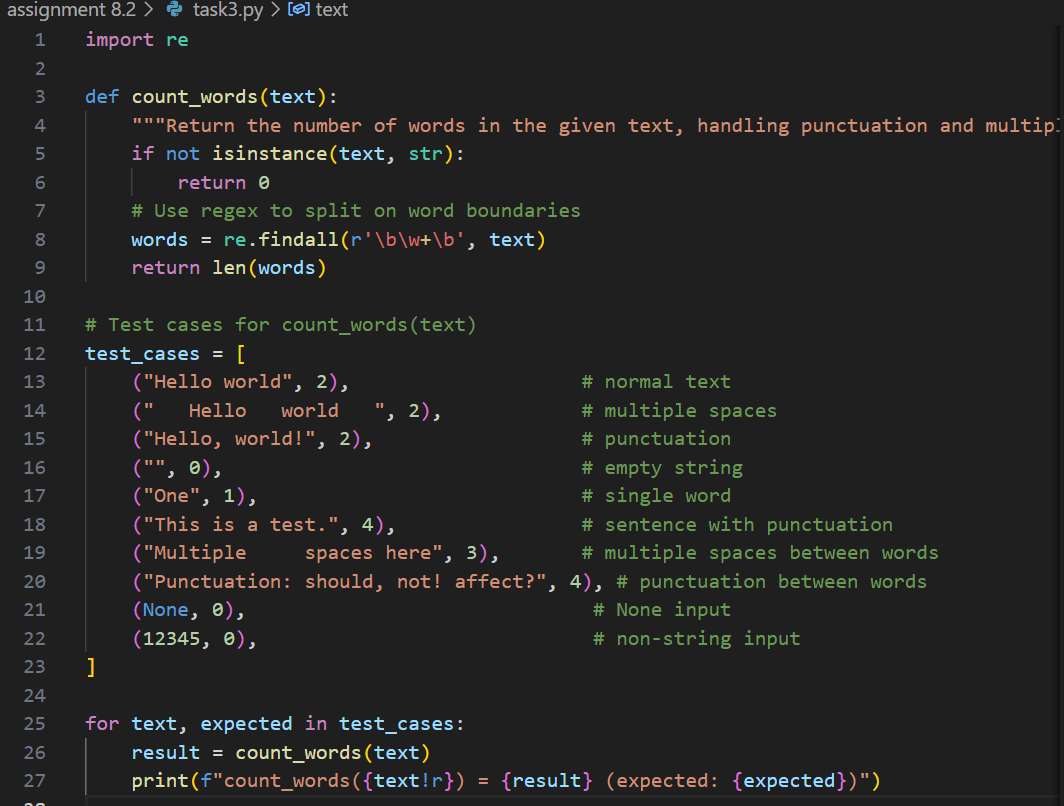
**Task 3:**

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

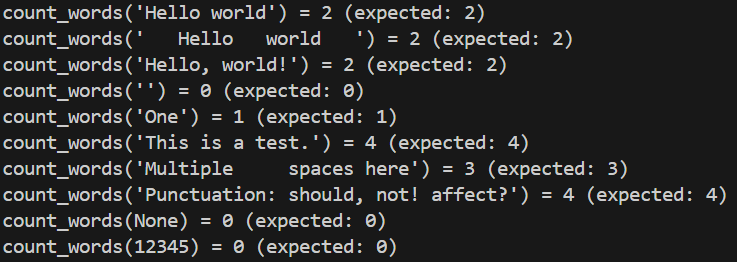
**Prompt:**

Generate test cases for count\_words(text) to handle normal text, multiple spaces, punctuation, and empty strings, ensuring accurate word count and robust validation.

**Code Generated:**

****

**Output:**

****

**Observation:**

* The count\_words(text) function accurately counts words in normal text, handles multiple spaces, and ignores punctuation.
* It returns 0 for empty strings, None, and non-string inputs, ensuring robust validation.
* The function is reliable for various edge cases, including single words and sentences with punctuation.

**Task 4 :**

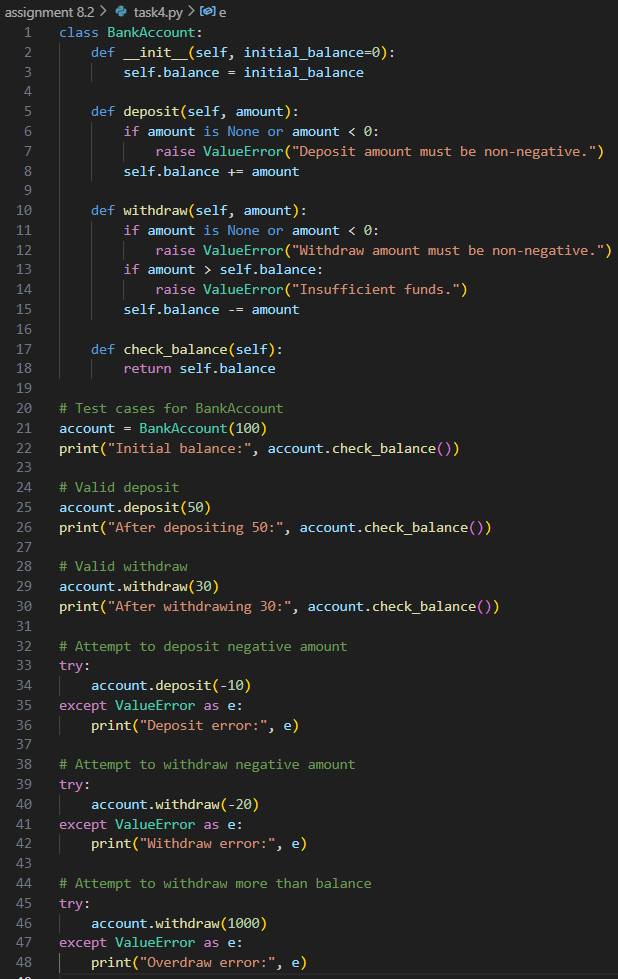
Generate test cases for a BankAccount class with:  
 Methods: deposit(amount)

withdraw(amount)

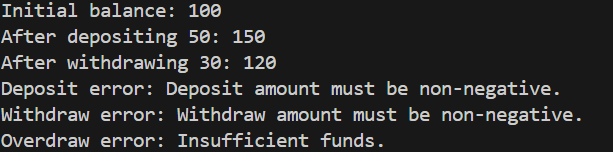
check\_balance().

**Prompt:** Generate test cases for a BankAccount class with deposit, withdraw, and check\_balance methods, ensuring errors for negative amounts and preventing withdrawals beyond balance.

**Code Generated:**

****

**Output:**

****

**Observation:**

* The BankAccount class correctly manages deposits, withdrawals, and balance checks.
* It raises errors for negative deposit/withdrawal amounts and prevents withdrawals that exceed the current balance.
* The class is robust against invalid operations, ensuring safe and accurate account management for all tested scenarios.

**Task 5**:

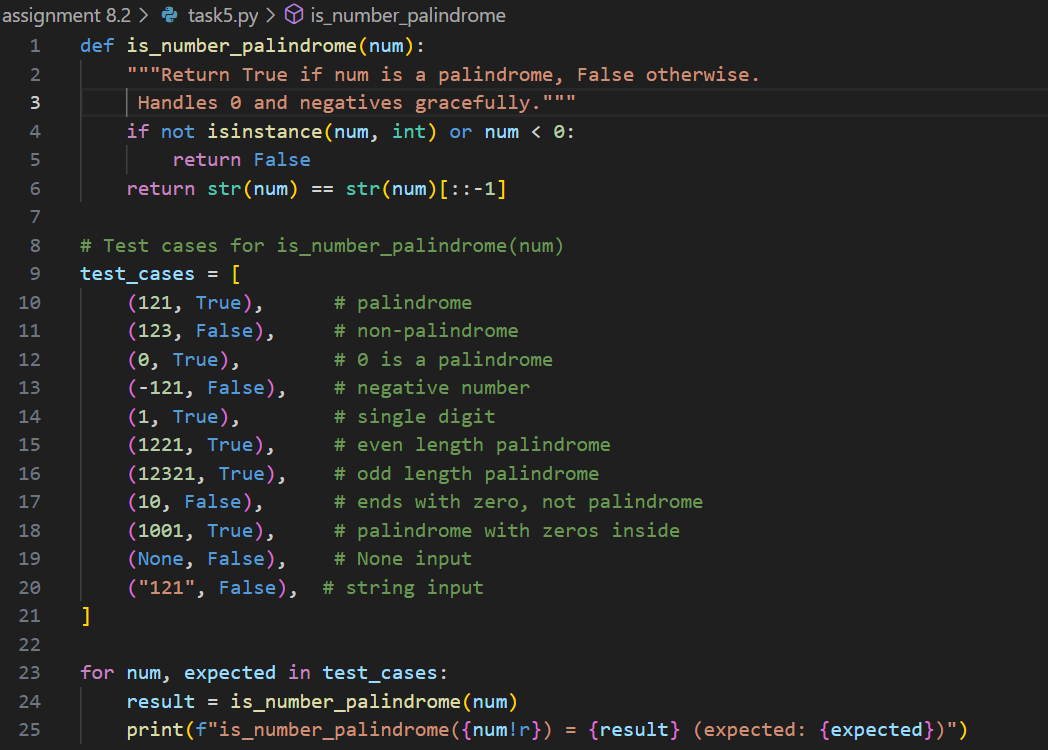
Generate test cases for is\_number\_palindrome(num), which checks if an integer reads the same backward.

**Prompt**:

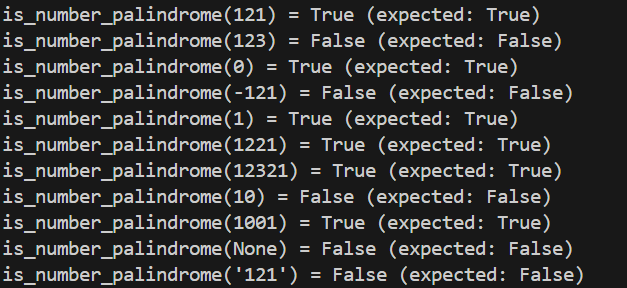
Generate test cases for is\_number\_palindrome(num) to validate palindromes like 121, non-palindromes like 123, and handle 0 and negative numbers gracefully.

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

**Code Generated:**

****

**Output:**

****

**Observation:**

* The is\_number\_palindrome(num) function accurately identifies palindromic numbers, including 0 and single digits.
* It returns False for negative numbers and non-integer inputs, handling these cases gracefully without errors.
* The function is robust for various edge cases, such as numbers with zeros inside and string/None inputs.