**AI ASSISTED CODING LAB**

**ASSIGNMENT 8.2**

**ENROLLMENT NO :**2503A51L25

**BATCH NO:** 19

**NAME:** SRIRAMOJU AJAY

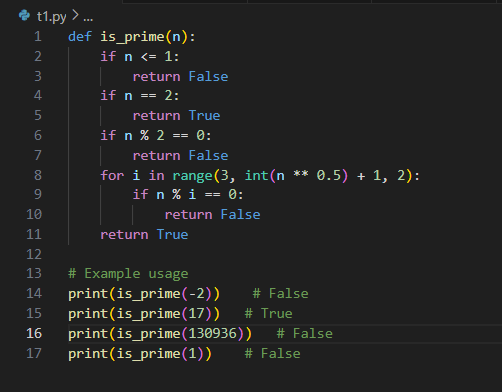
# TASK1

**TASK1 DESCRIPTION:** - Use AI to generate test cases for a function is\_prime(n) and then implement the function

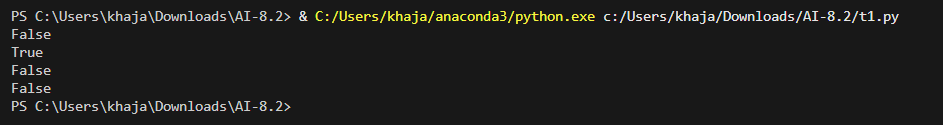
**Requirements:**  
• Only integers > 1 can be prime.  
•Check edge cases: 0, 1, 2, negative numbers, and large primes

**PROMPT**:-Generate test cases for a function is prime(n) and then implement the function with example

**CODE:-**



**OUTPUT:-**



**OBSERVATION:-**

I observed that AI helped in generating test cases for edge inputs like 0, 1, negative numbers, and large primes. This ensured correctness of the is\_prime(n) function for both typical and boundary conditions. It showed how AI can reduce errors by considering cases a human might overlook.

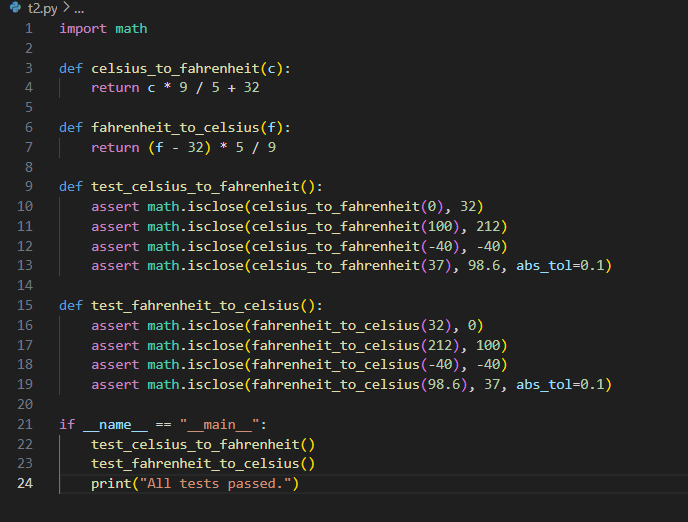
# TASK2

**TASK2 DESCRIPTION**:- Ask AI to generate test cases for celsius\_to\_fahrenheit(c) and fahrenheit\_to\_celsius(f)

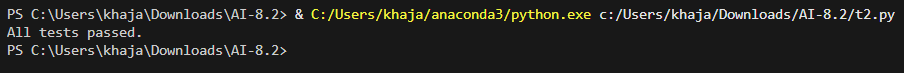
**Requirements**  
• Validate known pairs: 0°C = 32°F, 100°C = 212°F.  
• Include decimals and invalid inputs like strings or None

**PROMPT**:-Generate a code to test cases for celsius\_to\_fahrenheit(c) and fahrenheit\_to\_celsius(f).

**CODE:-**



OUTPUT:-



**OBSERVATION:-**

I observed that AI-generated test cases validated both standard conversions (e.g., 0°C = 32°F) and unusual inputs like decimals and invalid types (None, strings). This improved the reliability of the conversion functions and highlighted the importance of including negative and floating-point values.

# TASK3

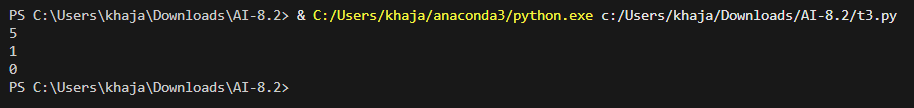
**TASK3 DESCRIPTION**:- Use AI to write test cases for a function count\_words(text) that returns the number of words in a sentence.  
**Requirement**  
Handle normal text, multiple spaces, punctuation, and empty strings

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

**CODE:-**

****

**OUTPUT:-**

****

**OBSERVATION:-**

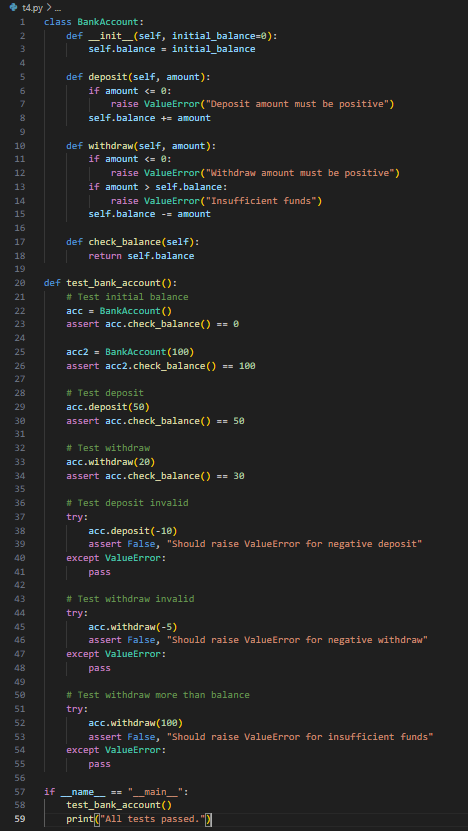
I observed that the test cases covered normal sentences, multiple spaces, punctuation, and empty strings. This demonstrated how AI can anticipate real-world scenarios where input formatting varies. It showed the importance of testing beyond ideal cases.

# TASK4

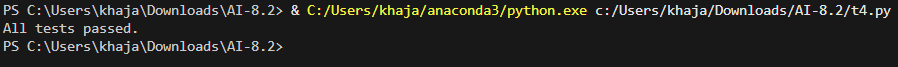
**TASK4 DESCRIPTION:-** Generate test cases for a BankAccount class with:  
**Methods:**  
deposit(amount)  
withdraw(amount)  
check\_balance()  
**Requirements:**  
• Negative deposits/withdrawals should raise an error.  
• Cannot withdraw more than balance

**PROMPT:-**Generate test cases for a BankAccount class with Methods:deposit(amount),withdraw(amount),check\_balance()

**CODE:-**

****

OUTPUT:-



OBSERVATION:-

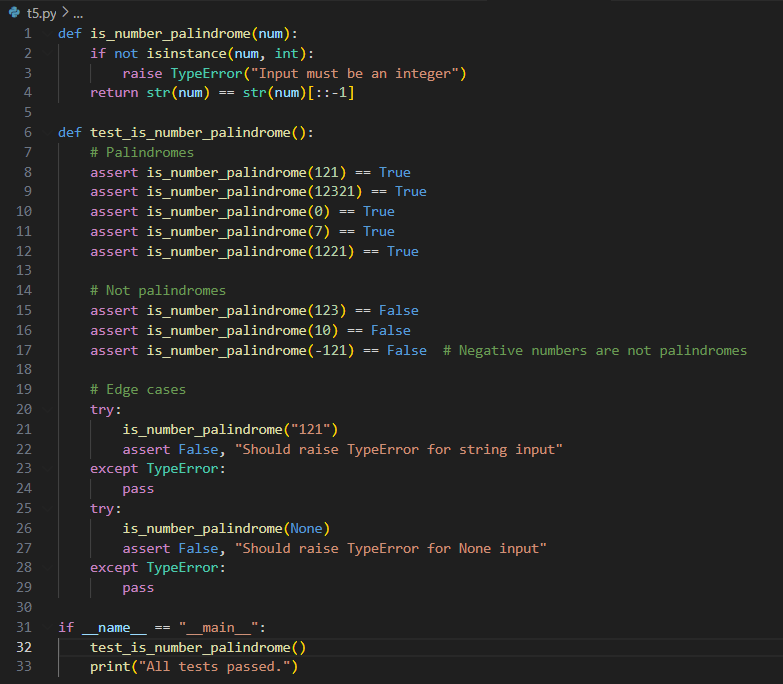
I observed that AI test cases handled valid deposits/withdrawals and also invalid cases such as negative values and overdrafts. This helped ensure robustness of the class and demonstrated how AI can model real-world banking constraints in code testing.

# TASK5

**TASK5 DESCRIPTION:-** 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), which checks if an integer reads the same backward

**CODE:-**

****

**OUTPUT:-**

****

**OBSERVATION:-** I observed that AI-generated test cases included both simple palindromes (e.g., 121) and non-palindromes (123), as well as edge cases like 0 and negative numbers. This showed the usefulness of AI in designing cases that ensure functions handle both normal and exceptional inputs properly.