# ASSIGNMENT – 8.2

# NAME : SAIRAM.Y

# ROLL NO. : 2403A51415

# BATCH : 16

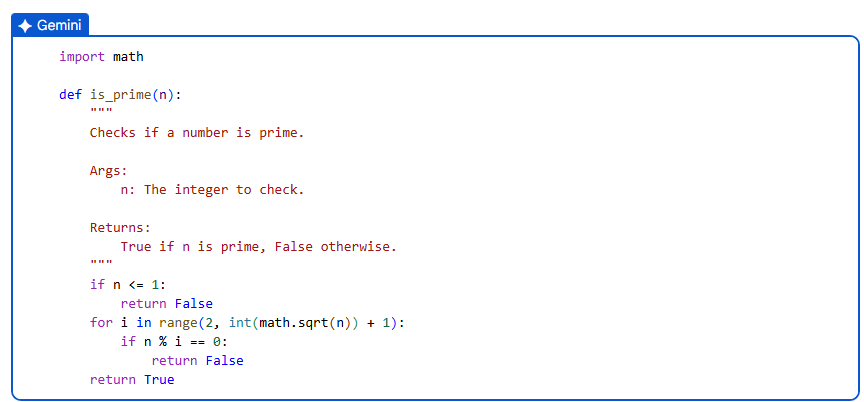
# COURSE : AI-ASSISSTED . CODING

TASK 1 :

PROMPT :

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

SCREENSHOT AND OUTPUT :



EXPLANATION :

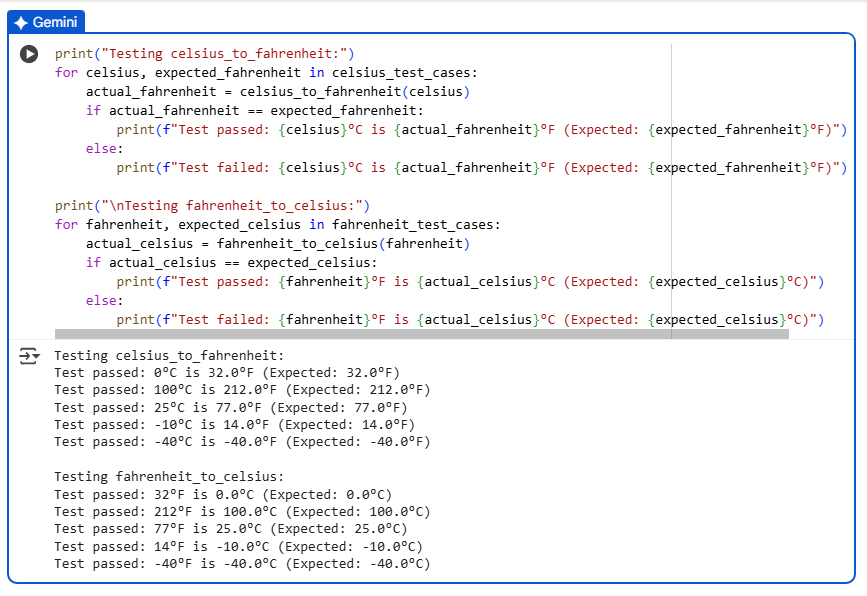
This Python code defines a function called is\_prime that takes an integer n as input and returns True if n is a prime number, and False otherwise. It first handles the base cases: numbers less than or equal to 1 are not prime. For numbers greater than 1, it iterates through integers starting from 2 up to the square root of n (inclusive). If it finds any number in this range that divides n evenly, it means n is not prime, and the function returns False. If the loop completes without finding any divisors, it means n is prime, and the function returns True. The math.sqrt() function is used to optimize the process, as a number n will not have any divisors greater than its square root if it doesn't have any divisors smaller than its square root.

TASK 2 :

PROMPT:

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

SCREENSHOT AND OUTPUT :



EXPLANATION :

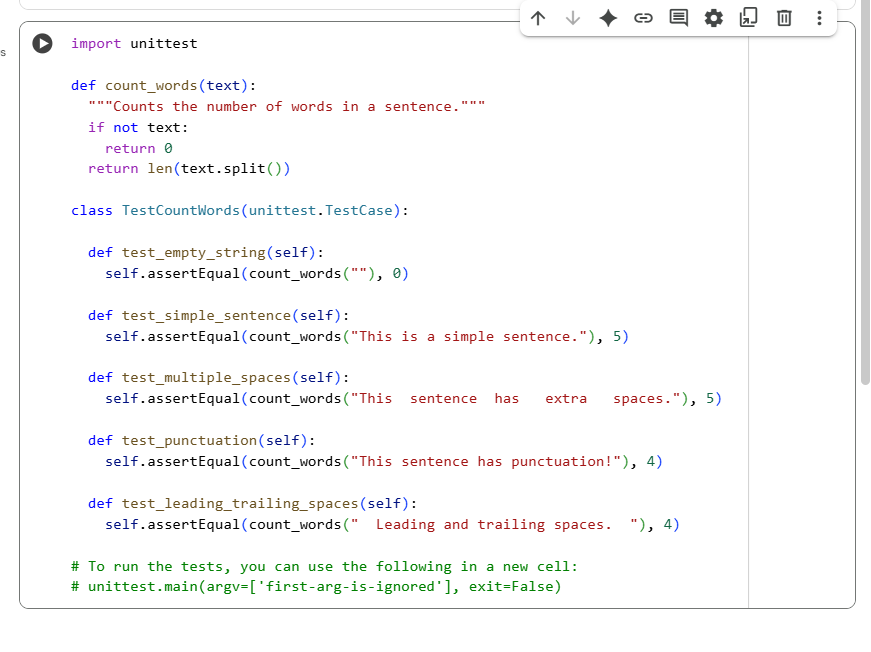
This code block tests the celsius\_to\_fahrenheit and fahrenheit\_to\_celsius functions using predefined test cases. It first prints a heading "Testing celsius\_to\_fahrenheit:" and then iterates through the celsius\_test\_cases list. For each test case (which is a tuple of Celsius value and expected Fahrenheit value), it calls the celsius\_to\_fahrenheit function with the Celsius value and compares the actual result with the expected Fahrenheit value. It then prints whether the test case passed or failed, along with the input, actual output, and expected output. After testing the Celsius to Fahrenheit conversion, it does the same for the fahrenheit\_to\_celsius function using the fahrenheit\_test\_cases list, printing a heading "Testing fahrenheit\_to\_celsius:" and reporting the test results for each case.

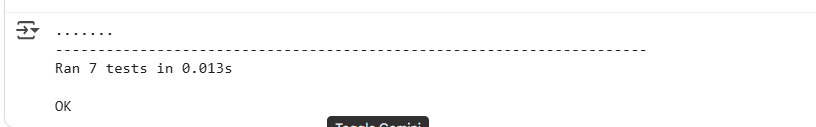
TASK 3 :

PROMPT:

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

SCREENSHOT AND OUTPUT :





EXPLANATION :

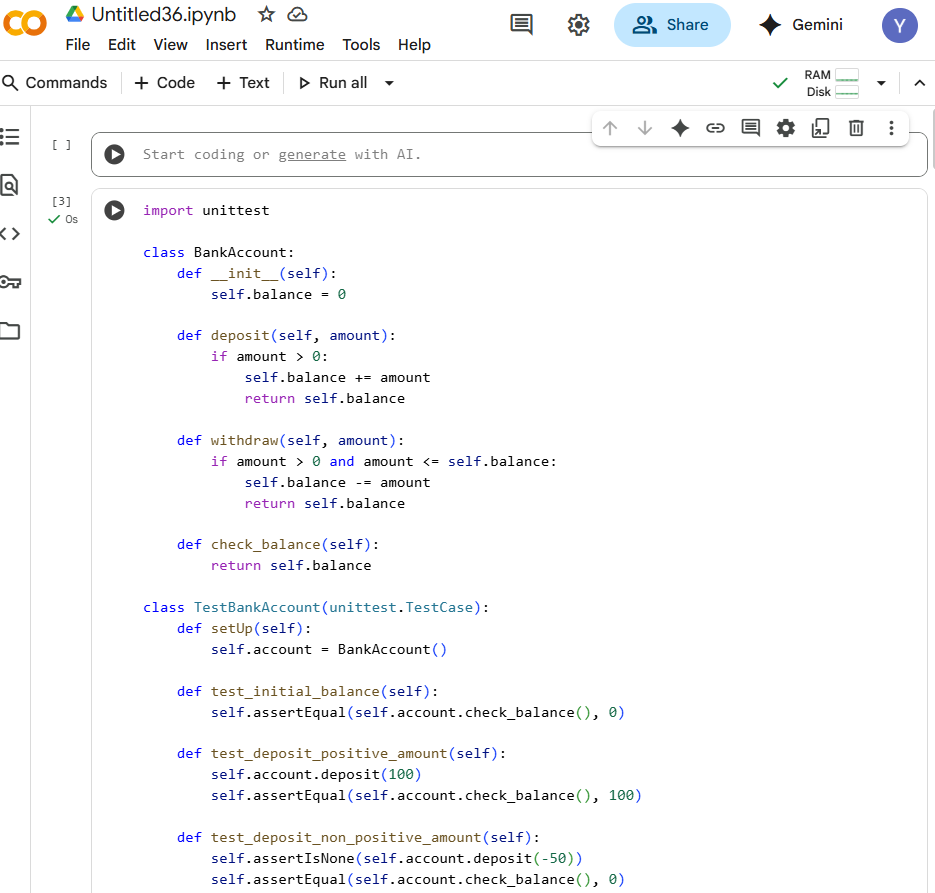
This code defines a function count\_words that counts the number of words in a given string by splitting the string by spaces. It also includes a unittest.TestCase class TestCountWords with several test methods to verify the correctness of the count\_words function under different scenarios, such as empty strings, simple sentences, multiple spaces, punctuation, and leading/trailing spaces.

TASK 4 :

PROMPT:

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

SCREENSHOT AND OUTPUT :





EXPLANATION :

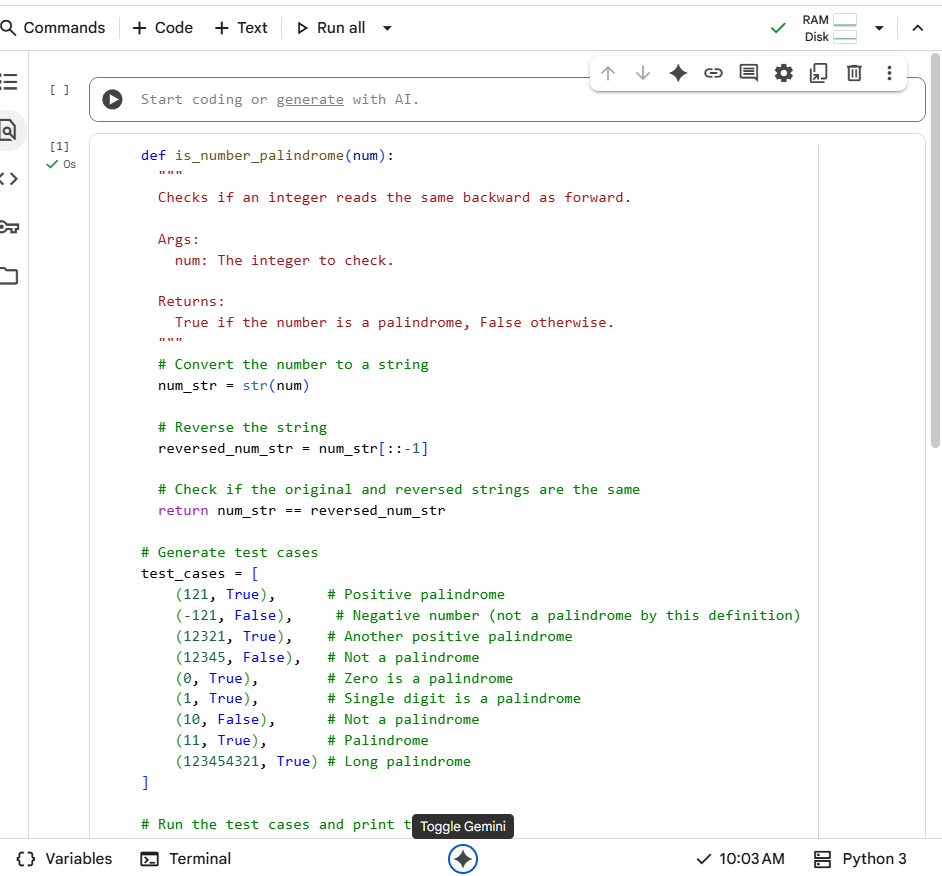
This code defines a BankAccount class with methods for depositing, withdrawing, and checking the balance. It also includes a TestBankAccount class using the unittest framework to verify the functionality of the BankAccount class with various test cases.

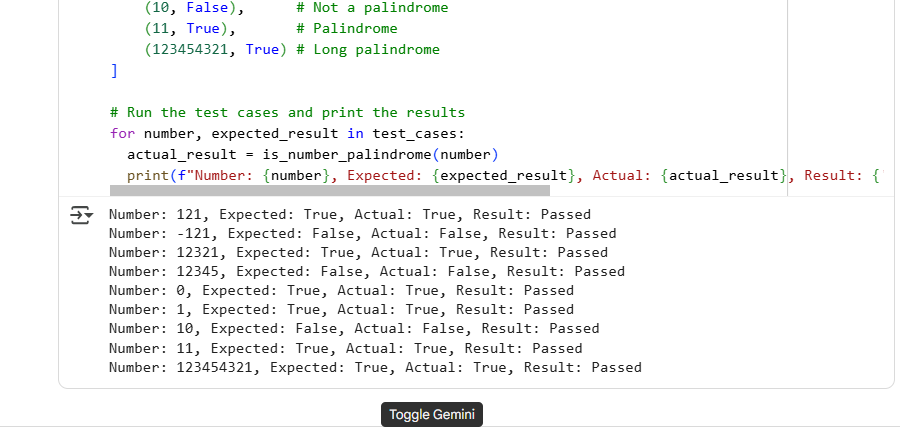
TASK 5 :

PROMPT:

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

SCREENSHOT AND OUTPUT :





EXPLANATION :

This Python code defines a function is\_number\_palindrome that determines if an integer is a palindrome by converting it to a string and comparing it with its reversed version. The code then includes a set of diverse test cases to validate the function's behavior, including positive, negative, single-digit, and multi-digit numbers, and prints the results of running these tests to show whether each case passed or failed.