

AI ASSISTED CODING LAB 8.2

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

ENROLLMENT NO: 2403A52006

NAME: MOHAMMED MUSTAFA

COURSE: CSM(AI/ML)

BATCH: 01

DATE: 23-09-2025

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

TEST_CASES_PROMPT:

Generate some test cases only for checking whether the given number is prime or not.

Here are some test cases for checking whether a number is prime or not:

- **Input: 1, Expected Output: False** (1 is not a prime number)
- **Input: 2, Expected Output: True** (2 is the only even prime number)
- **Input: 3, Expected Output: True**
- **Input: 4, Expected Output: False** ($4 = 2 * 2$)
- **Input: 5, Expected Output: True**
- **Input: 6, Expected Output: False** ($6 = 2 * 3$)
- **Input: 7, Expected Output: True**
- **Input: 10, Expected Output: False** ($10 = 2 * 5$)
- **Input: 11, Expected Output: True**
- **Input: 17, Expected Output: True**
- **Input: 29, Expected Output: True**
- **Input: 53, Expected Output: True**
- **Input: 100, Expected Output: False** ($100 = 10 * 10$)
- **Input: 121, Expected Output: False** ($121 = 11 * 11$)
- **Input: 0, Expected Output: False** (0 is not a prime number)
- **Input: -5, Expected Output: False** (Negative numbers are not prime)

USED PROMPT:

Generate a python code to check whether the given number is prime or not use test cases given above as : number > 1 to be prime and if ≤ 0 not a prime and give output as incorrect input. take input from the user

```
def is_prime(n):
    if n <= 1:
        return False
    for i in range(2, int(n**0.5) + 1):
        if n % i == 0:
            return False
    return True

try:
    num = int(input("Enter an integer to check if it's prime: "))

    if num <= 0:
        print("Incorrect input: Please enter an integer greater than 0.")
    elif is_prime(num):
        print(f"{num} is a prime number.")
    else:
        print(f"{num} is not a prime number.")
except ValueError:
    print("Invalid input. Please enter an integer.")
```

Enter an integer to check if it's prime: 9.2
Invalid input. Please enter an integer.

Task Description 2: (Loops)

- Ask AI to generate test cases for `celsius_to_fahrenheit(c)` and `fahrenheit_to_celsius(f)`.

Requirements:

- Validate known pairs: $0^{\circ}\text{C} = 32^{\circ}\text{F}$, $100^{\circ}\text{C} = 212^{\circ}\text{F}$.
- Include decimals and invalid inputs like strings or None

TEST CASES PROMPT:

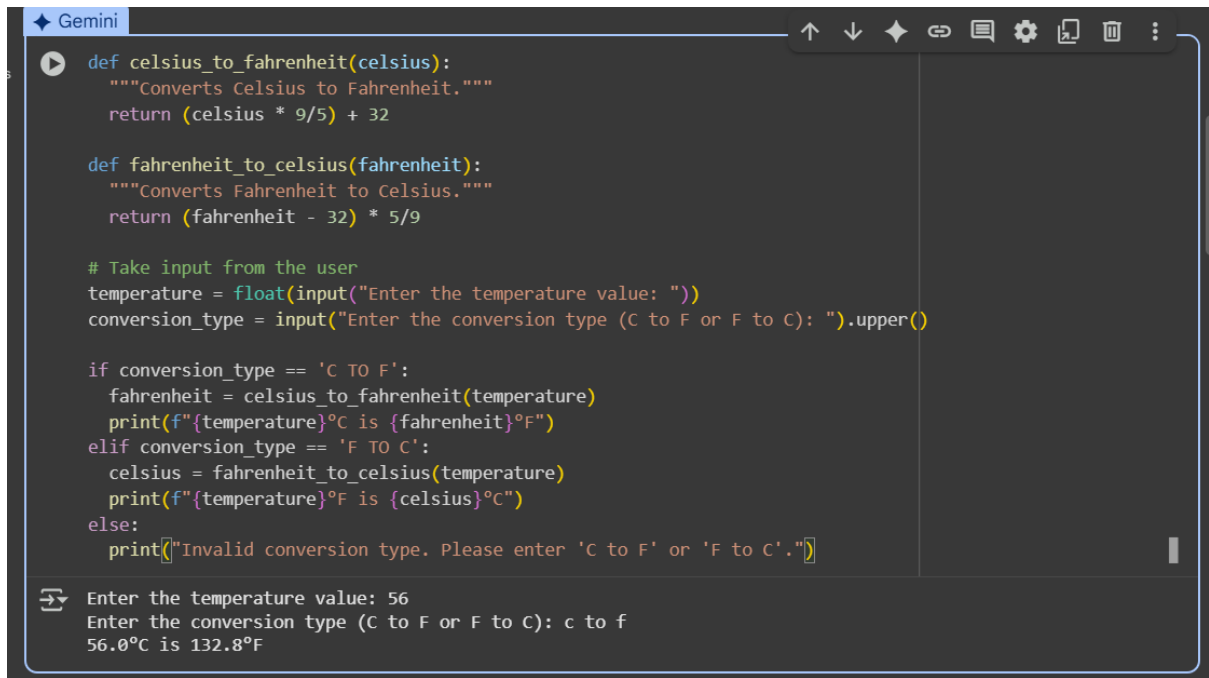
Generate some test cases only to convert temperature from Celsius to Fahrenheit and Fahrenheit to Celsius.

Here are some test cases for checking whether a number is prime or not:

- **Input: 1, Expected Output: False** (1 is not a prime number)
- **Input: 2, Expected Output: True** (2 is the only even prime number)
- **Input: 3, Expected Output: True**
- **Input: 4, Expected Output: False** ($4 = 2 * 2$)
- **Input: 5, Expected Output: True**
- **Input: 6, Expected Output: False** ($6 = 2 * 3$)
- **Input: 7, Expected Output: True**
- **Input: 10, Expected Output: False** ($10 = 2 * 5$)
- **Input: 11, Expected Output: True**
- **Input: 17, Expected Output: True**
- **Input: 29, Expected Output: True**
- **Input: 53, Expected Output: True**
- **Input: 100, Expected Output: False** ($100 = 10 * 10$)
- **Input: 121, Expected Output: False** ($121 = 11 * 11$)
- **Input: 0, Expected Output: False** (0 is not a prime number)
- **Input: -5, Expected Output: False** (Negative numbers are not prime)

USED PROMPT:

generate a python code to convert temperature from Celsius to Fahrenheit and Fahrenheit to Celsius using the above test cases and take input from the user



```
def celsius_to_fahrenheit(celsius):
    """Converts Celsius to Fahrenheit."""
    return (celsius * 9/5) + 32

def fahrenheit_to_celsius(fahrenheit):
    """Converts Fahrenheit to Celsius."""
    return (fahrenheit - 32) * 5/9

# Take input from the user
temperature = float(input("Enter the temperature value: "))
conversion_type = input("Enter the conversion type (C to F or F to C): ").upper()

if conversion_type == 'C TO F':
    fahrenheit = celsius_to_fahrenheit(temperature)
    print(f"{temperature}°C is {fahrenheit}°F")
elif conversion_type == 'F TO C':
    celsius = fahrenheit_to_celsius(temperature)
    print(f"{temperature}°F is {celsius}°C")
else:
    print("Invalid conversion type. Please enter 'C to F' or 'F to C'.")
```

Enter the temperature value: 56
Enter the conversion type (C to F or F to C): c to f
56.0°C is 132.8°F

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

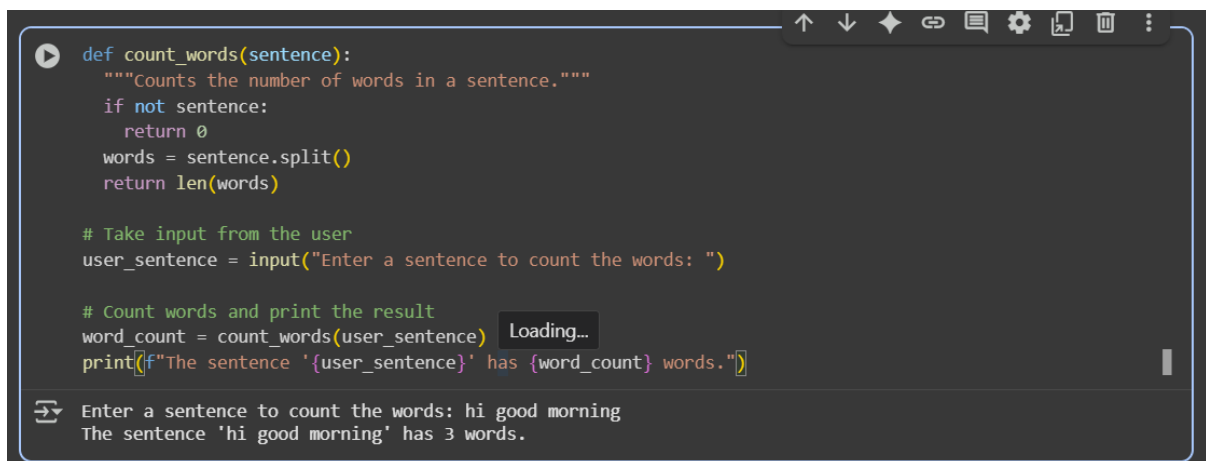
TEST_CASES_PROMPT:

give me some test cases for a code which returns the count of the number of words in a sentence

Here are some test cases for a code that returns the count of words in a sentence:

- **Input:** "", **Expected Output:** 0 (Empty string)
- **Input:** "hello", **Expected Output:** 1 (Single word)
- **Input:** "hello world", **Expected Output:** 2 (Two words)
- **Input:** " hello world ", **Expected Output:** 2 (Leading and trailing spaces)
- **Input:** "hello world", **Expected Output:** 2 (Multiple spaces between words)
- **Input:** "Hello, world!", **Expected Output:** 2 (Sentence with punctuation)
- **Input:** "This is a sentence with five words.", **Expected Output:** 5
- **Input:** " Word with spaces ", **Expected Output:** 3 (Leading/trailing and multiple spaces)
- **Input:** "One.", **Expected Output:** 1 (Single word with punctuation)
- **Input:** " Two words. ", **Expected Output:** 2 (Leading/trailing spaces and punctuation)

USED PROMPT: using the same test cases generate a python code to check the number of words in a sentence.



```
def count_words(sentence):  
    """Counts the number of words in a sentence."""  
    if not sentence:  
        return 0  
    words = sentence.split()  
    return len(words)  
  
# Take input from the user  
user_sentence = input("Enter a sentence to count the words: ")  
  
# Count words and print the result  
word_count = count_words(user_sentence)  
print(f"The sentence '{user_sentence}' has {word_count} words.")
```

Enter a sentence to count the words: hi good morning
The sentence 'hi good morning' has 3 words.

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

TEST CASES PROMPT:

generate test cases for bank account class with methods: deposit(amount) , withdraw(amount) , check_balance() negative deposits/withdrawals should raise an error and cannot withdraw more than balance.

Here are the test cases for a bank account with deposit, withdraw, and check balance operations:

- **Initial Balance:** 0
- **Test Cases:**
 - **Initial Account Creation:** Expected Balance: 0
 - **Deposit 100:** Expected Balance: 100
 - **Deposit 50, then Deposit 75:** Expected Balance: 125
 - **Initial Balance 200, Withdraw 50:** Expected Balance: 150
 - **Initial Balance 300, Withdraw 100, then Withdraw 75:** Expected Balance: 125
 - **Initial Balance 150, Withdraw 150:** Expected Balance: 0
 - **Initial Balance 100, Deposit -50:** Expected to raise an error (e.g., ValueError), Balance remains 100
 - **Initial Balance 100, Withdraw -50:** Expected to raise an error (e.g., ValueError), Balance remains 100
 - **Initial Balance 100, Withdraw 150:** Expected to raise an error (e.g., ValueError), Balance remains 100
 - **Initial Balance 100, Withdraw 0:** Expected Balance: 100
 - **Initial Balance 100, Deposit 0:** Expected Balance: 100

USED PROMPT:

generate a python code for bank account class with methods : deposit(amount), withdraw (amount), and check balance().using the above test cases above and take input from the use

```

class BankAccount:
    def __init__(self):
        self.balance = 1000
        print("Bank account created with a balance of 1000.")

    def deposit(self, amount):
        if amount <= 0:
            raise ValueError("Deposit amount must be positive.")
        self.balance += amount
        print(f"Deposited {amount}. Current balance: {self.balance}")

    def withdraw(self, amount):
        if amount <= 0:
            raise ValueError("Withdrawal amount must be positive.")
        if amount > self.balance:
            raise ValueError("Insufficient funds.")
        self.balance -= amount
        print(f"Withdrew {amount}. Current balance: {self.balance}")

    def check_balance(self):
        print(f"Current balance: {self.balance}")

# Create a bank account
account = BankAccount()

# Interactive session with the user
while True:
    action = input("Enter action (1: deposit, 2: withdraw, 3: balance, exit): ")

```

```

    if action == '1':
        try:
            amount = float(input("Enter deposit amount: "))
            account.deposit(amount)
        except ValueError as e:
            print(f"Error: {e}")
    elif action == '2':
        try:
            amount = float(input("Enter withdrawal amount: "))
            account.withdraw(amount)
        except ValueError as e:
            print(f"Error: {e}")
    elif action == '3':
        account.check_balance()
    elif action.lower() == 'exit':
        print("Exiting.")
        break
    else:
        print("Invalid action. Please enter 1, 2, 3, or exit.")

```

```

Bank account created with a balance of 1000.
Enter action (1: deposit, 2: withdraw, 3: balance, exit): 2
Enter withdrawal amount: 650
Withdrew 650.0. Current balance: 350.0
Enter action (1: deposit, 2: withdraw, 3: balance, exit): 1
Enter deposit amount: 200
Deposited 200.0. Current balance: 550.0
Enter action (1: deposit, 2: withdraw, 3: balance, exit): exit
Exiting.

```

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

TEST CASES PROMPT:

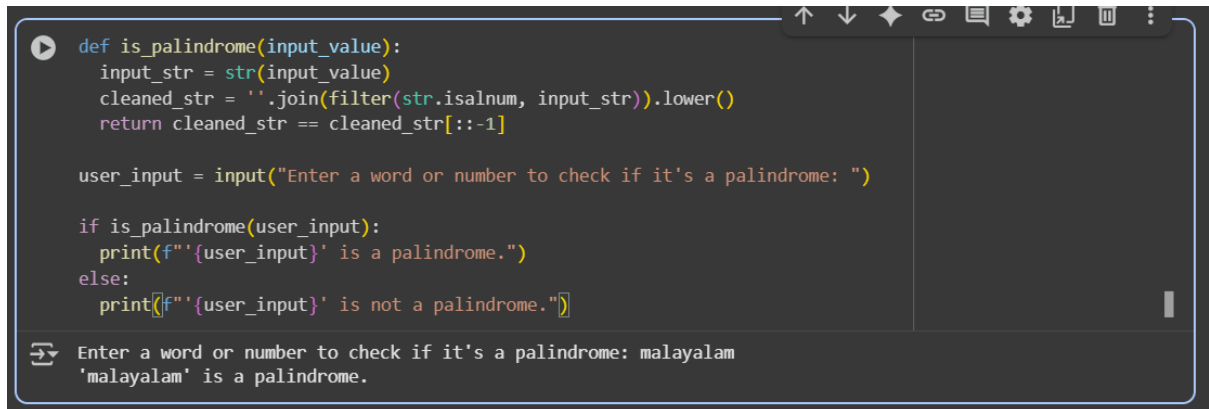
generate some test cases to
`is_number_palindrome(num)` to check whether a word/number is
palindrome or not

Here are some shortened test cases to check if a word or number is a palindrome:

- **Input:** "madam", **Expected Output:** True
- **Input:** "hello", **Expected Output:** False
- **Input:** "121", **Expected Output:** True
- **Input:** 121, **Expected Output:** True
- **Input:** -121, **Expected Output:** False (Or indicate as an error based on requirements)
- **Input:** "", **Expected Output:** True
- **Input:** "a", **Expected Output:** True
- **Input:** 1, **Expected Output:** True

USED PROMPT:

generate a python code to check whether a word/number is a palindrome or not using the above test cases mentioned above and take the input from the user

A screenshot of a Python IDE with a dark theme. The editor shows a function `is_palindrome` that takes `input_value`, converts it to a string, removes non-alphanumeric characters, and checks if the cleaned string is equal to its reverse. Below the function, there's a prompt for user input and an `if` statement that prints the result. The output console at the bottom shows the program's execution with the input 'malayalam' and the output 'malayalam' is a palindrome.

```
def is_palindrome(input_value):  
    input_str = str(input_value)  
    cleaned_str = ''.join(filter(str.isalnum, input_str)).lower()  
    return cleaned_str == cleaned_str[::-1]  
  
user_input = input("Enter a word or number to check if it's a palindrome: ")  
  
if is_palindrome(user_input):  
    print(f"{user_input} is a palindrome.")  
else:  
    print(f"{user_input} is not a palindrome.")
```

Enter a word or number to check if it's a palindrome: malayalam
'malayalam' is a palindrome.