AI ASSISTED CODING

LAB ASSIGNMENT -8.2

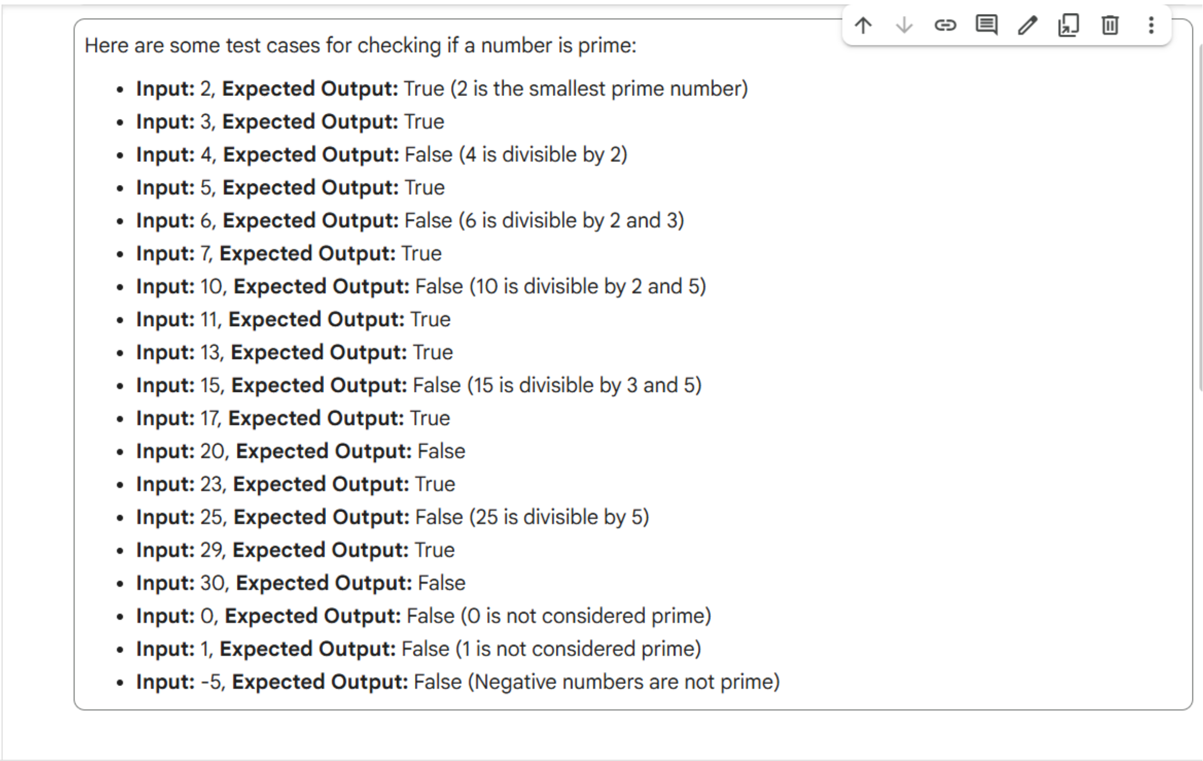
Program	: B-TECH
Specialization	: AIML
Name of Student	: AKULA MANOJ
Enrollment-No	: 2403A52031
Batch No	: 02
Date	: 23-09-2025

TASK DESCRIPTION-1

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

PROMPT FOR TESTCASES: generate some test cases only for checking whether the given number is prime or not

GENERATED TESTCASES:



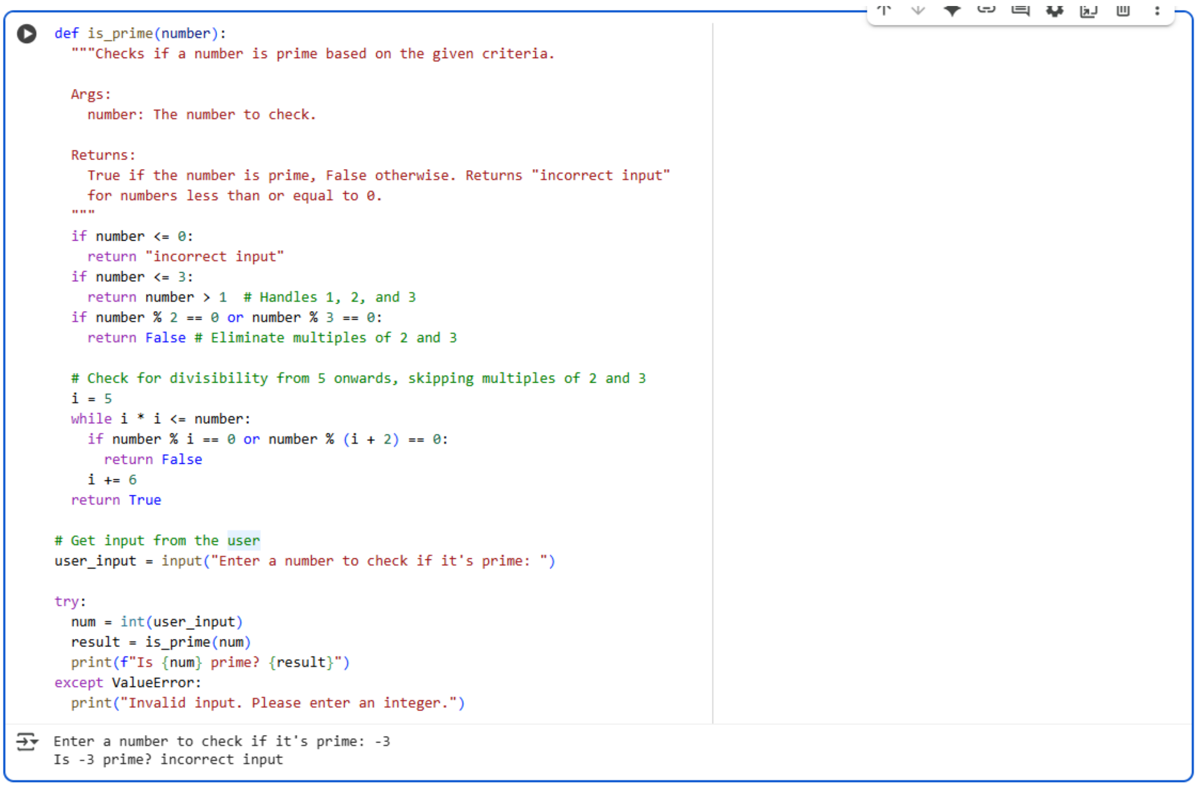
Here are some test cases for checking if a number is prime:

- **Input:** 2, **Expected Output:** True (2 is the smallest prime number)
- **Input:** 3, **Expected Output:** True
- **Input:** 4, **Expected Output:** False (4 is divisible by 2)
- **Input:** 5, **Expected Output:** True
- **Input:** 6, **Expected Output:** False (6 is divisible by 2 and 3)
- **Input:** 7, **Expected Output:** True
- **Input:** 10, **Expected Output:** False (10 is divisible by 2 and 5)
- **Input:** 11, **Expected Output:** True
- **Input:** 13, **Expected Output:** True
- **Input:** 15, **Expected Output:** False (15 is divisible by 3 and 5)
- **Input:** 17, **Expected Output:** True
- **Input:** 20, **Expected Output:** False
- **Input:** 23, **Expected Output:** True
- **Input:** 25, **Expected Output:** False (25 is divisible by 5)
- **Input:** 29, **Expected Output:** True
- **Input:** 30, **Expected Output:** False
- **Input:** 0, **Expected Output:** False (0 is not considered prime)
- **Input:** 1, **Expected Output:** False (1 is not considered prime)
- **Input:** -5, **Expected Output:** False (Negative numbers are not prime)

PROMPT FOR CODE:-

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.

CODE:-



```python
def is_prime(number):
  """Checks if a number is prime based on the given criteria.

  Args:
    number: The number to check.

  Returns:
    True if the number is prime, False otherwise. Returns "incorrect input"
    for numbers less than or equal to 0.
  """
  if number <= 0:
    return "incorrect input"
  if number <= 3:
    return number > 1  # Handles 1, 2, and 3
  if number % 2 == 0 or number % 3 == 0:
    return False # Eliminate multiples of 2 and 3

  # Check for divisibility from 5 onwards, skipping multiples of 2 and 3
  i = 5
  while i * i <= number:
    if number % i == 0 or number % (i + 2) == 0:
      return False
    i += 6
  return True

# Get input from the user
user_input = input("Enter a number to check if it's prime: ")

try:
  num = int(user_input)
  result = is_prime(num)
  print(f"Is {num} prime? {result}")
except ValueError:
  print("Invalid input. Please enter an integer.")
```

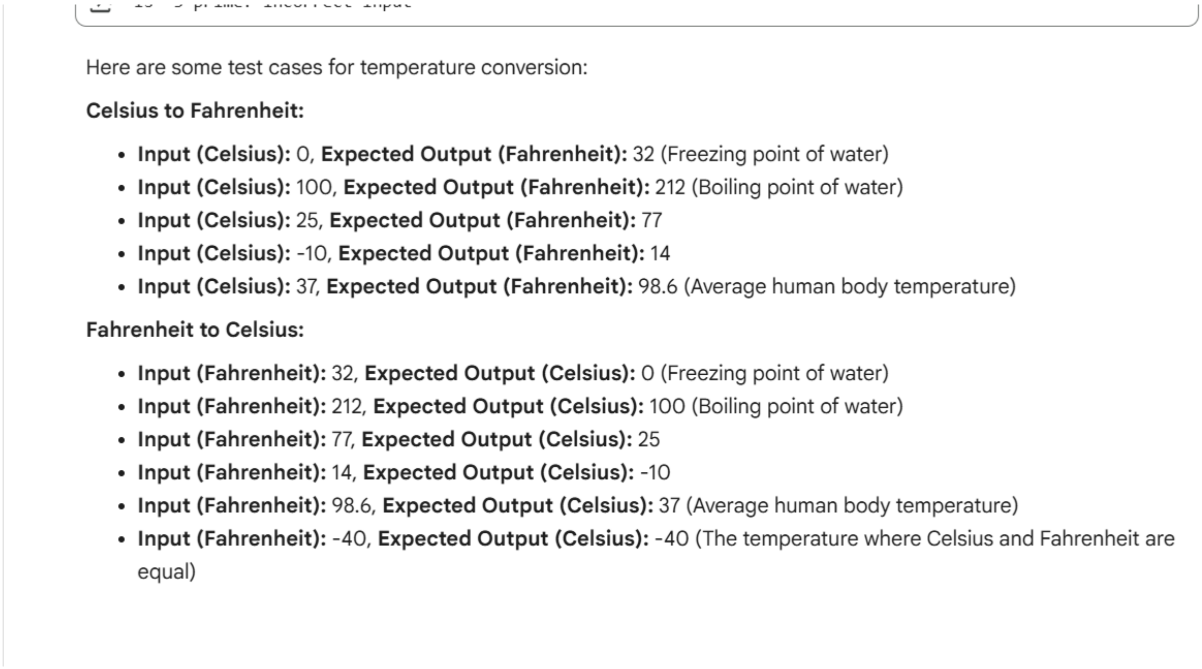
```
Enter a number to check if it's prime: -3
Is -3 prime? incorrect input
```

TASK DESCRIPTION-2:-

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

PROMPT FOR TEST CASES:- generate me some test cases only to convert temperature from celcius to fahrenheit and fahrenheit to celcius.

GENERATED TESTCASES:



Here are some test cases for temperature conversion:

Celsius to Fahrenheit:

- **Input (Celsius):** 0, **Expected Output (Fahrenheit):** 32 (Freezing point of water)
- **Input (Celsius):** 100, **Expected Output (Fahrenheit):** 212 (Boiling point of water)
- **Input (Celsius):** 25, **Expected Output (Fahrenheit):** 77
- **Input (Celsius):** -10, **Expected Output (Fahrenheit):** 14
- **Input (Celsius):** 37, **Expected Output (Fahrenheit):** 98.6 (Average human body temperature)

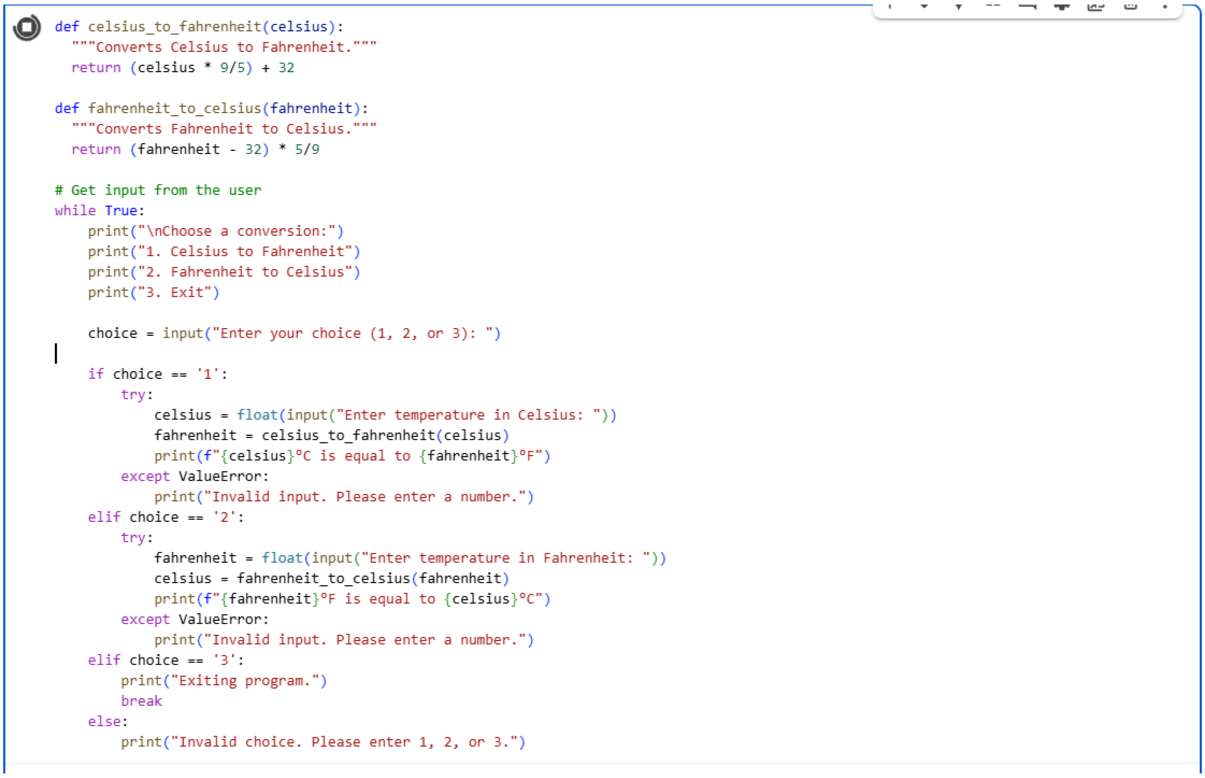
Fahrenheit to Celsius:

- **Input (Fahrenheit):** 32, **Expected Output (Celsius):** 0 (Freezing point of water)
- **Input (Fahrenheit):** 212, **Expected Output (Celsius):** 100 (Boiling point of water)
- **Input (Fahrenheit):** 77, **Expected Output (Celsius):** 25
- **Input (Fahrenheit):** 14, **Expected Output (Celsius):** -10
- **Input (Fahrenheit):** 98.6, **Expected Output (Celsius):** 37 (Average human body temperature)
- **Input (Fahrenheit):** -40, **Expected Output (Celsius):** -40 (The temperature where Celsius and Fahrenheit are equal)

PROMPT FOR CODE:-

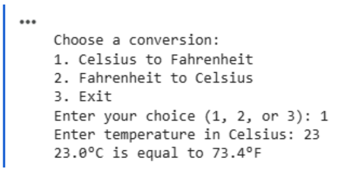
Generate a Python code that converts Celsius to Fahrenheit and vice versa. The input should be taken from the user

GENERATED CODE:-



```python
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

# Get input from the user
while True:
    print("\nChoose a conversion:")
    print("1. Celsius to Fahrenheit")
    print("2. Fahrenheit to Celsius")
    print("3. Exit")

    choice = input("Enter your choice (1, 2, or 3): ")

    if choice == '1':
        try:
            celsius = float(input("Enter temperature in Celsius: "))
            fahrenheit = celsius_to_fahrenheit(celsius)
            print(f"{celsius}°C is equal to {fahrenheit}°F")
        except ValueError:
            print("Invalid input. Please enter a number.")
    elif choice == '2':
        try:
            fahrenheit = float(input("Enter temperature in Fahrenheit: "))
            celsius = fahrenheit_to_celsius(fahrenheit)
            print(f"{fahrenheit}°F is equal to {celsius}°C")
        except ValueError:
            print("Invalid input. Please enter a number.")
    elif choice == '3':
        print("Exiting program.")
        break
    else:
        print("Invalid choice. Please enter 1, 2, or 3.")
```

OUTPUT:-



```
Choose a conversion:
1. Celsius to Fahrenheit
2. Fahrenheit to Celsius
3. Exit
Enter your choice (1, 2, or 3): 1
Enter temperature in Celsius: 23
23.0°C is equal to 73.4°F
```

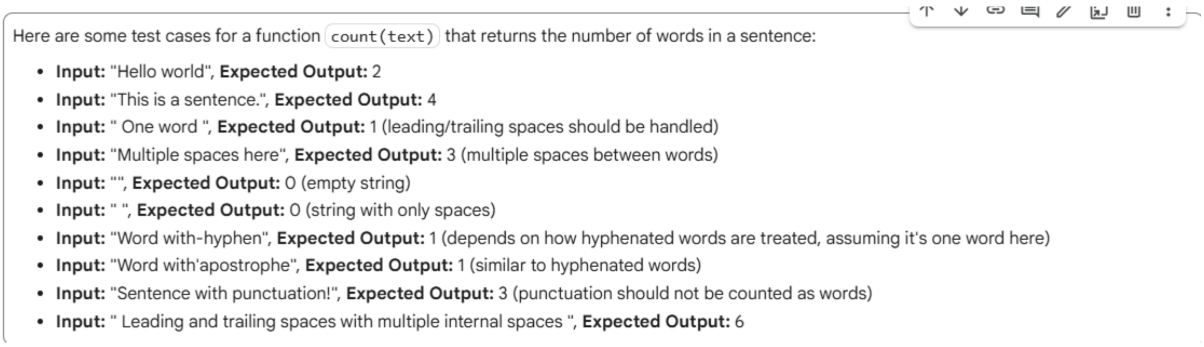
TASK DESCRIPTION-3:-

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

PROMPT FOR TEST CASES:-

generate some test cases for a function count(text) that returns the number of words in a sentence.

GENERATED TESTCASES:



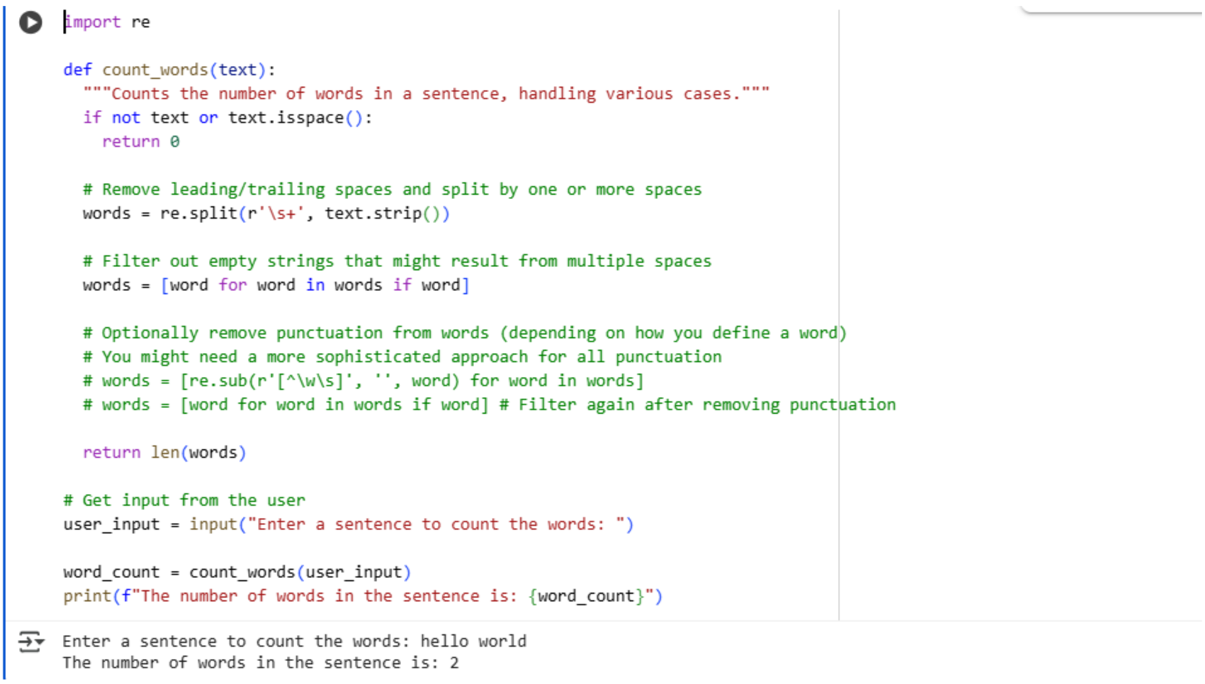
Here are some test cases for a function `count(text)` that returns the number of words in a sentence:

- **Input:** "Hello world", **Expected Output:** 2
- **Input:** "This is a sentence.", **Expected Output:** 4
- **Input:** " One word ", **Expected Output:** 1 (leading/trailing spaces should be handled)
- **Input:** "Multiple spaces here", **Expected Output:** 3 (multiple spaces between words)
- **Input:** "", **Expected Output:** 0 (empty string)
- **Input:** " ", **Expected Output:** 0 (string with only spaces)
- **Input:** "Word with-hyphen", **Expected Output:** 1 (depends on how hyphenated words are treated, assuming it's one word here)
- **Input:** "Word with'apostrophe", **Expected Output:** 1 (similar to hyphenated words)
- **Input:** "Sentence with punctuation!", **Expected Output:** 3 (punctuation should not be counted as words)
- **Input:** " Leading and trailing spaces with multiple internal spaces ", **Expected Output:** 6

PROMPT FOR CODE:

by using the above test cases, generate a Python code that shows number of words in the given input given by user.

GENERATED CODE:-



```python
import re

def count_words(text):
  """Counts the number of words in a sentence, handling various cases."""
  if not text or text.isspace():
    return 0

  # Remove leading/trailing spaces and split by one or more spaces
  words = re.split(r'\s+', text.strip())

  # Filter out empty strings that might result from multiple spaces
  words = [word for word in words if word]

  # Optionally remove punctuation from words (depending on how you define a word)
  # You might need a more sophisticated approach for all punctuation
  # words = [re.sub(r'[^\w\s]', '', word) for word in words]
  # words = [word for word in words if word] # Filter again after removing punctuation

  return len(words)

# Get input from the user
user_input = input("Enter a sentence to count the words: ")

word_count = count_words(user_input)
print(f"The number of words in the sentence is: {word_count}")
```

```
Enter a sentence to count the words: hello world
The number of words in the sentence is: 2
```

TASK DESCRIPTION-4:-

Generate test cases for a BankAccount class with:

Methods:

deposit(amount)

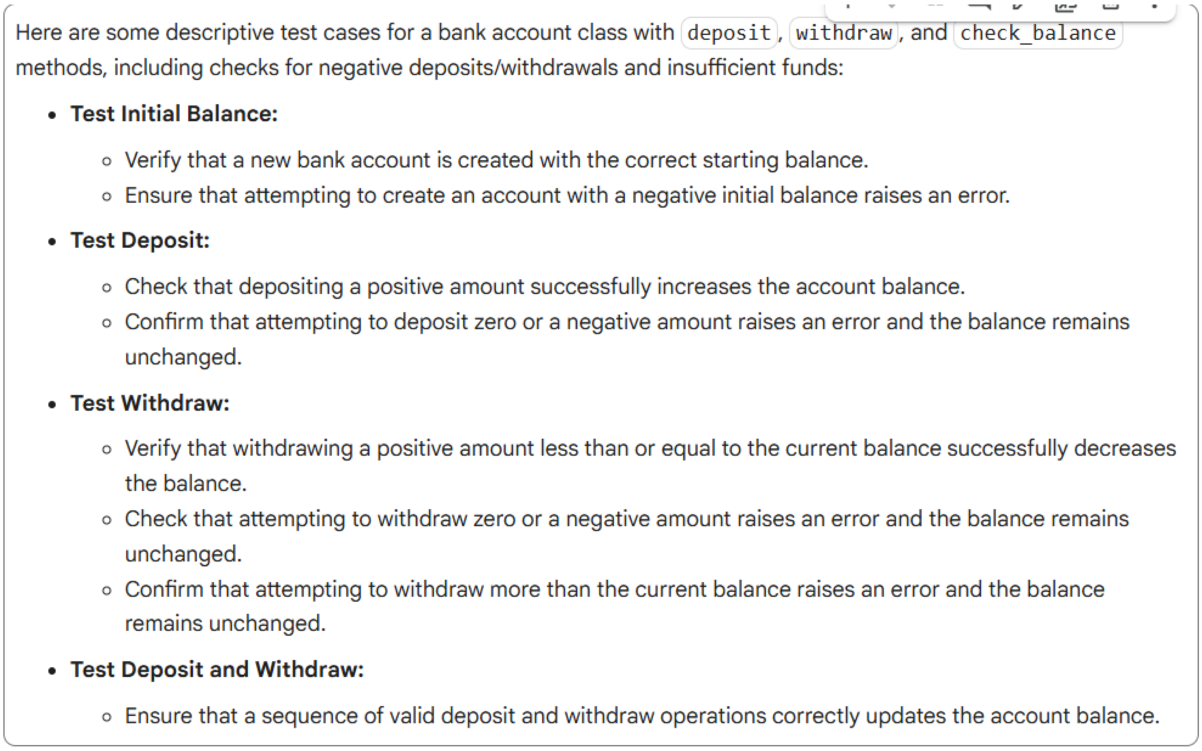
withdraw(amount)

check_balance().

PROMPT FOR TEST CASES:-

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

GENERATED TESTCASES:



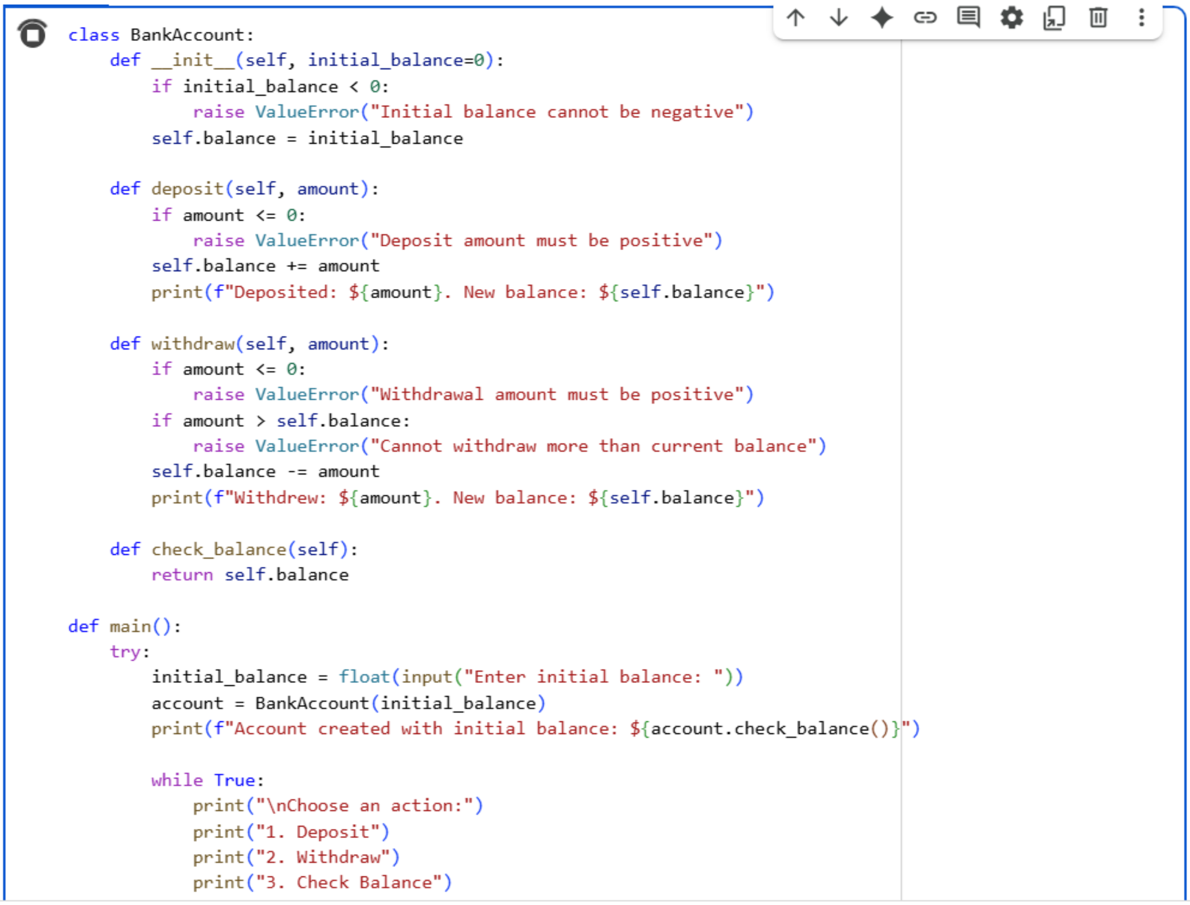
Here are some descriptive test cases for a bank account class with `deposit`, `withdraw`, and `check_balance` methods, including checks for negative deposits/withdrawals and insufficient funds:

- **Test Initial Balance:**
    - Verify that a new bank account is created with the correct starting balance.
    - Ensure that attempting to create an account with a negative initial balance raises an error.
- **Test Deposit:**
    - Check that depositing a positive amount successfully increases the account balance.
    - Confirm that attempting to deposit zero or a negative amount raises an error and the balance remains unchanged.
- **Test Withdraw:**
    - Verify that withdrawing a positive amount less than or equal to the current balance successfully decreases the balance.
    - Check that attempting to withdraw zero or a negative amount raises an error and the balance remains unchanged.
    - Confirm that attempting to withdraw more than the current balance raises an error and the balance remains unchanged.
- **Test Deposit and Withdraw:**
    - Ensure that a sequence of valid deposit and withdraw operations correctly updates the account balance.

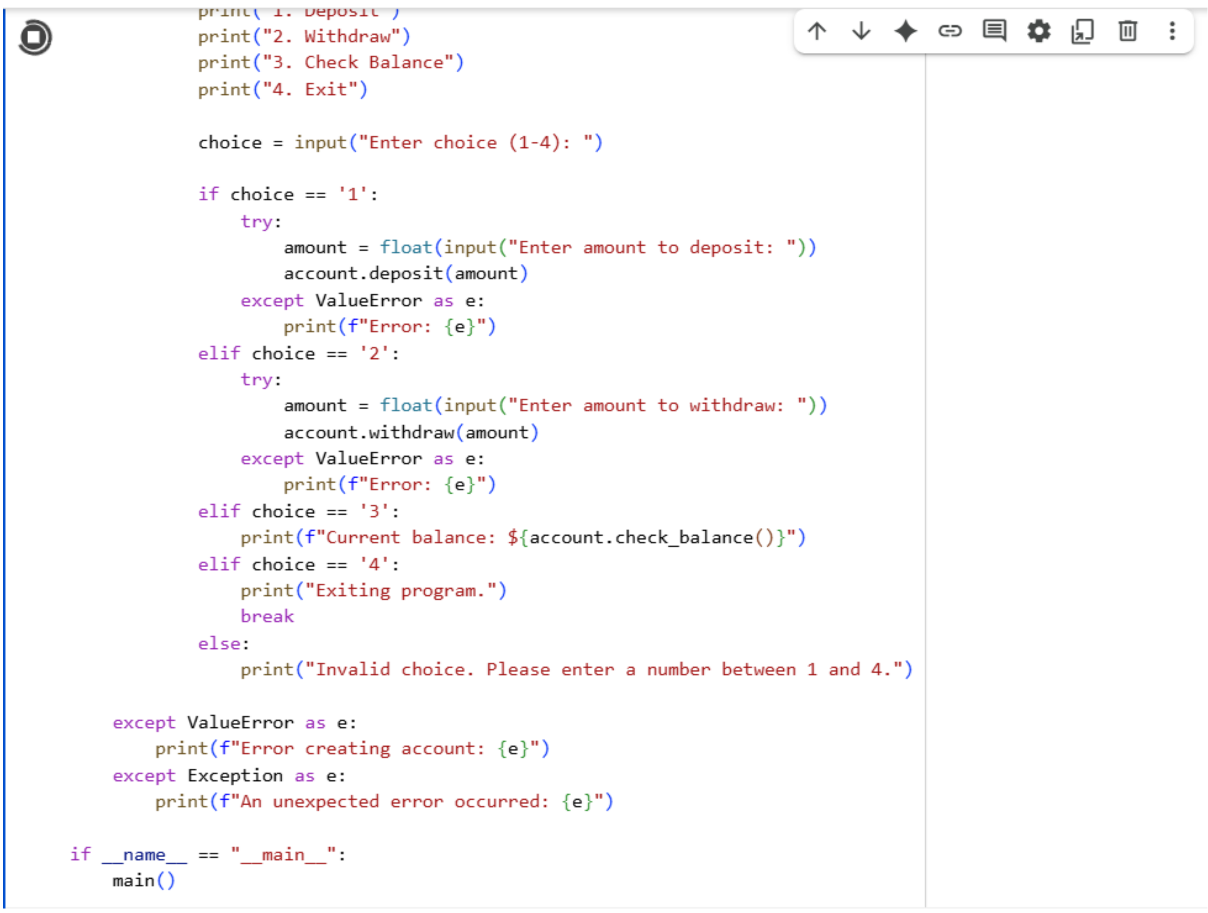
PROMPT FOR CODE:

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

GENERATED CODE:-

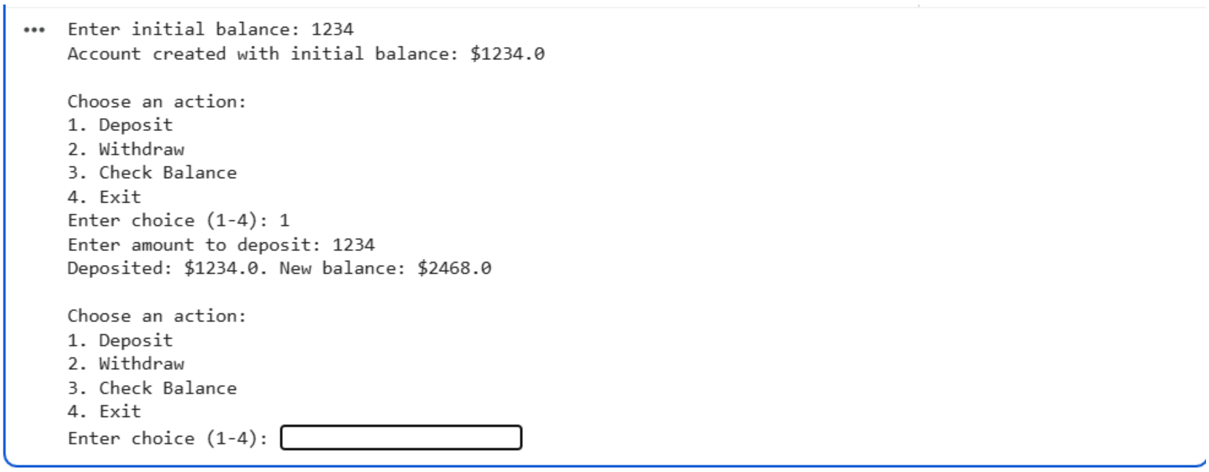


```python
class BankAccount:
    def __init__(self, initial_balance=0):
        if initial_balance < 0:
            raise ValueError("Initial balance cannot be negative")
        self.balance = initial_balance

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

    def withdraw(self, amount):
        if amount <= 0:
            raise ValueError("Withdrawal amount must be positive")
        if amount > self.balance:
            raise ValueError("Cannot withdraw more than current balance")
        self.balance -= amount
        print(f"Withdrew: ${amount}. New balance: ${self.balance}")

    def check_balance(self):
        return self.balance

def main():
    try:
        initial_balance = float(input("Enter initial balance: "))
        account = BankAccount(initial_balance)
        print(f"Account created with initial balance: ${account.check_balance()}")

        while True:
            print("\nChoose an action:")
            print("1. Deposit")
            print("2. Withdraw")
            print("3. Check Balance")
```



```python
            print("1. Deposit")
            print("2. Withdraw")
            print("3. Check Balance")
            print("4. Exit")

            choice = input("Enter choice (1-4): ")

            if choice == '1':
                try:
                    amount = float(input("Enter amount to deposit: "))
                    account.deposit(amount)
                except ValueError as e:
                    print(f"Error: {e}")
            elif choice == '2':
                try:
                    amount = float(input("Enter amount to withdraw: "))
                    account.withdraw(amount)
                except ValueError as e:
                    print(f"Error: {e}")
            elif choice == '3':
                print(f"Current balance: ${account.check_balance()}")
            elif choice == '4':
                print("Exiting program.")
                break
            else:
                print("Invalid choice. Please enter a number between 1 and 4.")

    except ValueError as e:
        print(f"Error creating account: {e}")
    except Exception as e:
        print(f"An unexpected error occurred: {e}")

if __name__ == "__main__":
    main()
```

OUTPUT:-



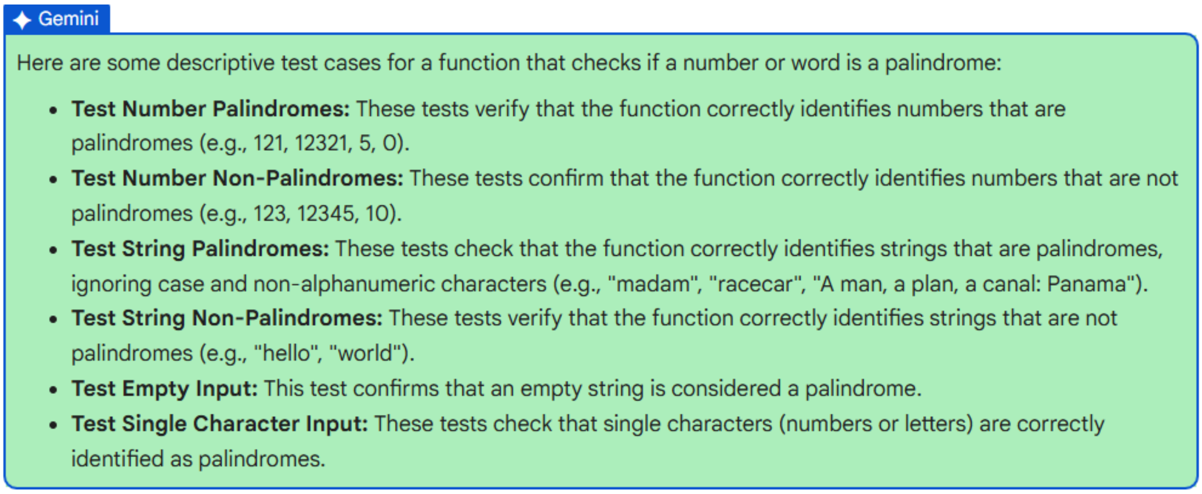
```
•••  Enter initial balance: 1234
     Account created with initial balance: $1234.0

     Choose an action:
     1. Deposit
     2. Withdraw
     3. Check Balance
     4. Exit
     Enter choice (1-4): 1
     Enter amount to deposit: 1234
     Deposited: $1234.0. New balance: $2468.0

     Choose an action:
     1. Deposit
     2. Withdraw
     3. Check Balance
     4. Exit
     Enter choice (1-4):
```

TASK DESCRIPTION-5:-

PROMPT FOR TEST CASES:-

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

GENERATED TESTCASES:



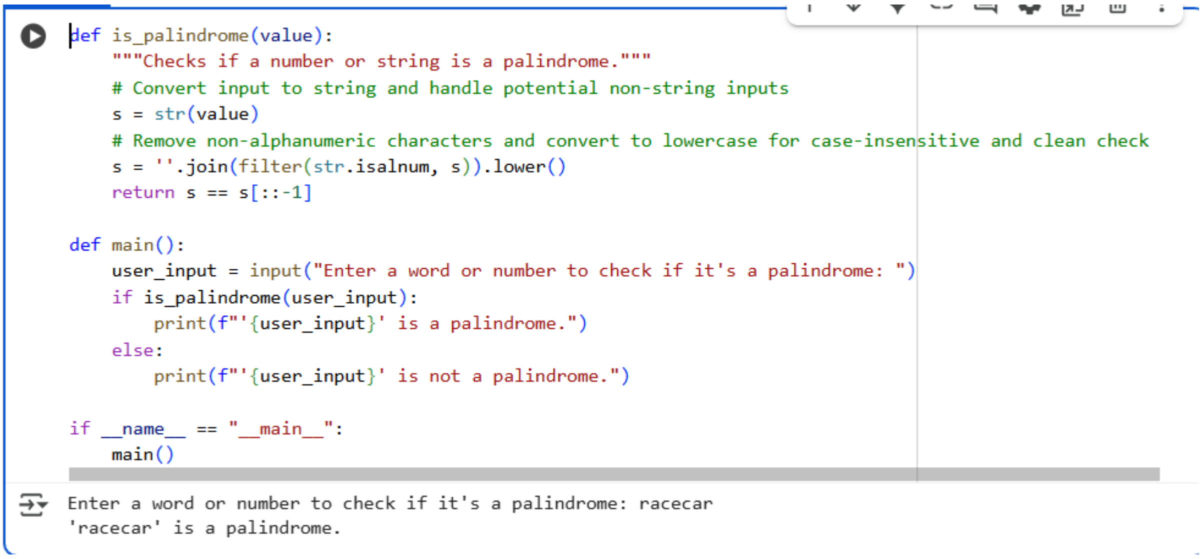
Here are some descriptive test cases for a function that checks if a number or word is a palindrome:

- **Test Number Palindromes:** These tests verify that the function correctly identifies numbers that are palindromes (e.g., 121, 12321, 5, 0).
- **Test Number Non-Palindromes:** These tests confirm that the function correctly identifies numbers that are not palindromes (e.g., 123, 12345, 10).
- **Test String Palindromes:** These tests check that the function correctly identifies strings that are palindromes, ignoring case and non-alphanumeric characters (e.g., "madam", "racecar", "A man, a plan, a canal: Panama").
- **Test String Non-Palindromes:** These tests verify that the function correctly identifies strings that are not palindromes (e.g., "hello", "world").
- **Test Empty Input:** This test confirms that an empty string is considered a palindrome.
- **Test Single Character Input:** These tests check that single characters (numbers or letters) are correctly identified as palindromes.

PROMPT FOR CODE:

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

GENERATED CODE:-



```python
def is_palindrome(value):
    """Checks if a number or string is a palindrome."""
    # Convert input to string and handle potential non-string inputs
    s = str(value)
    # Remove non-alphanumeric characters and convert to lowercase for case-insensitive and clean check
    s = ''.join(filter(str.isalnum, s)).lower()
    return s == s[::-1]

def main():
    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.")

if __name__ == "__main__":
    main()
```

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

THANK YOU