AI Assistant Coding

Assignment-6.3

Name:V.Sruthi

Ht.no:2303A51308

Batch:05

Task-01:

Classes (Student Class)

You are developing a simple student information management module.

Prompt:

Code:

```python
class Student:
    def __init__(self, name, roll_number, branch):
        self.name = name
        self.roll_number = roll_number
        self.branch = branch

    def display_details(self):
        print(f"Student Name: {self.name}")
        print(f"Roll Number: {self.roll_number}")
        print(f"Branch: {self.branch}")

# Create a sample student object
student1 = Student("Alice Smith", "CS101", "Computer Science")

# Display the student's details
student1.display_details()
```

Output:

```
Student Name: Alice Smith
Roll Number: CS101
Branch: Computer Science
```

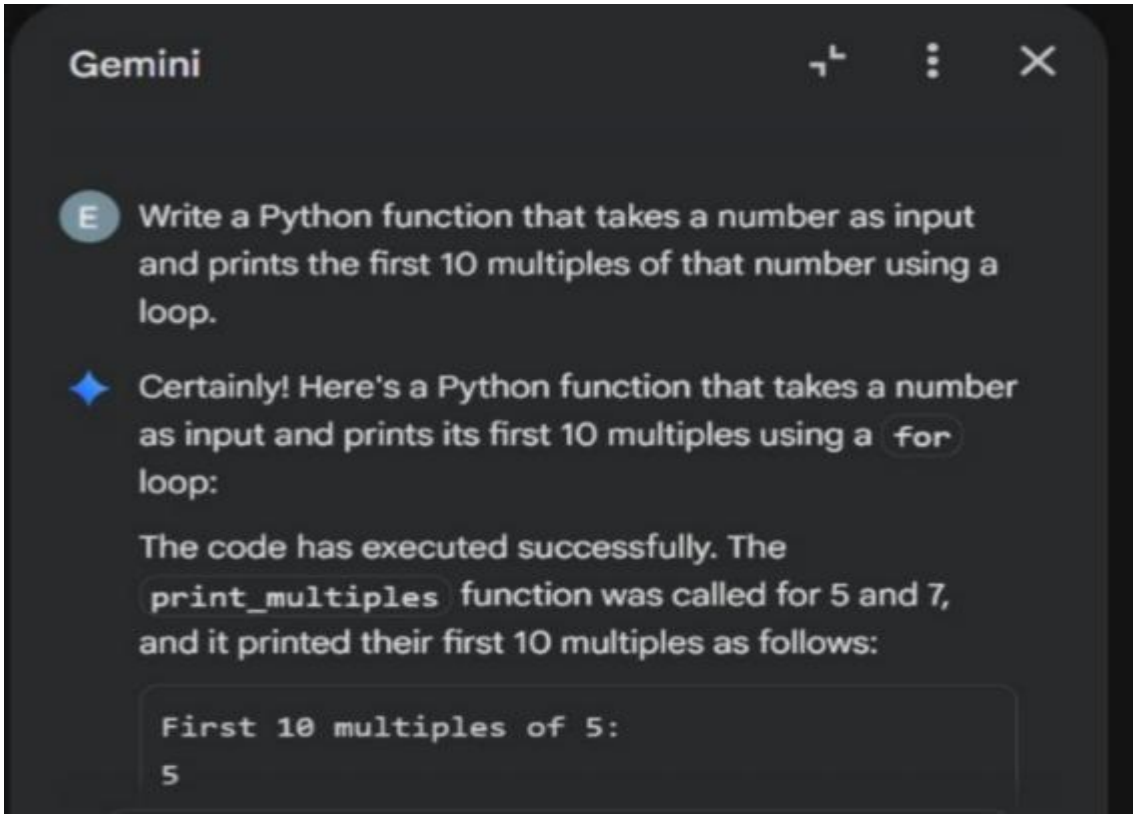
Explanation:

- A class named Student is created to represent student information.
- The __init__() constructor is defined and is called automatically when a Student object is created.
- The constructor initializes the student’s name, roll number, and branch.
- A method display_details() is defined to print the student details.
- A Student object is created by passing values for name, roll number, and branch.
- The display_details() method is called using the object.
- The student information is displayed on the console.

Task-02:

Loops (Multiples of a Number)
Scenario
You are writing a utility function to display multiples of a given number.

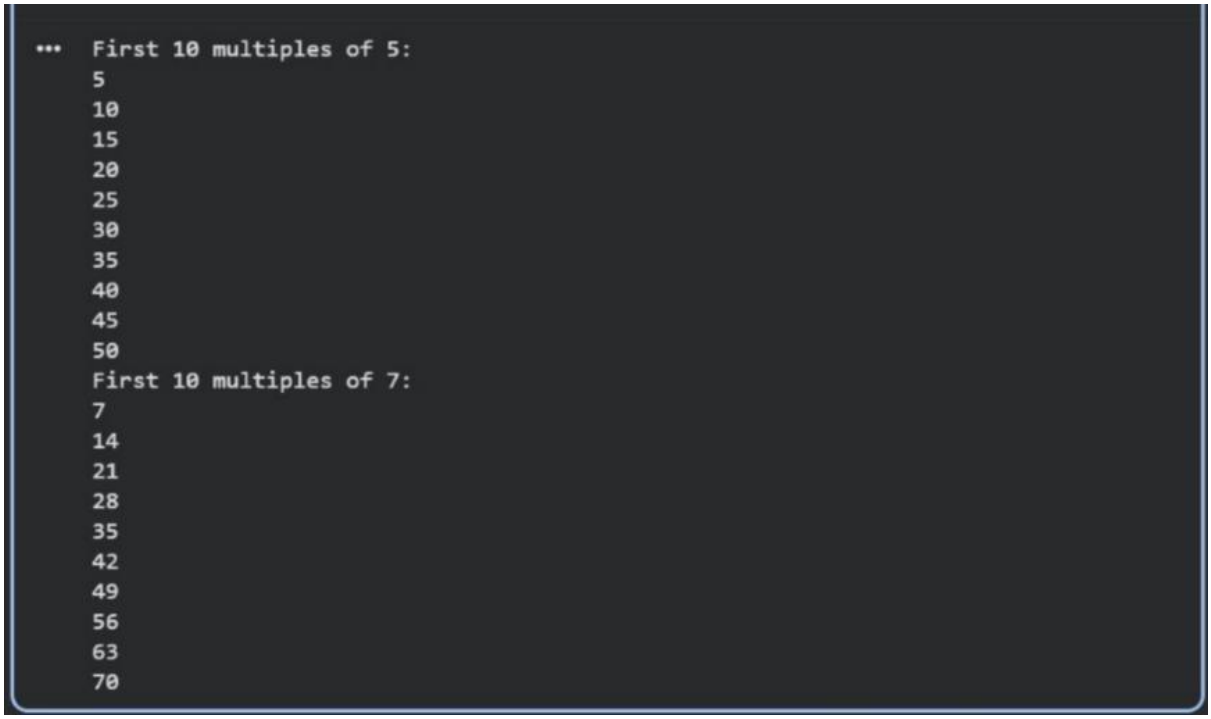
Prompt:



Code:

```
def print_multiples(number):
    print(f"First 10 multiples of {number}:")
    for i in range(1, 11):  # Loop from 1 to 10 (inclusive)
        multiple = number * i
        print(multiple)

# Example usage:
print_multiples(5)
print_multiples(7)
```

Output:



Explanation:

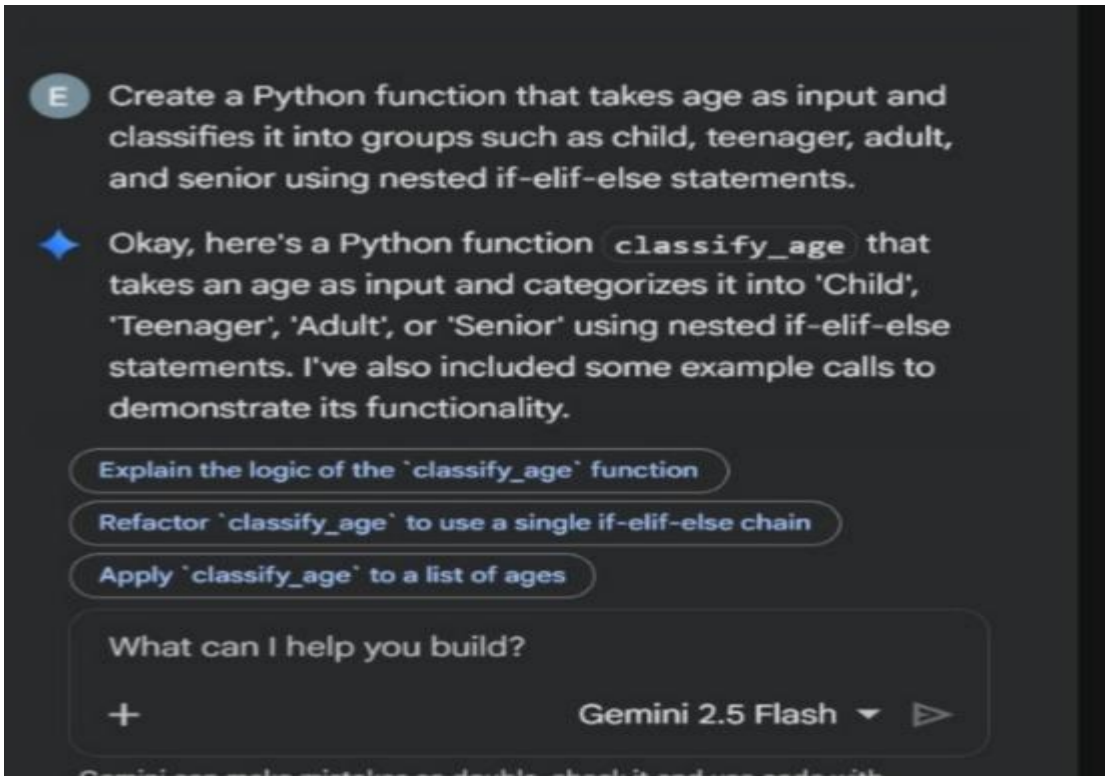
- A function is defined that accepts a number as input.
- A loop is used to repeat the process 10 times.
- In each iteration, the input number is multiplied by the loop counter.
- The result of the multiplication gives one multiple of the number.
- Each multiple is printed on the console.
- The loop continues until the first 10 multiples are displayed.
- When the loop ends, the program stops execution.

Task-03:

Conditional Statements (Age Classification)
Scenario

You are building a basic classification system based on age.

Prompt:



Code:

```python
def classify_age(age):
    if age >= 0:
        if age <= 12:
            return "Child"
        elif age <= 19:
            return "Teenager"
        elif age <= 64:
            return "Adult"
        else:
            return "Senior"
    else:
        return "Invalid Age (Age cannot be negative)"

# Test cases
print(f"Age 5 is: {classify_age(5)}")
print(f"Age 15 is: {classify_age(15)}")
print(f"Age 30 is: {classify_age(30)}")
print(f"Age 70 is: {classify_age(70)}")
print(f"Age -2 is: {classify_age(-2)}")
```

Output:

```
Age 5 is: Child
Age 15 is: Teenager
Age 30 is: Adult
Age 70 is: Senior
Age -2 is: Invalid Age (Age cannot be negative)
```

Explanation:

A function is defined that accepts age as an input value.

- The program checks the age using if-elif-else conditions.
- If the age is less than a certain value, it is classified as a child.
- If the age falls in the next range, it is classified as a teenager.
- If the age is in the adult range, it is classified as an adult.
- If the age is above the adult range, it is classified as a senior.

- Only one condition is executed because once a condition is true, the remaining checks are skipped.
- The function returns or prints the appropriate age group.

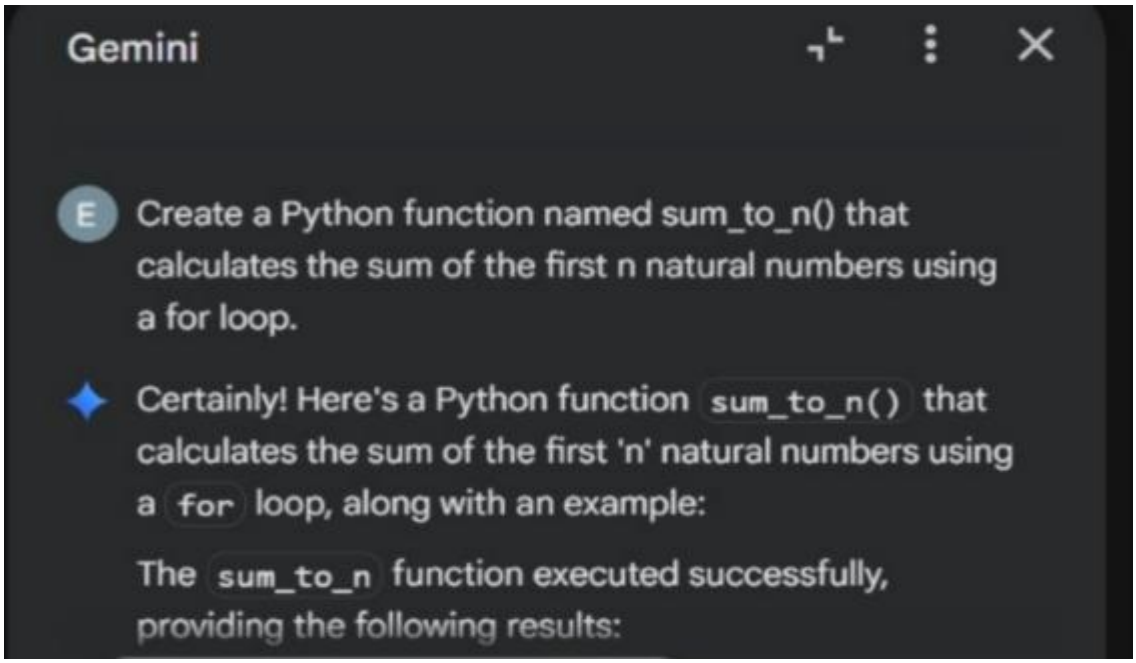
Task-04:

For and While Loops (Sum of First n Numbers)

Scenario
You need to calculate the sum of the first n natural numbers.

a)Using For Loop:

Prompt:



Code:

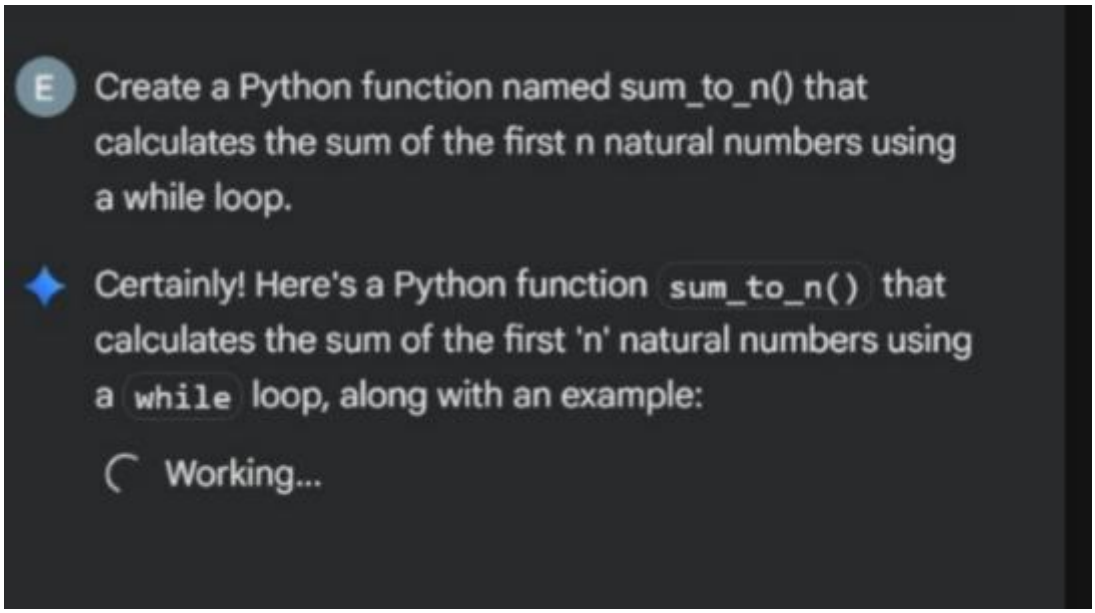
```python
def sum_to_n(n):
    if n < 0:
        return "Input must be a non-negative integer."

    total_sum = 0
    for i in range(1, n + 1):
        total_sum += i
    return total_sum

# Example usage:
print(f"The sum of the first 5 natural numbers is: {sum_to_n(5)}")
print(f"The sum of the first 10 natural numbers is: {sum_to_n(10)}")
print(f"The sum of the first 0 natural numbers is: {sum_to_n(0)}")
print(f"Trying with a negative number: {sum_to_n(-3)}")
```

Output:

```
The sum of the first 5 natural numbers is: 15
The sum of the first 10 natural numbers is: 55
The sum of the first 0 natural numbers is: 0
Trying with a negative number: Input must be a non-negative integer.
```

b)using While Loop:

Prompt:



Code:

```python
def sum_to_n_while(n):
    if n < 0:
        return "Input must be a non-negative integer."

    total_sum = 0
    count = 1
    while count <= n:
        total_sum += count
        count += 1
    return total_sum

# Example usage:
print(f"The sum of the first 5 natural numbers using while loop is: {sum_to_n_whil
print(f"The sum of the first 10 natural numbers using while loop is: {sum_to_n_whi
print(f"The sum of the first 0 natural numbers using while loop is: {sum_to_n_whil
print(f"Trying with a negative number: {sum_to_n_while(-3)}")
```

Output:

```
The sum of the first 5 natural numbers using while loop is: 15
The sum of the first 10 natural numbers using while loop is: 55
The sum of the first 0 natural numbers using while loop is: 0
Trying with a negative number: Input must be a non-negative integer.
```

Explanation:

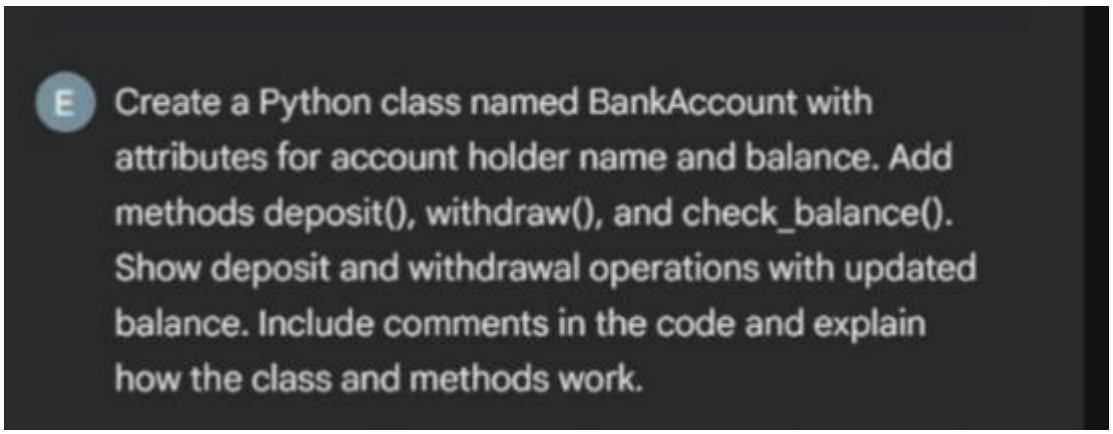
- A function named sum_to_n() is defined to calculate the sum of natural numbers.
- The function takes an integer n as input.
- A variable (for example, total) is initialized to 0 to store the sum.
- A for loop runs from 1 to n.
- In each iteration, the current number is added to total.

- After the loop finishes, total contains the sum of the first n natural numbers.
- The function returns or prints the final sum.
- The output is displayed for the given sample input.

Task-05:

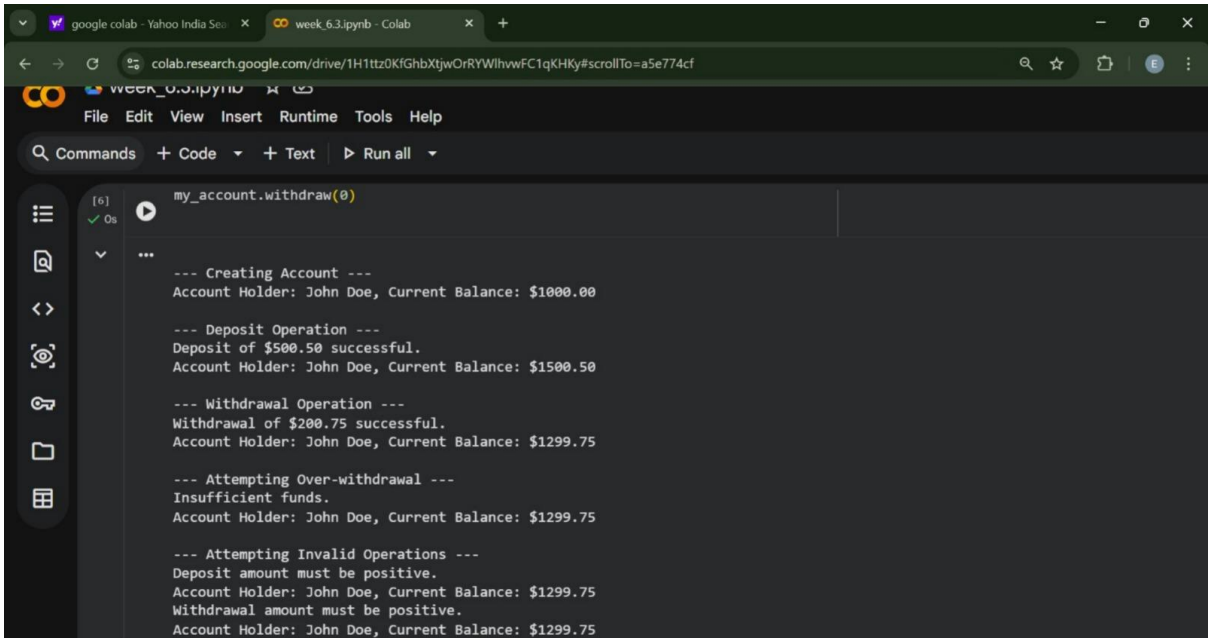
Classes (Bank Account Class)
Scenario
You are designing a basic banking application.

Prompt:



Code:

Output:



Explanation:

- A class named BankAccount is created to represent a bank account.
- The constructor (__init__) initializes the account holder’s name and the starting balance.

- The deposit() method is used to add a given amount to the current balance.
- The withdraw() method subtracts an amount from the balance after checking if sufficient funds are available.
- If the balance is not enough, the withdrawal is not allowed and a message is shown.
- The check_balance() method displays the current account balance.
- An object of the BankAccount class is created to perform banking operations.
- Deposit and withdrawal methods are called, and the updated balance is shown after each operation.