Assignment-6.3

Name: M. Preethi Meghana

Ht.no: 2303A51584

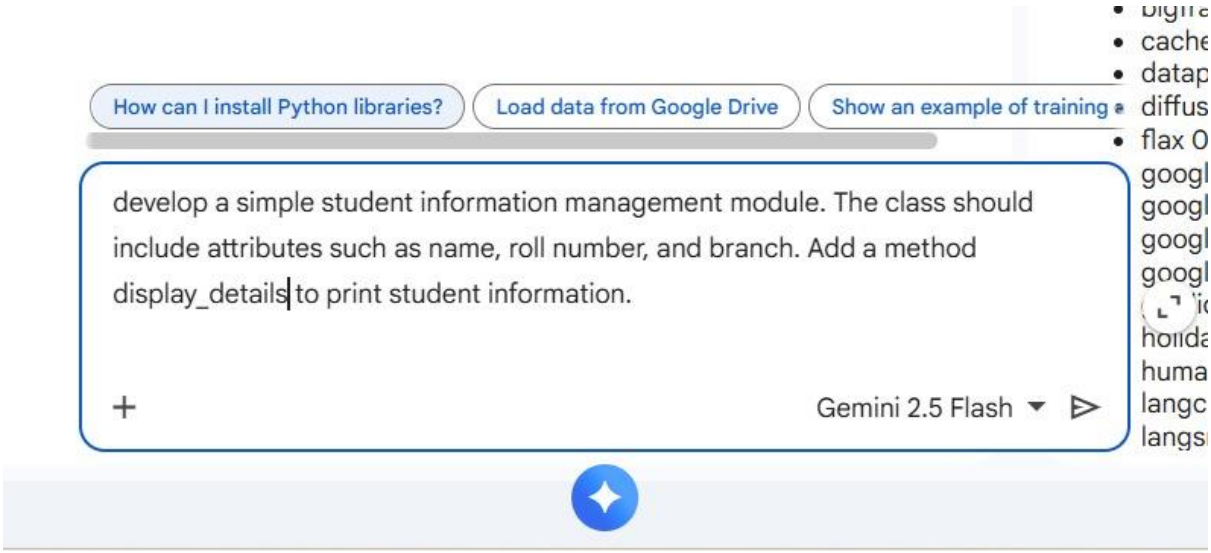
Batch: 05

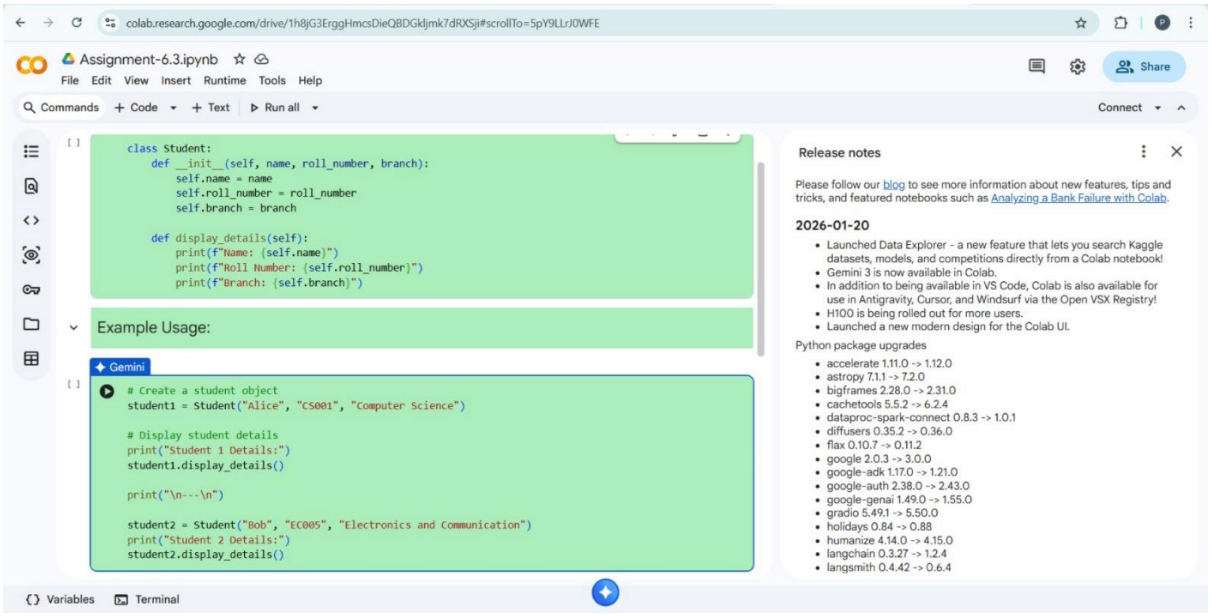
Task-01:

Classes (Student Class)

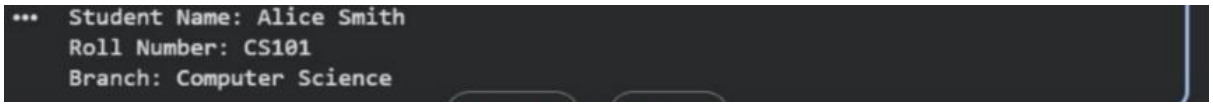
You are developing a simple student information management module.

Prompt:





Output:



Explanation:

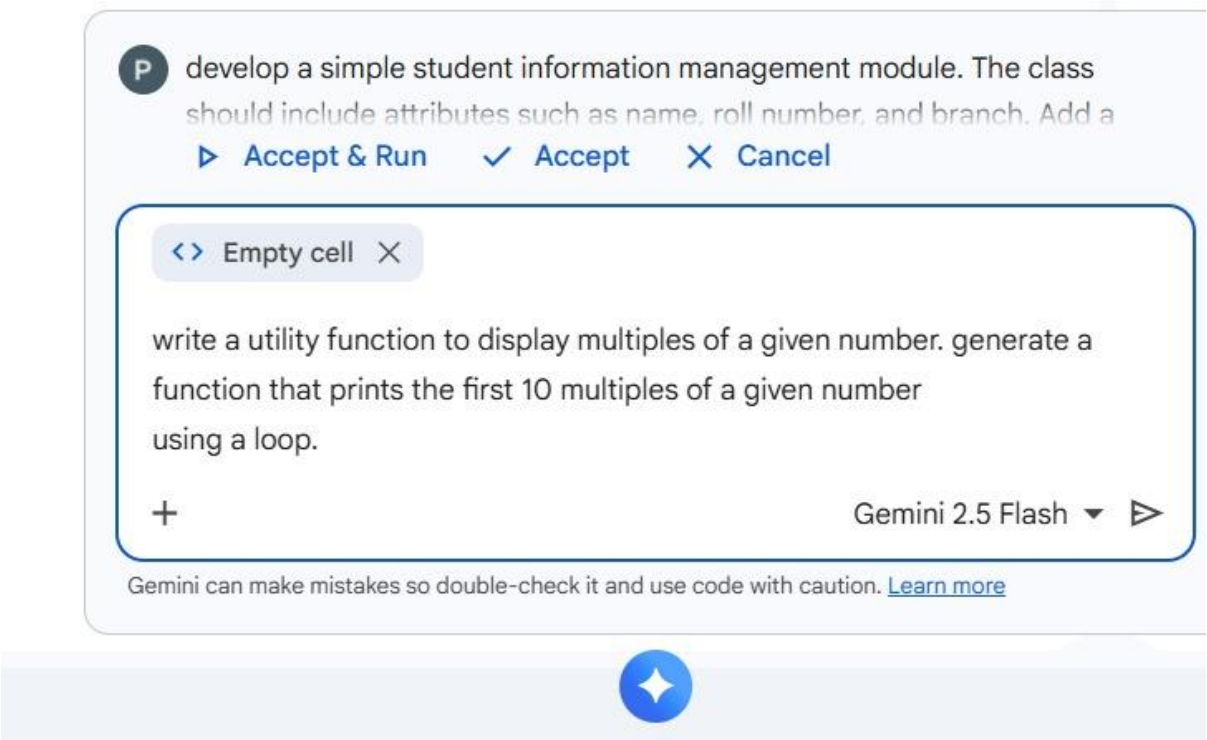
- A class named Student is created to represent student information.
- The __init__() constructor is defined and is called automatically when a Student object is created.
- The constructor initializes the student's name, roll number, and branch.
- A method display_details() is defined to print the student 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.

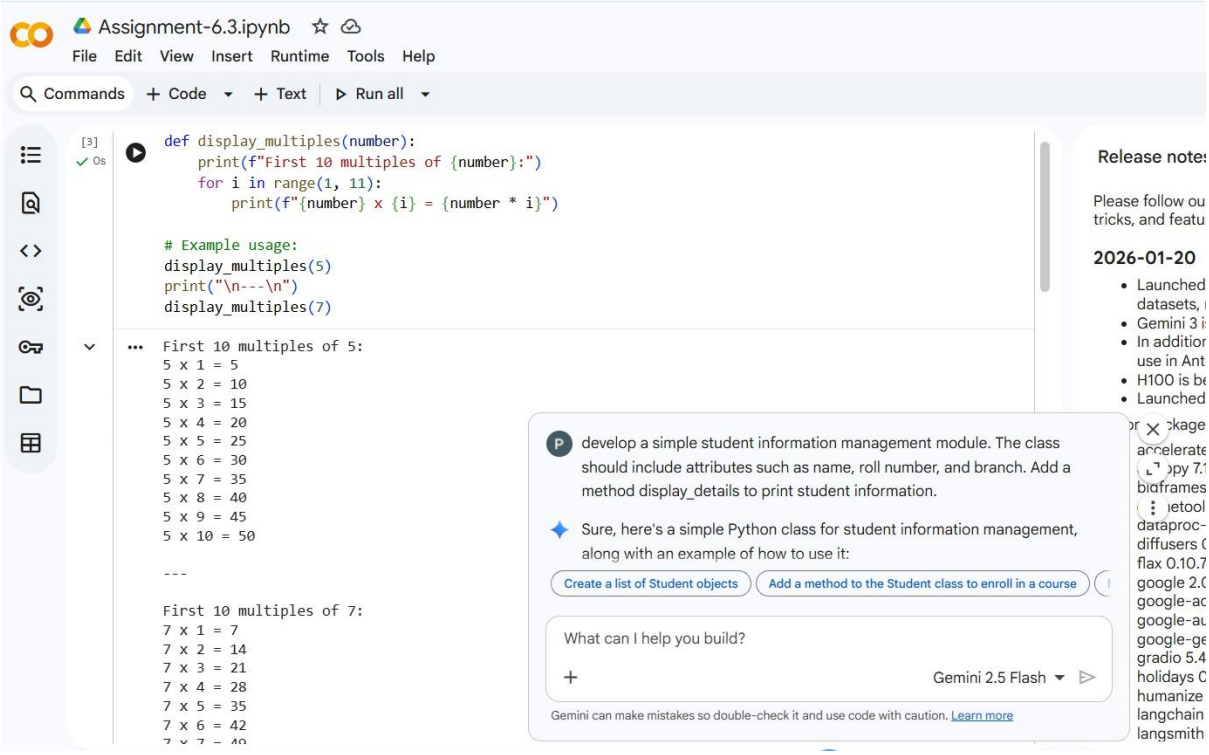
Task-02:

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

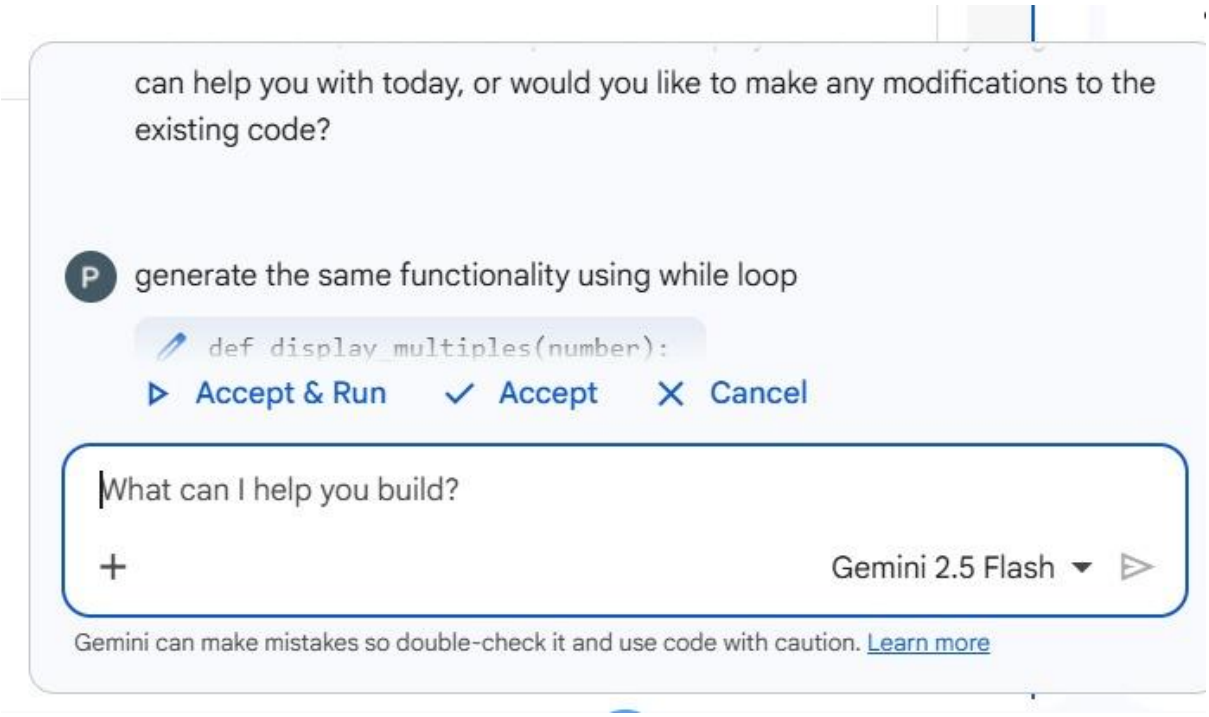
Prompt:



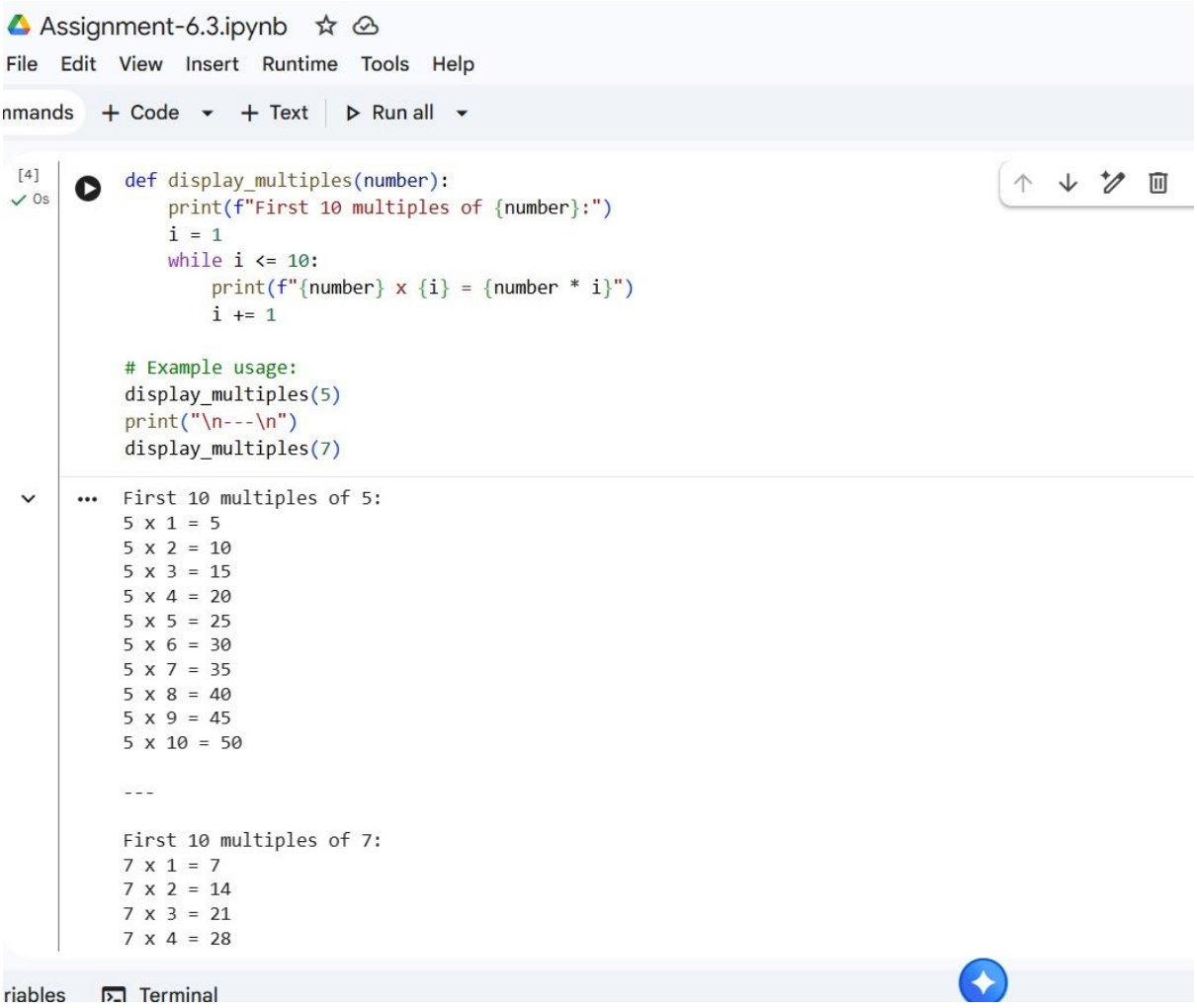
Code:


Assignment-6.3.ipynb
File Edit View Insert Runtime Tools Help
Commands + Code + Text Run all
[3] 0s
def display_multiples(number):
print(f"First 10 multiples of {number}:")
for i in range(1, 11):
print(f"{number} x {i} = {number * i}")
# Example usage:
display_multiples(5)
print("\n---\n")
display_multiples(7)
First 10 multiples of 5:
5 x 1 = 5
5 x 2 = 10
5 x 3 = 15
5 x 4 = 20
5 x 5 = 25
5 x 6 = 30
5 x 7 = 35
5 x 8 = 40
5 x 9 = 45
5 x 10 = 50

First 10 multiples of 7:
7 x 1 = 7
7 x 2 = 14
7 x 3 = 21
7 x 4 = 28
7 x 5 = 35
7 x 6 = 42
develop a simple student information management module. The class should include attributes such as name, roll number, and branch. Add a method display_details to print student information.
Sure, here's a simple Python class for student information management, along with an example of how to use it:
Create a list of Student objects
Add a method to the Student class to enroll in a course
What can I help you build?
Gemini 2.5 Flash
Gemini can make mistakes so double-check it and use code with caution. Learn more
Release notes
Please follow ou
tricks, and featu
2026-01-20
Launched
datasets,
Gemini 3
In additior
use in Ant
H100 is be
Launched


can help you with today, or would you like to make any modifications to the existing code?
generate the same functionality using while loop
def display_multiples(number):
Accept & Run
Accept
Cancel
What can I help you build?
Gemini 2.5 Flash
Gemini can make mistakes so double-check it and use code with caution. Learn more

Output:



Explanation:

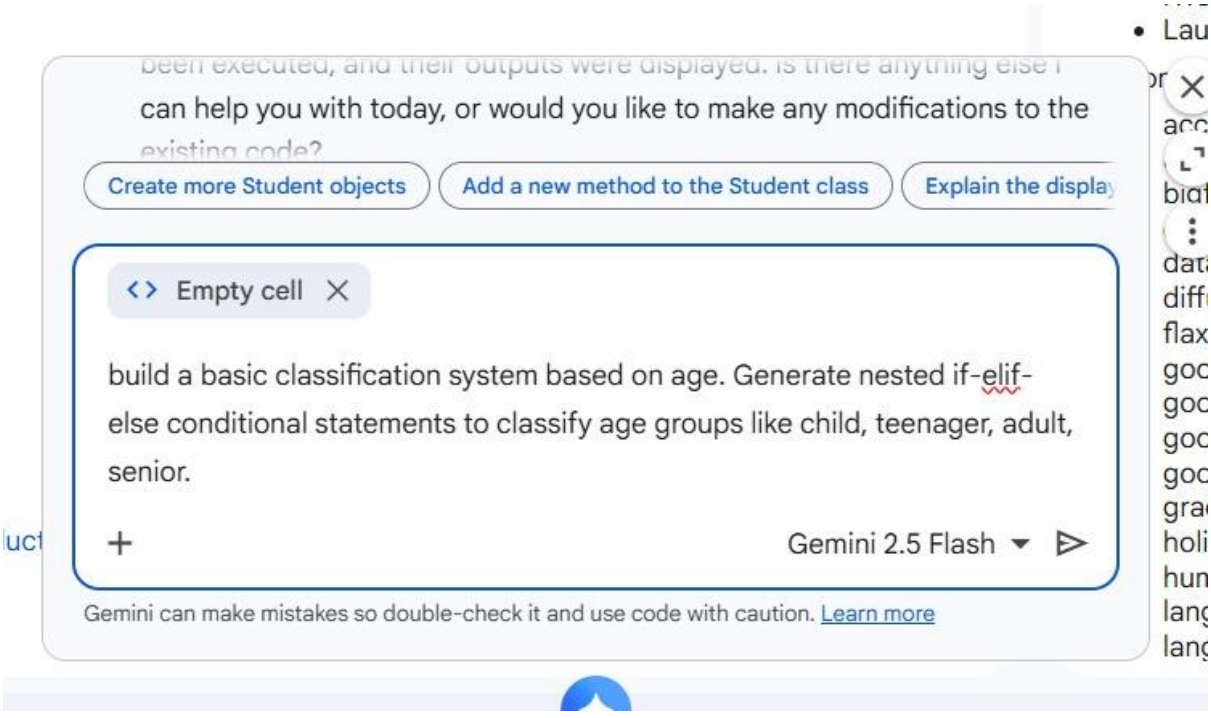
- A function is defined that accepts a number as input.
- A loop is used to repeat the process 10 times.
- In each iteration, the input number is multiplied by the loop counter.
- The result of the multiplication gives one multiple of the number.
- Each multiple is printed on the console.

- The loop continues until the first 10 multiples are displayed.
- When the loop ends, the program stops execution.

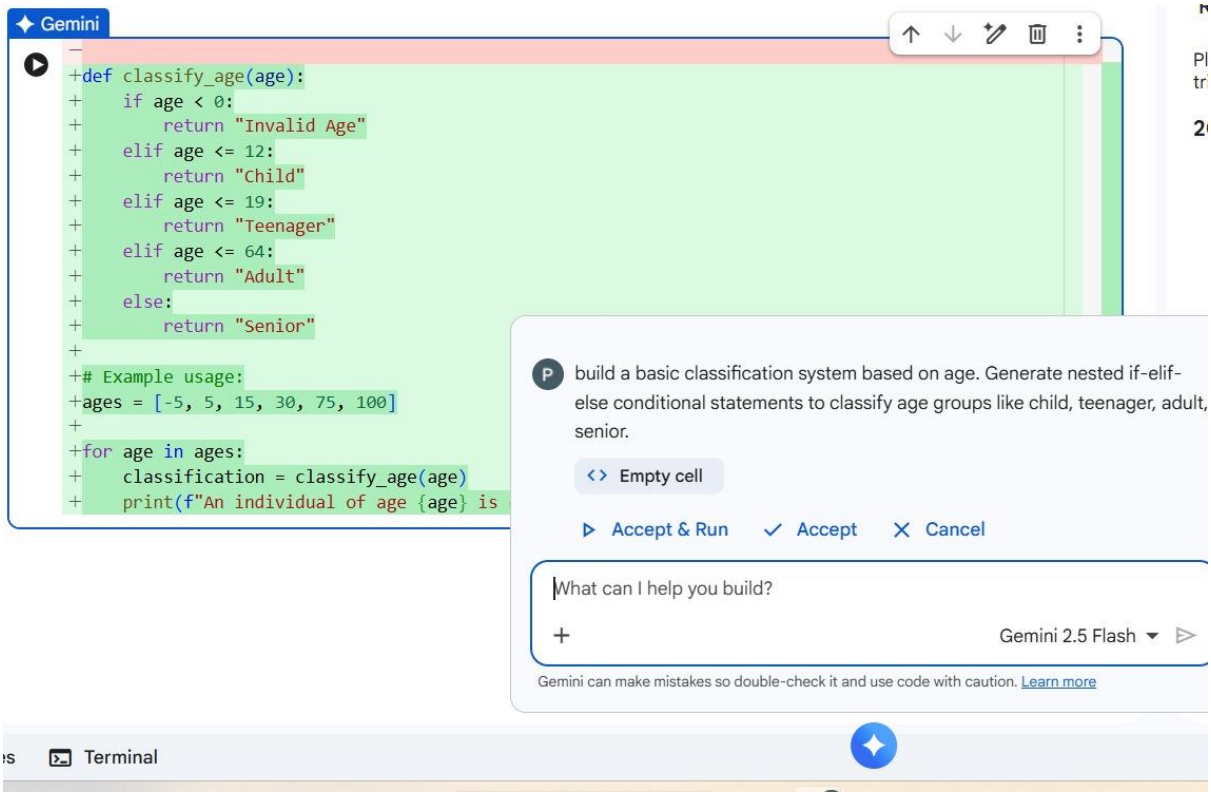
Task-03:

Conditional Statements (Age Classification)
Scenario
You are building a basic classification system based on age.

Prompt:



Code:


Gemini
+def classify_age(age):
+ if age < 0:
+ return "Invalid Age"
+ elif age <= 12:
+ return "Child"
+ elif age <= 19:
+ return "Teenager"
+ elif age <= 64:
+ return "Adult"
+ else:
+ return "Senior"
+
+# Example usage:
+ages = [-5, 5, 15, 30, 75, 100]
+
+for age in ages:
+ classification = classify_age(age)
+ print(f"An individual of age {age} is
P
build a basic classification system based on age. Generate nested if-elif-else conditional statements to classify age groups like child, teenager, adult, senior.
Empty cell
Accept & Run
Accept
Cancel
What can I help you build?
Gemini 2.5 Flash
Gemini can make mistakes so double-check it and use code with caution. Learn more
Terminal

mmands + Code + Text ▷ Run all

[6] ✓ 0s

```
def classify_age(age):
    if age < 0:
        return "Invalid Age"

    age_categories = [
        {'upper_bound': 12, 'label': 'Child'},
        {'upper_bound': 19, 'label': 'Teenager'},
        {'upper_bound': 64, 'label': 'Adult'}
    ]

    for category_info in age_categories:
        if age <= category_info['upper_bound']:
            return category_info['label']

    return "Senior" # If age is greater than all upper bounds

# Example usage:
ages = [-5, 5, 15, 30, 75, 100]

for age in ages:
    classification = classify_age(age)
    print(f"An individual of age {age} is classified as: {classification}")
```

```
An individual of age -5 is classified as: Invalid Age
An individual of age 5 is classified as: Child
An individual of age 15 is classified as: Teenager
An individual of age 30 is classified as: Adult
An individual of age 75 is classified as: Senior
An individual of age 100 is classified as: Senior
```

Output:

```
def classify_age(age):
    if age < 0:
        return "Invalid Age"
    elif age <= 12:
        return "Child"
    elif age <= 19:
        return "Teenager"
    elif age <= 64:
        return "Adult"
    else:
        return "Senior"

# Example usage:
ages = [-5, 5, 15, 30, 75, 100]

for age in ages:
    classification = classify_age(age)
    print(f"An individual of age {age} is classified as: {classification}")
```

```
An individual of age -5 is classified as: Invalid Age
An individual of age 5 is classified as: Child
An individual of age 15 is classified as: Teenager
An individual of age 30 is classified as: Adult
An individual of age 75 is classified as: Senior
An individual of age 100 is classified as: Senior
```

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.

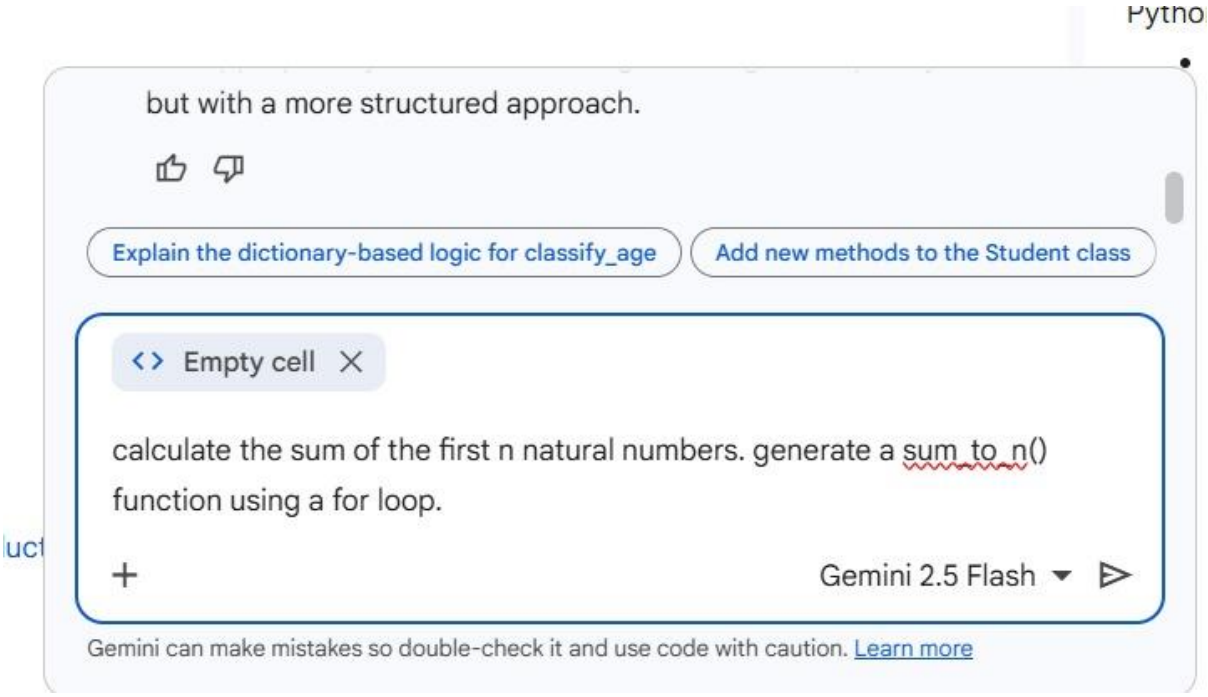
Task-04:

For and While Loops (Sum of First n Numbers)

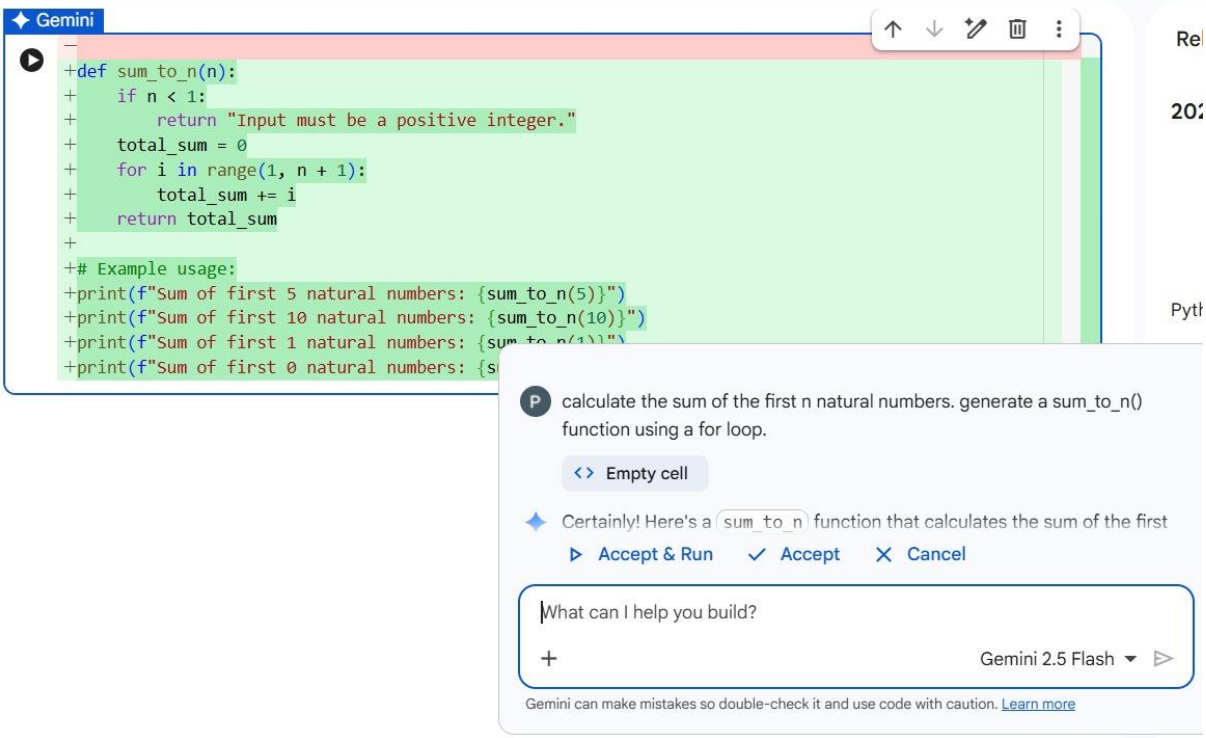
Scenario
You need to calculate the sum of the first n natural numbers.

a)UsingFor Loop:

Prompt:



Code:



Output:

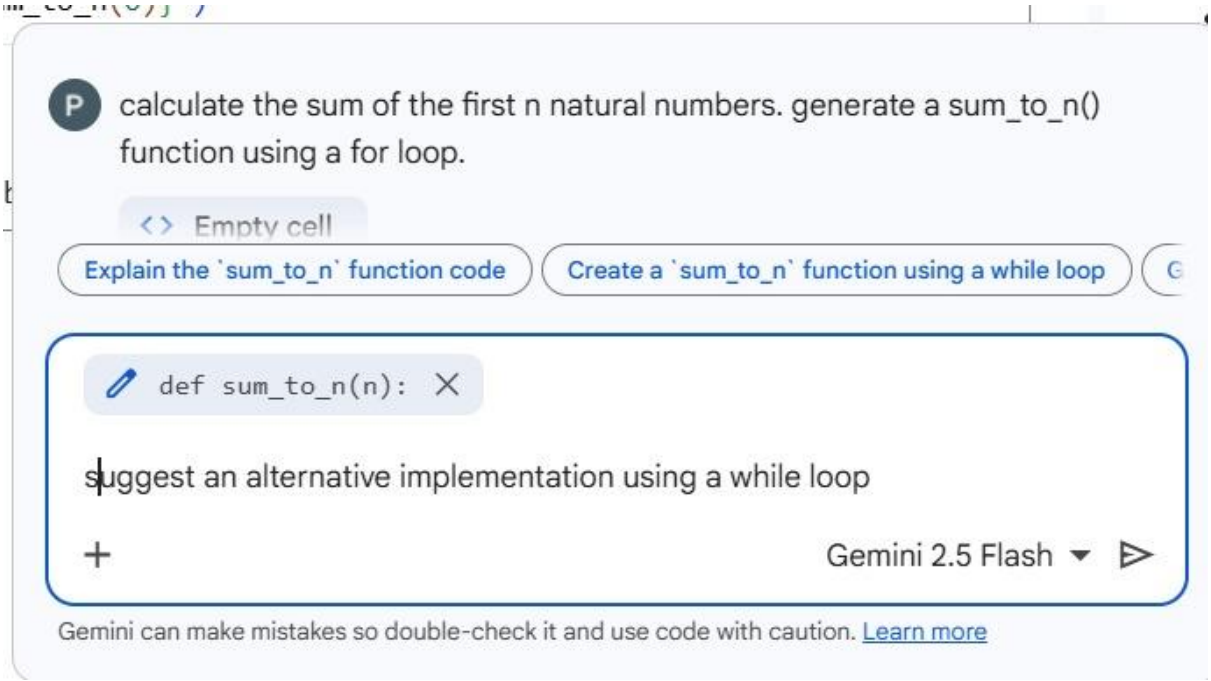
```python
def sum_to_n(n):
    if n < 1:
        return "Input must be a positive integer."
    total_sum = 0
    for i in range(1, n + 1):
        total_sum += i
    return total_sum

# Example usage:
print(f"Sum of first 5 natural numbers: {sum_to_n(5)}")
print(f"Sum of first 10 natural numbers: {sum_to_n(10)}")
print(f"Sum of first 1 natural numbers: {sum_to_n(1)}")
print(f"Sum of first 0 natural numbers: {sum_to_n(0)}")
```

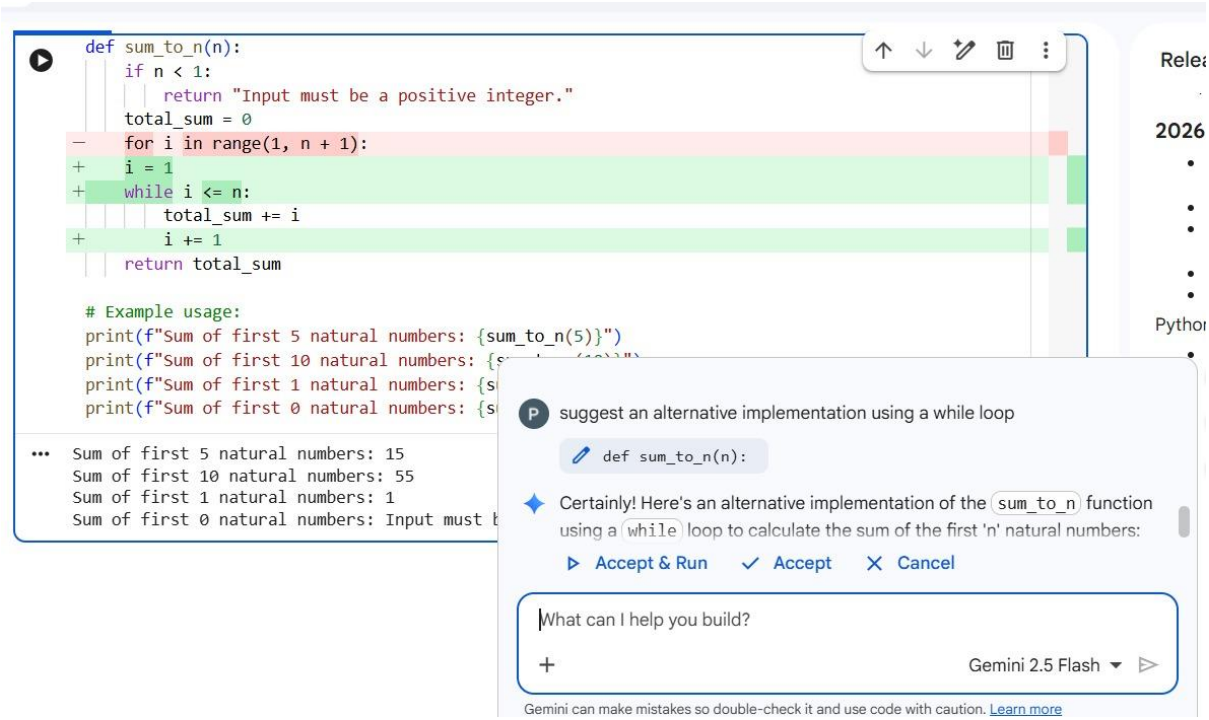
```
Sum of first 5 natural numbers: 15
Sum of first 10 natural numbers: 55
Sum of first 1 natural numbers: 1
Sum of first 0 natural numbers: Input must be a positive integer.
```

b)using While Loop:

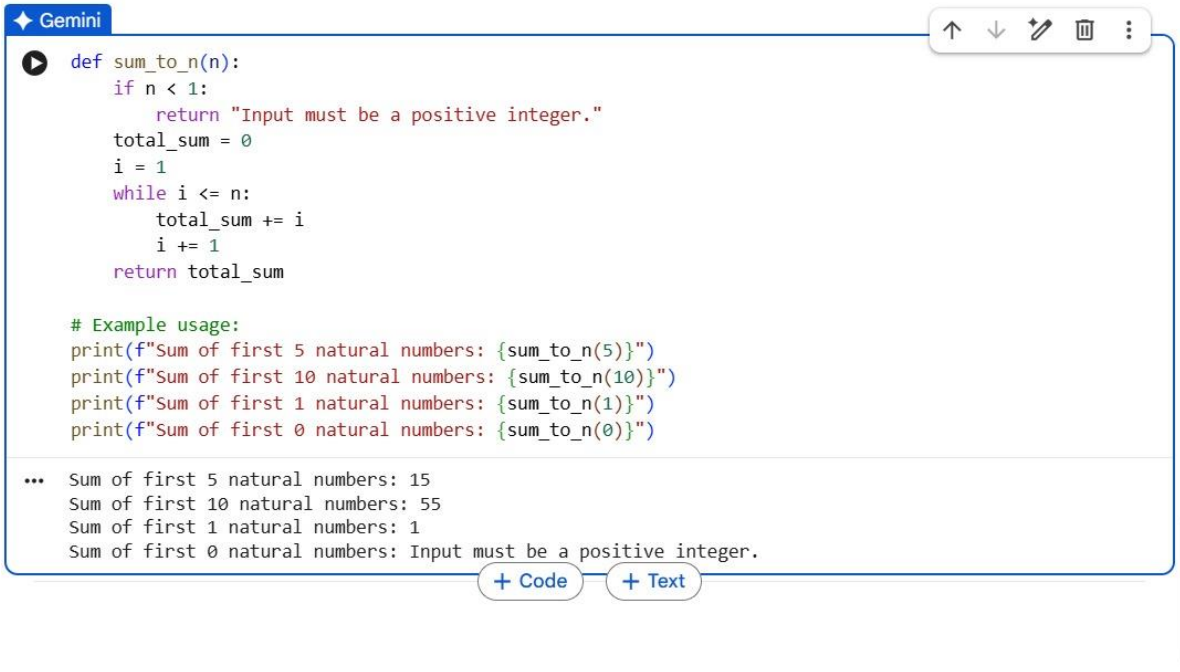
Prompt:



Code:



Output:



Explanation:

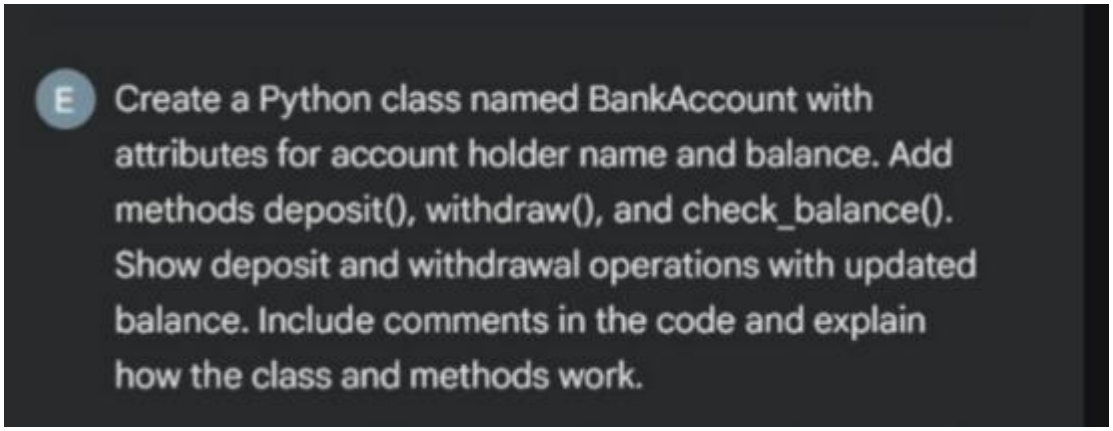
- A function named sum_to_n() is defined to calculate the sum of natural numbers.

- 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:

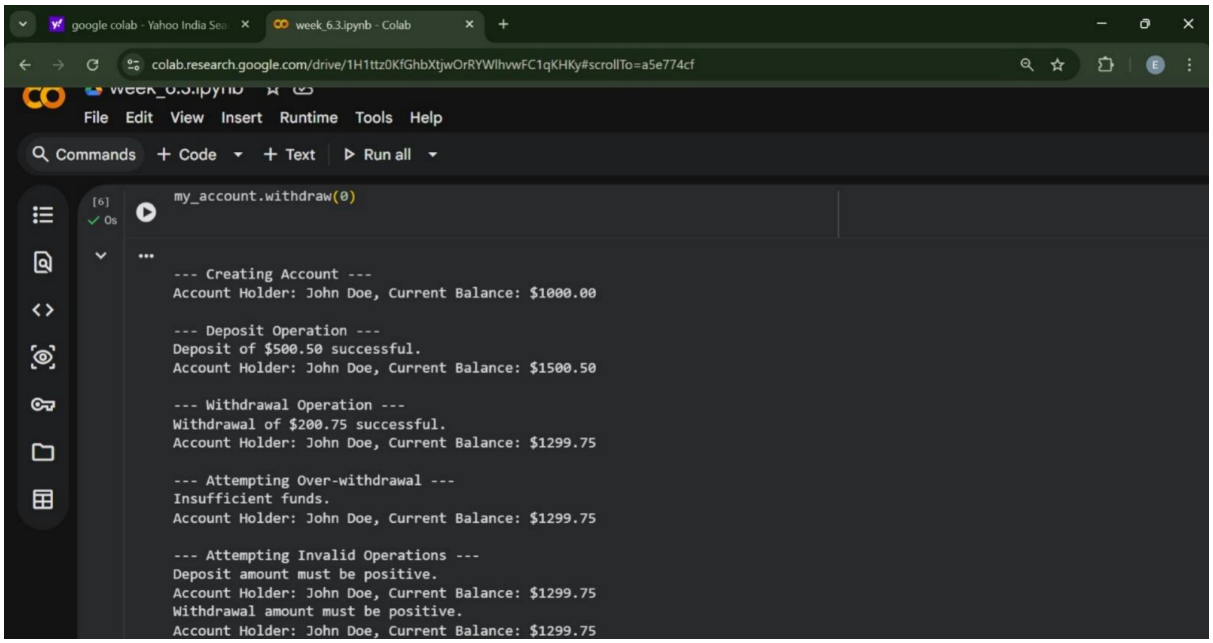
Classes (Bank Account Class)
Scenario
You are designing a basic banking application.

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.