PROGRAM: B.TECH

SPECIALIZATION: CSE - AIML

COURSE TITLE: AI CODING

SEMESTER : 3RD SEM

NAME OF STUDENT: SRIYA

ENROLLMENT NO: 2403A51350

BATCH NO: 01

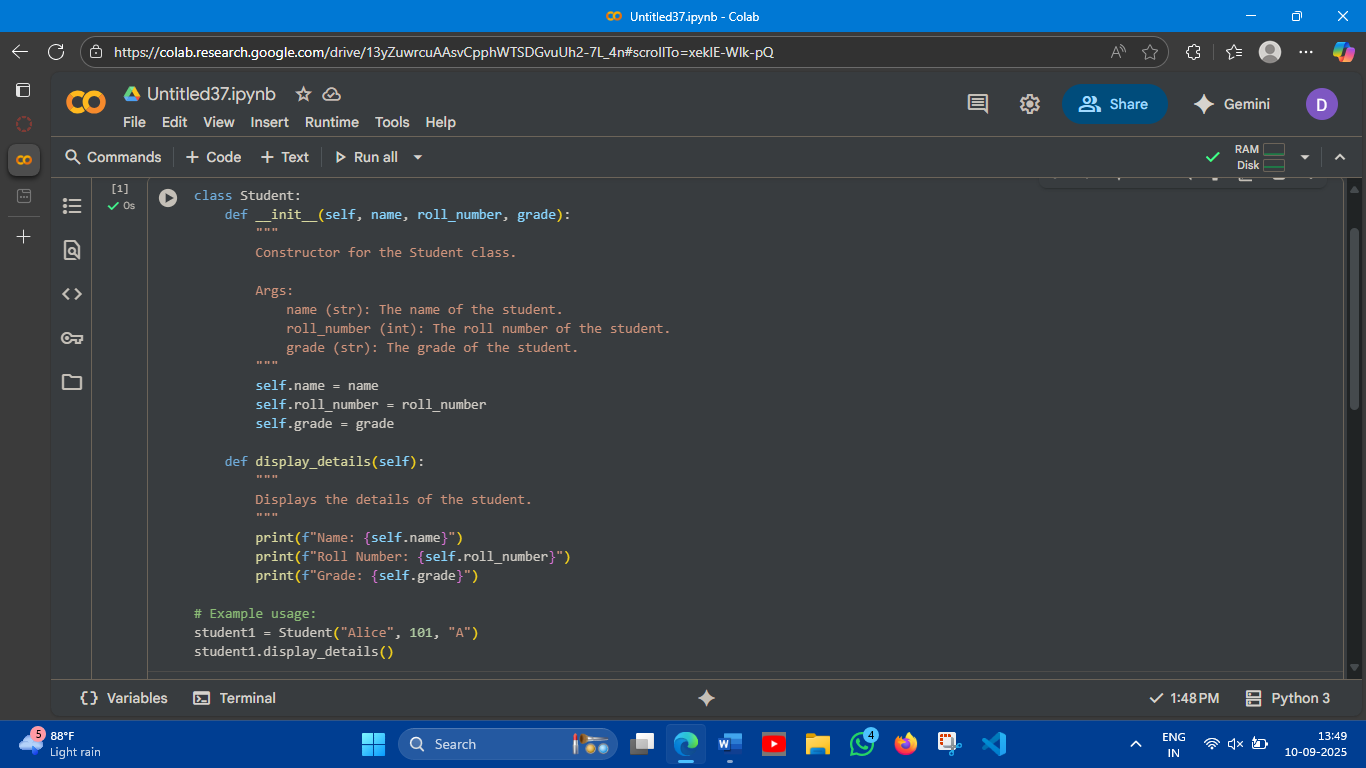
**Task Description#1 (Classes)  
• Use AI to complete a Student class with attributes and a method.  
• Check output  
• Analyze the code generated by AI tool.**

**Expected Output#1  
• Class with constructor and display\_details() method**

**Prompt:**

**generate a code for student class with constructor , attributes and display\_details() method**

**screenshots:**

****

**Output:**

**Name: Alice**

**Roll Number: 101**

**Grade: A**

**Code Analysis:**

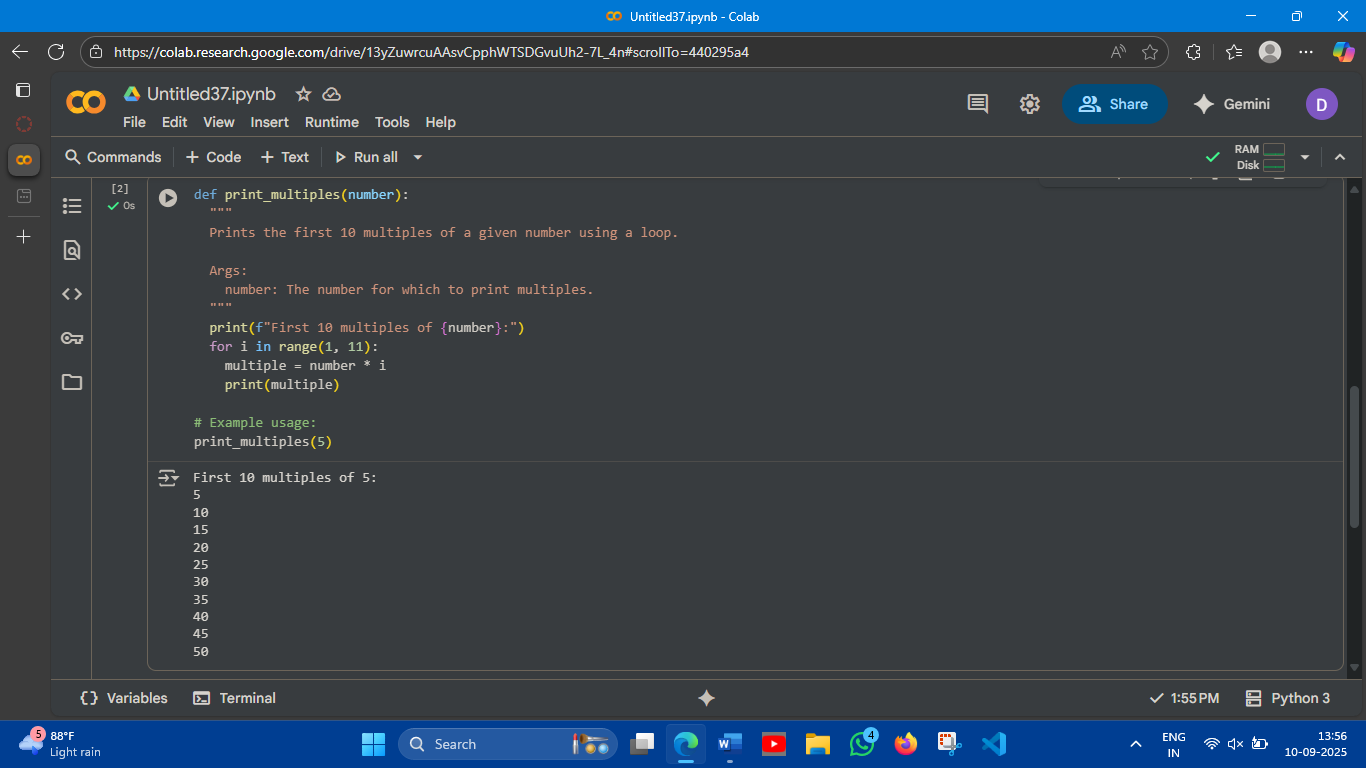
* **class Student:: This line defines a new class named Student. Classes are blueprints for creating objects (instances).**
* **def \_\_init\_\_(self, name, roll\_number, grade):: This is the constructor method. It's called automatically when you create a new Student object.**
  + **self: Refers to the instance of the class being created.**
  + **name, roll\_number, grade: These are parameters that you pass when creating a Student object.**
  + **Inside the constructor, self.name = name, self.roll\_number = roll\_number, and self.grade = grade assign the values passed as arguments to the object's attributes (properties).**
* **def display\_details(self):: This defines a method named display\_details within the Student class. Methods are functions that belong to a class.**
  + **self: Again, refers to the instance of the class.**
  + **This method prints the name, roll\_number, and grade attributes of the Student object in a formatted string.**
* **student1 = Student("Alice", 101, "A"): This line creates an instance of the Student class.**
  + **Student("Alice", 101, "A") calls the constructor with the values "Alice", 101, and "A" for the name, roll\_number, and grade parameters, respectively.**
  + **The resulting Student object is assigned to the variable student1.**
* **student1.display\_details(): This line calls the display\_details method on the student1 object. This executes the code inside the display\_details method for student1, printing its details**

**Task Description#2 (Loops)  
• Prompt AI to complete a function that prints the first 10 multiples of a number using a  
loop.  
• Analyze the generated code  
• Ask AI to generate code using other controlled looping**

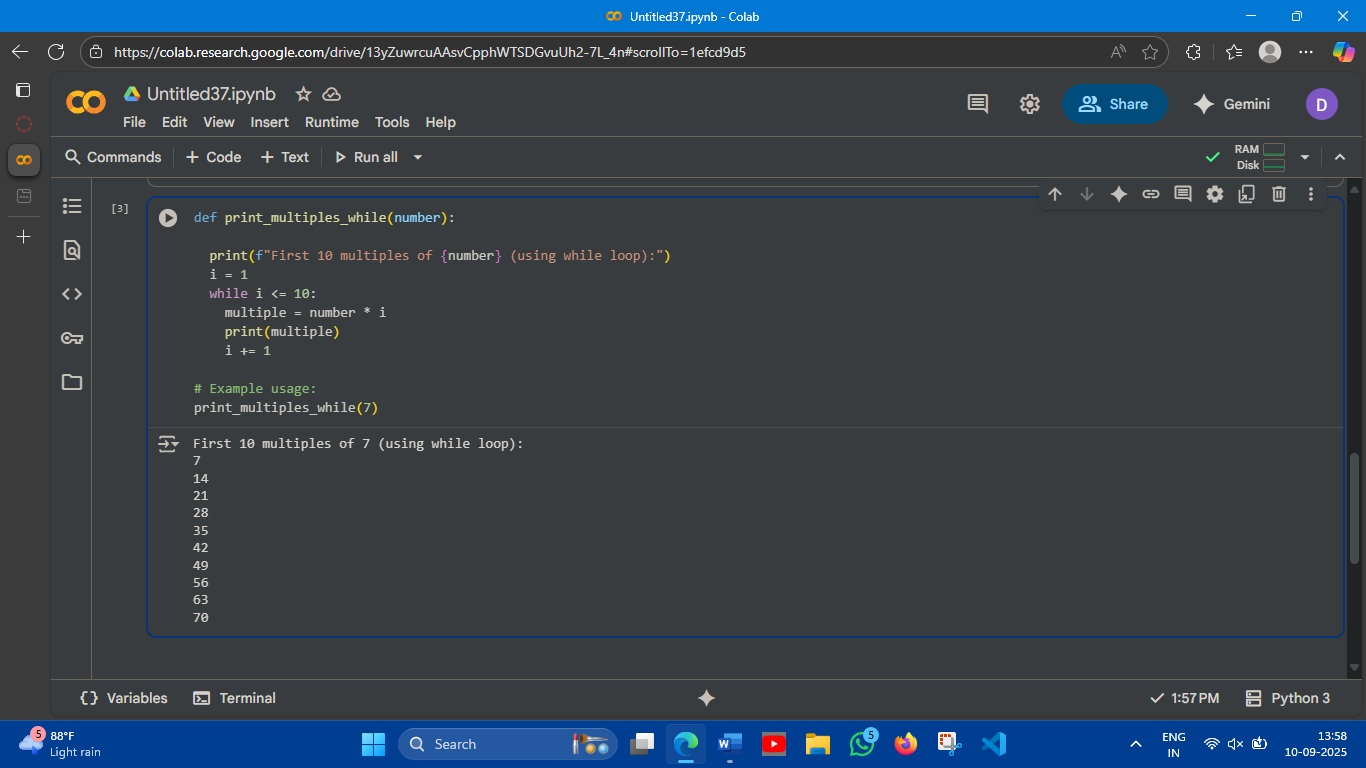
**Expected Output#2  
• Correct loop-based implementation**

**Prompt:** **generate code of function with loops that prints first 10 multiples of numbers**

**Screenshots:**

****

**Generate code using other controlled looping**

****

**Code Analysis:**

* **def print\_multiples\_while(number):: This line defines a function named print\_multiples\_while that takes one argument, number.**
* **print(f"First 10 multiples of {number} (using while loop):"): This line prints a descriptive header indicating the output that follows.**
* **i = 1: This initializes a variable i to 1. This variable will act as a counter for the multiples.**
* **while i <= 10:: This is the while loop condition. The code block inside the while loop will continue to execute as long as the value of i is less than or equal to 10.**
* **multiple = number \* i: Inside the loop, this line calculates the current multiple by multiplying the input number by the counter i.**
* **print(multiple): This line prints the calculated multiple.**
* **i += 1: This is a crucial step in a while loop. It increments the value of i by 1 in each iteration. This ensures that the loop will eventually terminate when i becomes greater than 10. If this line were missing, the loop would run indefinitely (an infinite loop).**
* **print\_multiples\_while(7): This line calls the print\_multiples\_while function with the argument 7, triggering the execution of the code within the function to print the first 10 multiples of 7.**

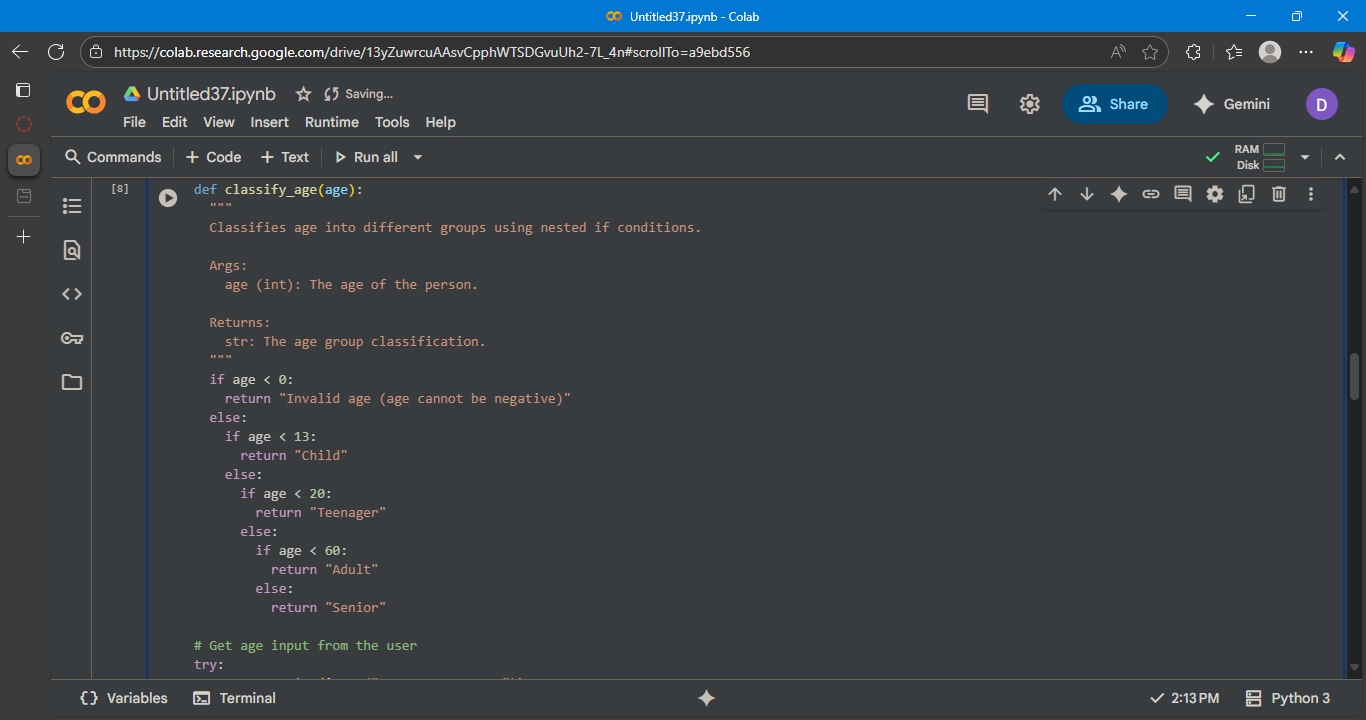
**Task Description#3 (Conditional Statements)  
• Ask AI to write nested if-elif-else conditionals to classify age groups.  
• Analyze the generated code  
• Ask AI to generate code using other conditional statements**

**Expected Output#3  
• Age classification function with appropriate conditions and with explanation**

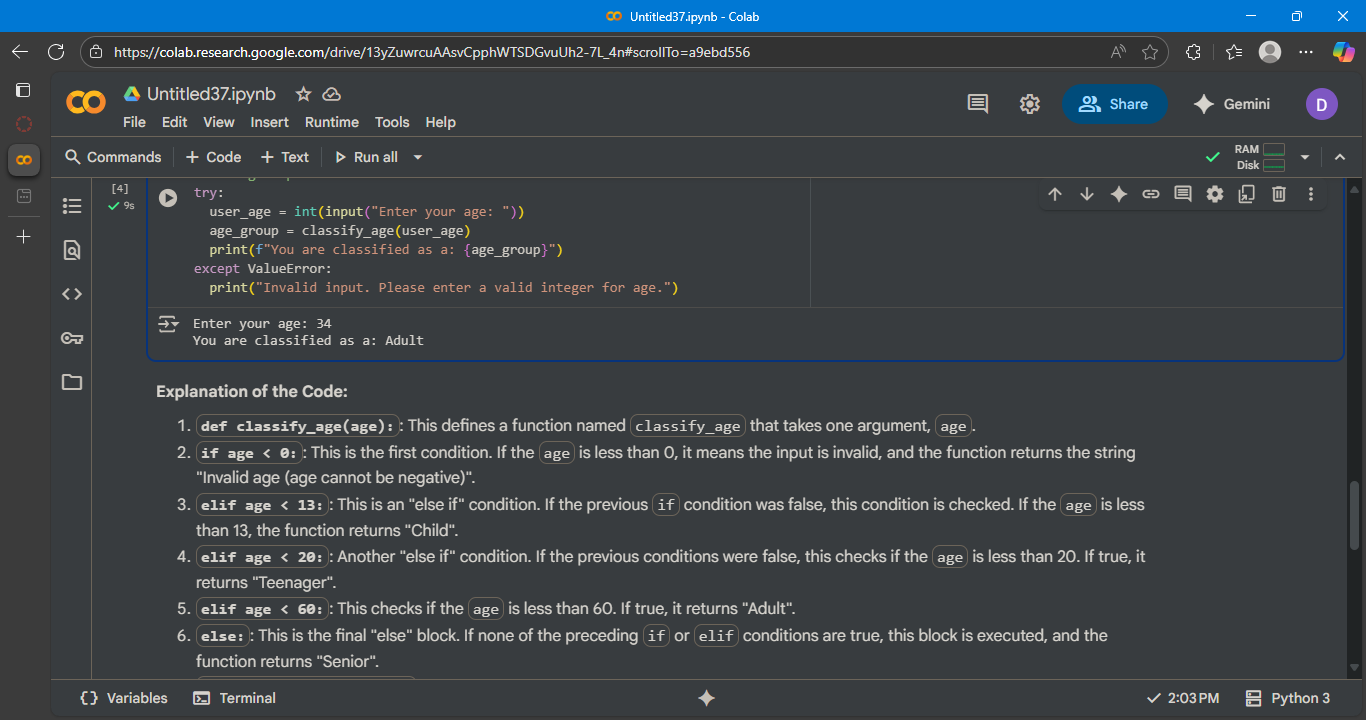
**Prompt:**

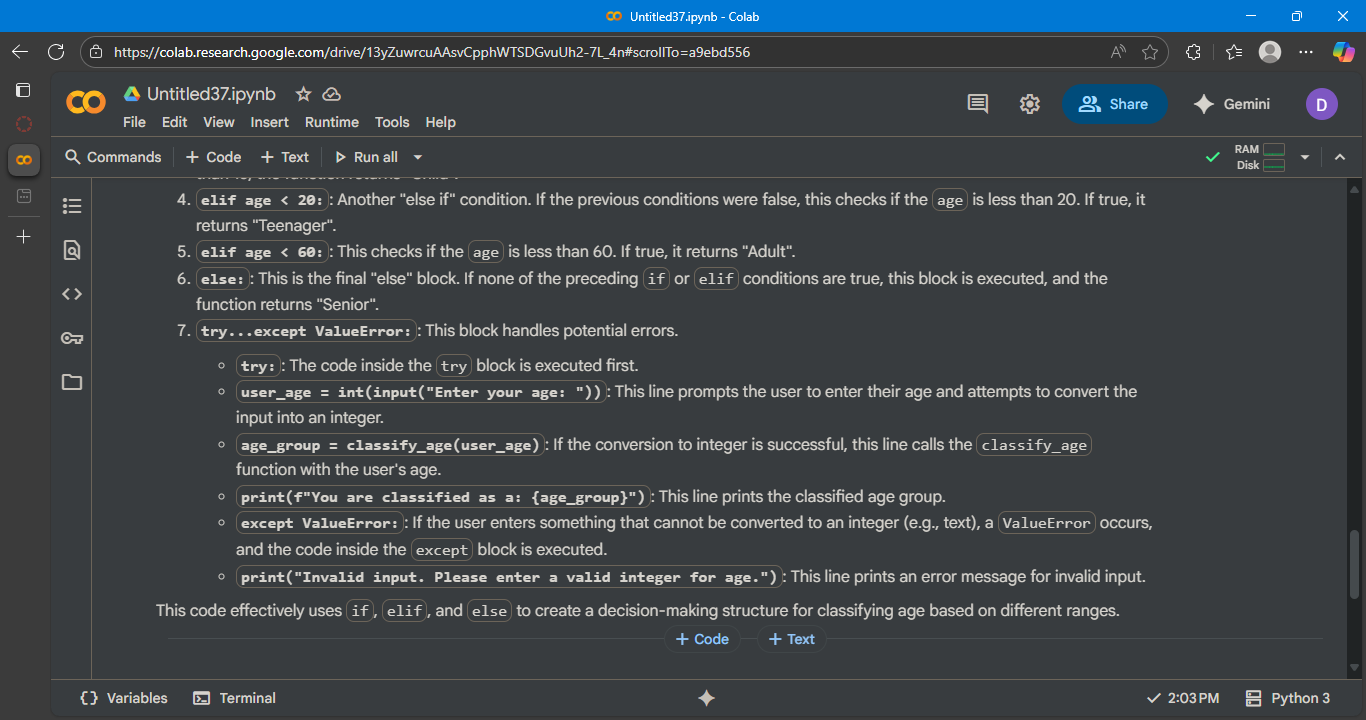
**generate code for Age classification function with appropriate conditions and with explanation using if,elif,else conditionals to classify age groups with user given input**

**Screenshots:**

****

**Explaination:**

****

****

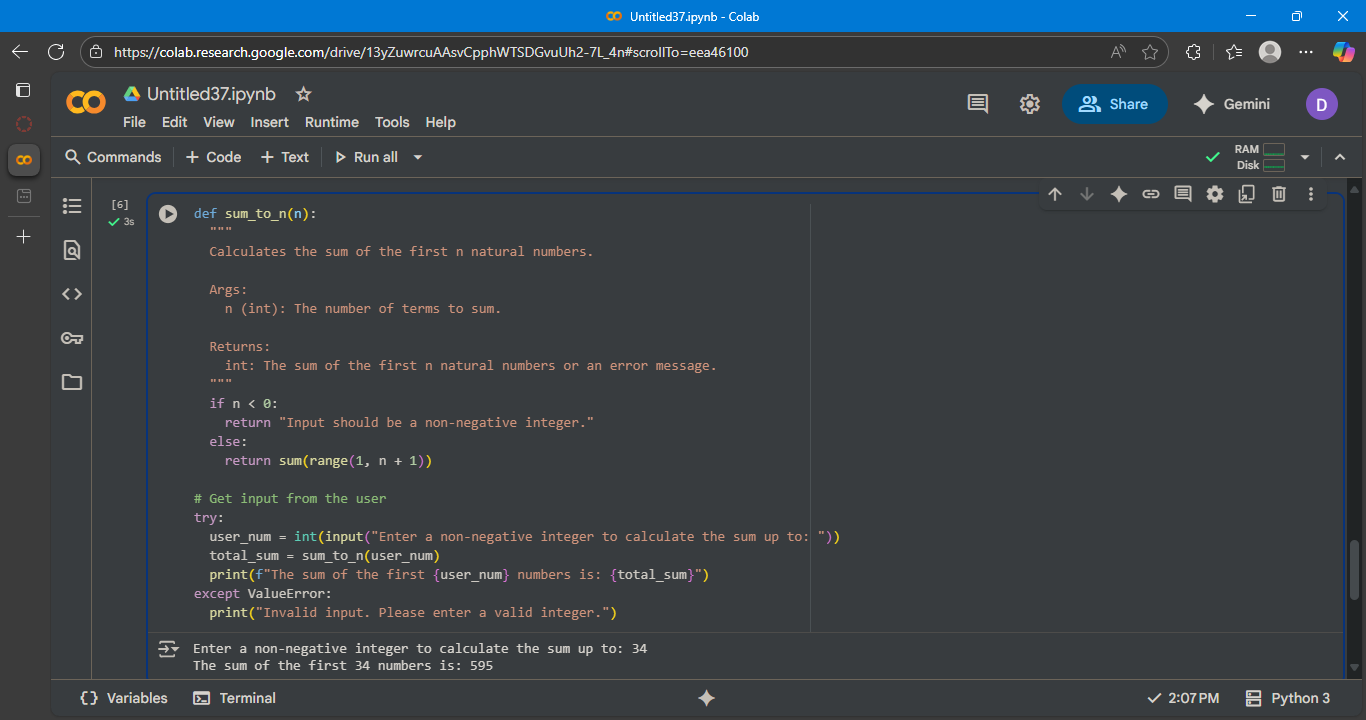
**This code defines a function classify\_age that categorizes an input age. It uses if, elif, and else to check different age ranges. The try...except block handles potential invalid input from the user. It returns a string indicating the age group or an error message for negative/invalid ages.**

**Task Description#4 (For and While loops)  
• Generate a sum\_to\_n() function to calculate sum of first n numbers  
• Analyze the generated code  
• Get suggestions from AI with other controlled looping  
Expected Output#4  
• Python code with explanation**

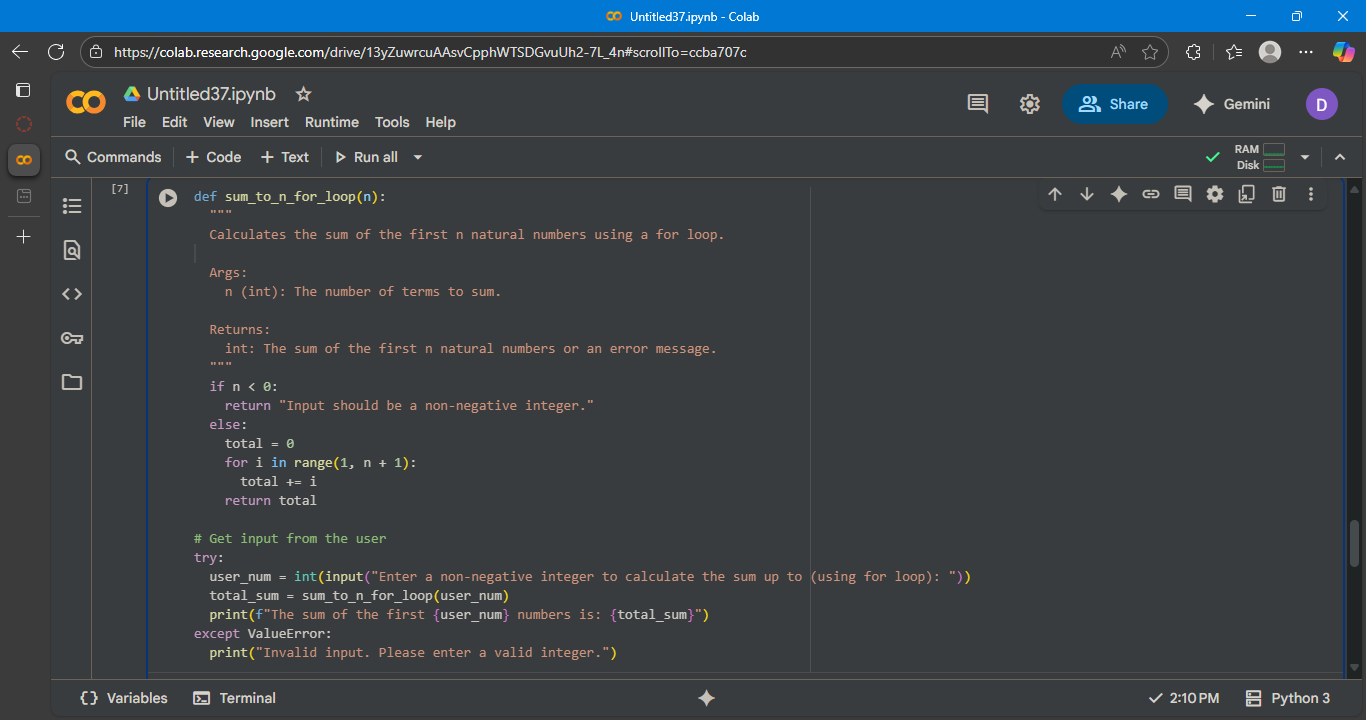
**Prompt:**

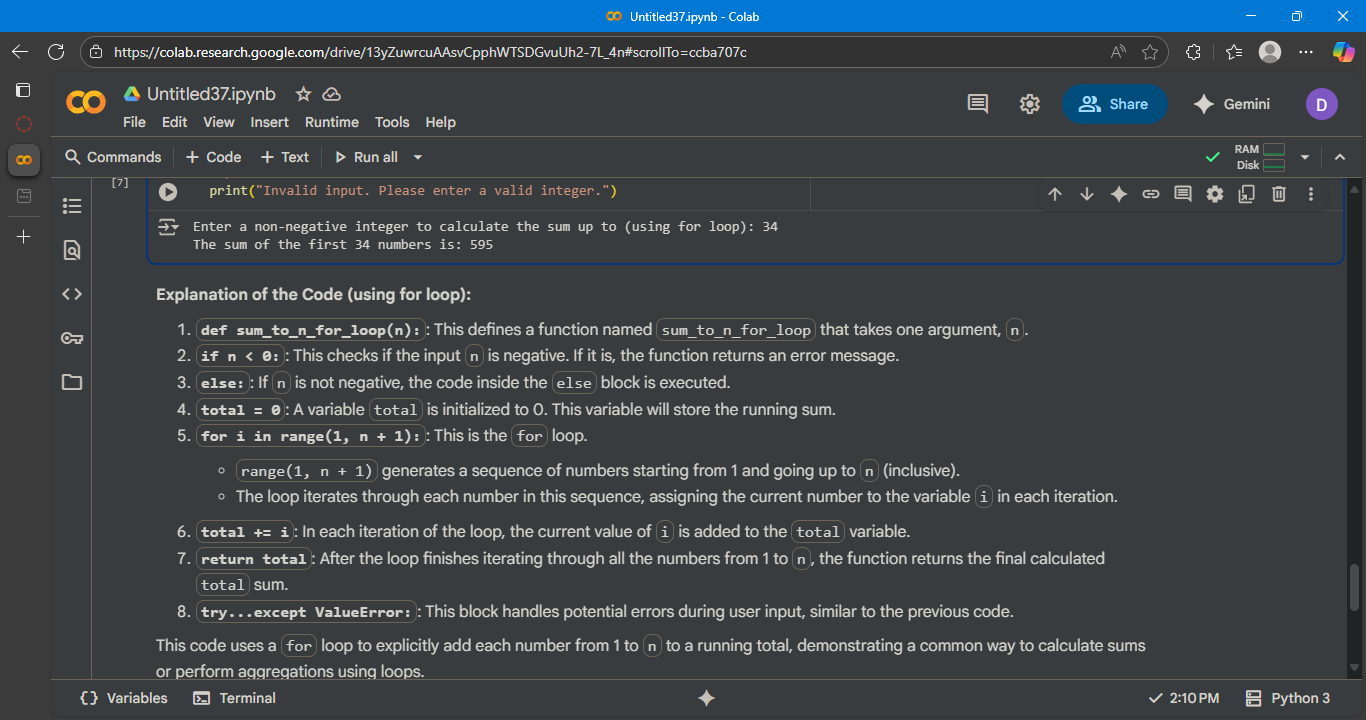
**Generate a sum\_to\_n() function to calculate sum of first n numbers with user given input**

**Screenshots:**

****

**generate other controlled looping with explaination**

****

****

**Code analysis:**

**This function calculates the sum of the first n non-negative integers. It initializes a total to zero. A for loop iterates from 1 up to n. In each iteration, the current number is added to total. Finally, the accumulated total is returned**

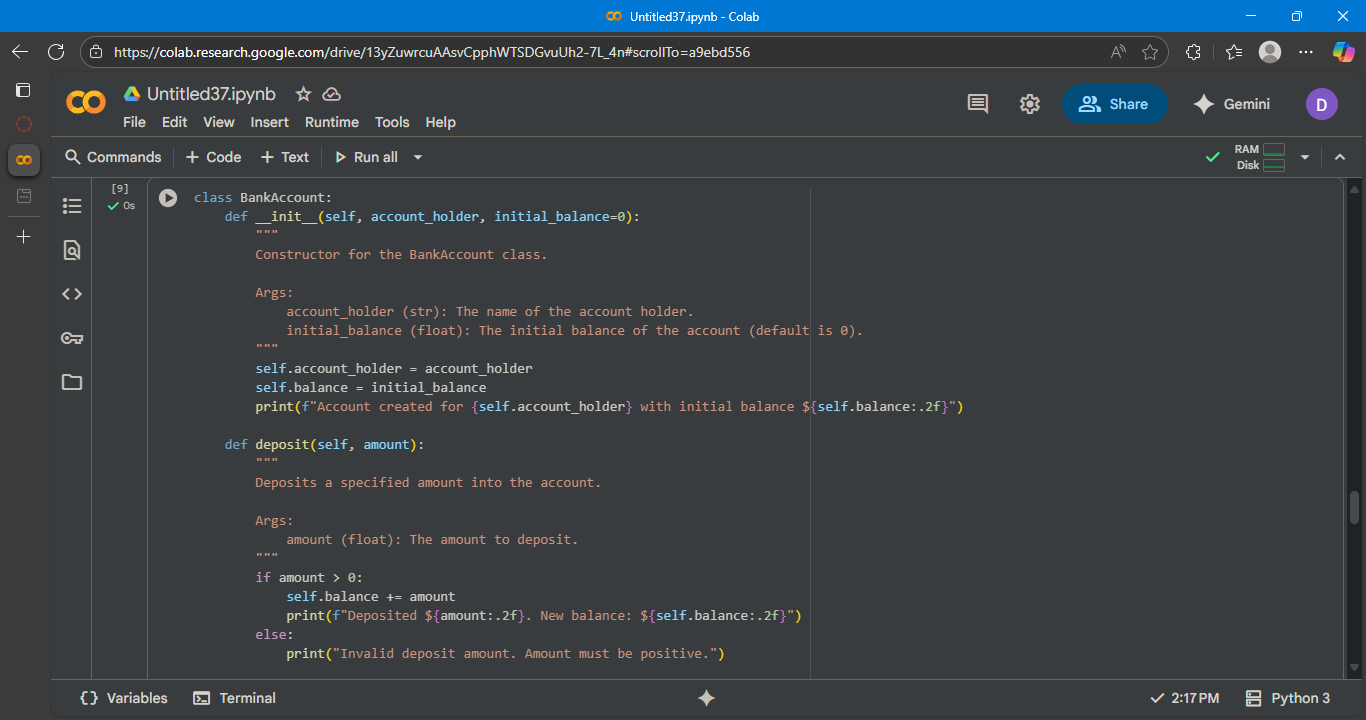
**Task Description#5 (Class)  
• Use AI to build a BankAccount class with deposit, withdraw, and balance methods.  
• Analyze the generated code  
• Add comments and explain code**

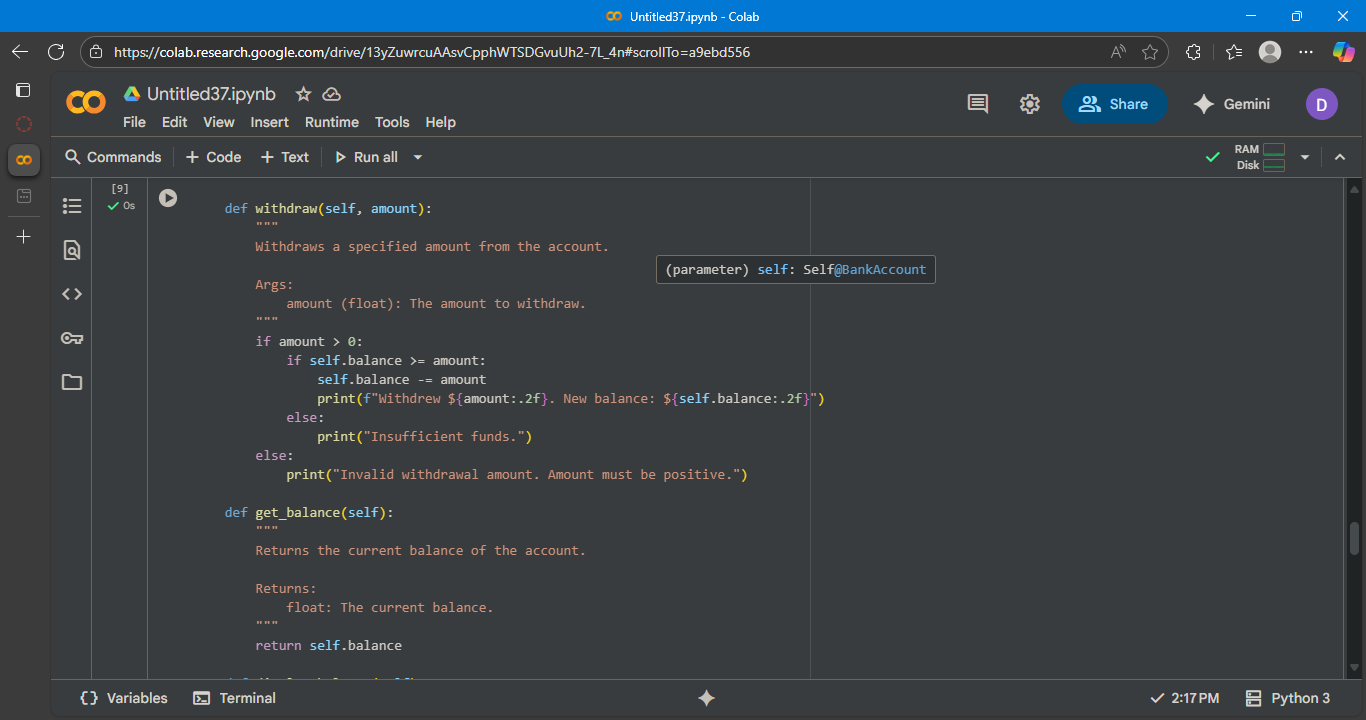
**Expected Output#5  
• Python code with explanation**

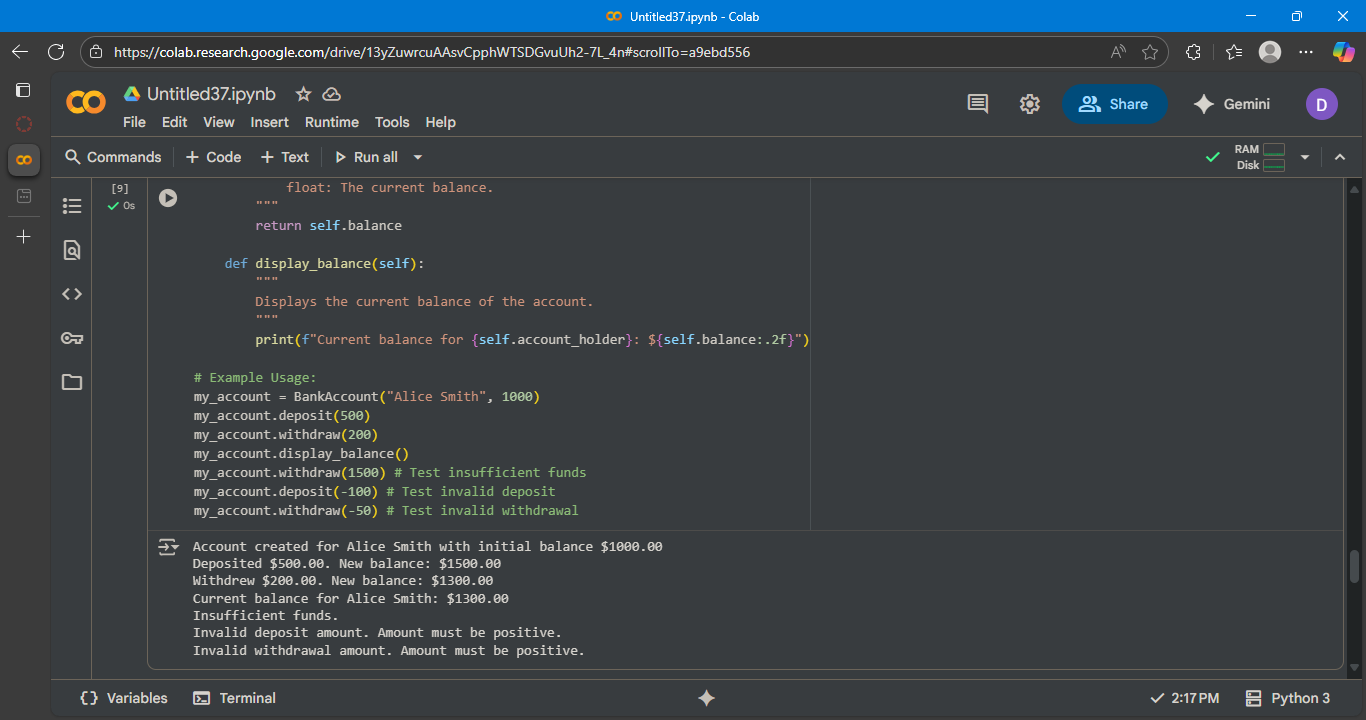
**Prompt:**

**generate a python code for building a BankAccount class with deposit, withdraw, and balance methods with comments and explainations**

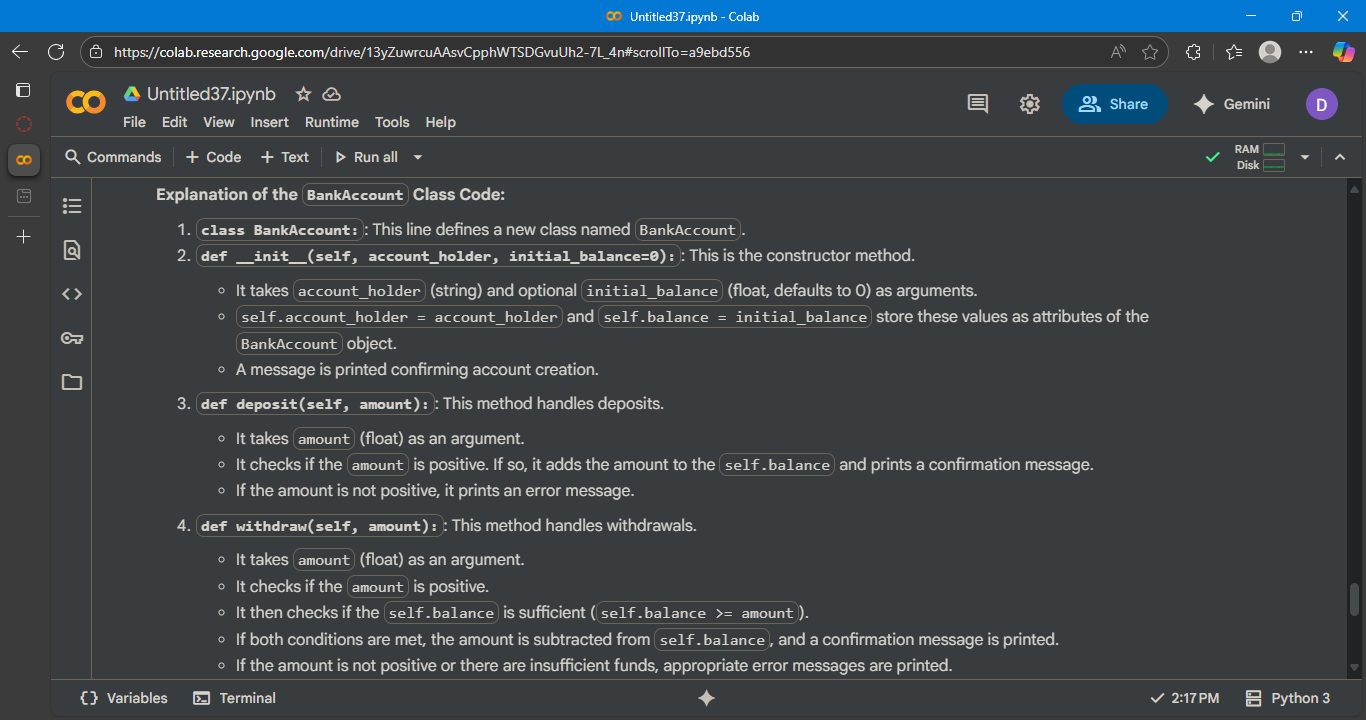
**screenshots:**

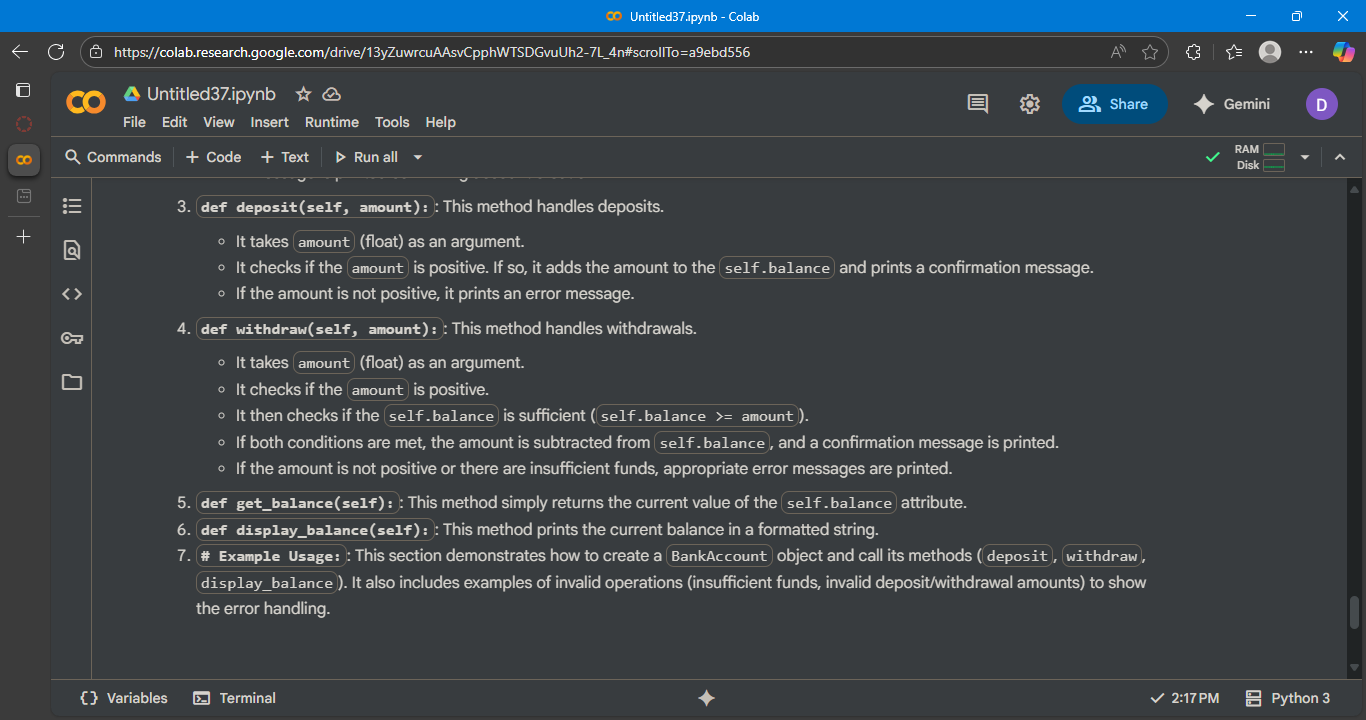
****

****

****

**Explaination:**

****

****