

## **AI ASSISTED CODING ASSIGNMENT – 7.2**

### **Lab 7:**

**Error Debugging with AI: Systematic approaches to finding and fixing bugs**

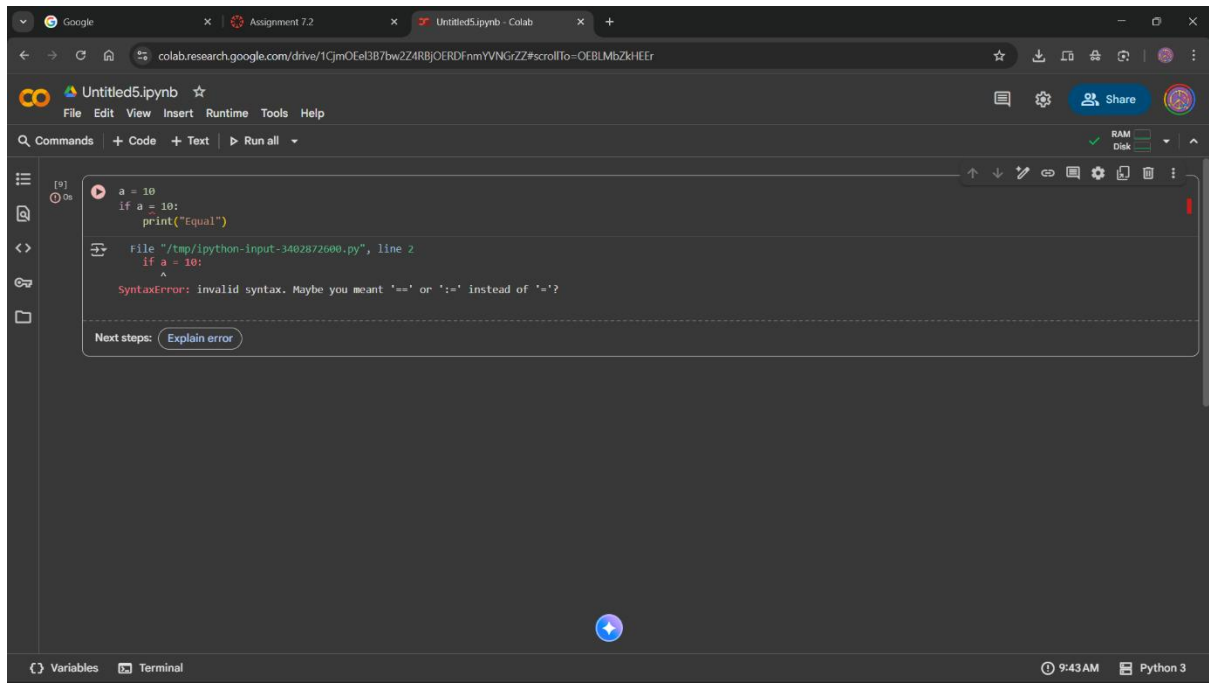
### **Lab Objectives:**

- **To identify and correct syntax, logic, and runtime errors in Python programs using AI tools.**
- **To understand common programming bugs and AI-assisted debugging suggestions.**
- **To evaluate how AI explains, detects, and fixes different types of coding errors.**
- **To build confidence in using AI to perform structured debugging practices.**

### **Lab Outcomes (LOs):**

- **Use AI tools to detect and correct syntax, logic, and runtime errors.**
- **Interpret AI-suggested bug fixes and explanations.**
- **Apply systematic debugging strategies supported by AI-generated insights.**

## • Task #1 – Syntax Error in Conditionals :



The image shows a Google Colab notebook titled "Untitled5.ipynb". The code cell contains the following Python code:

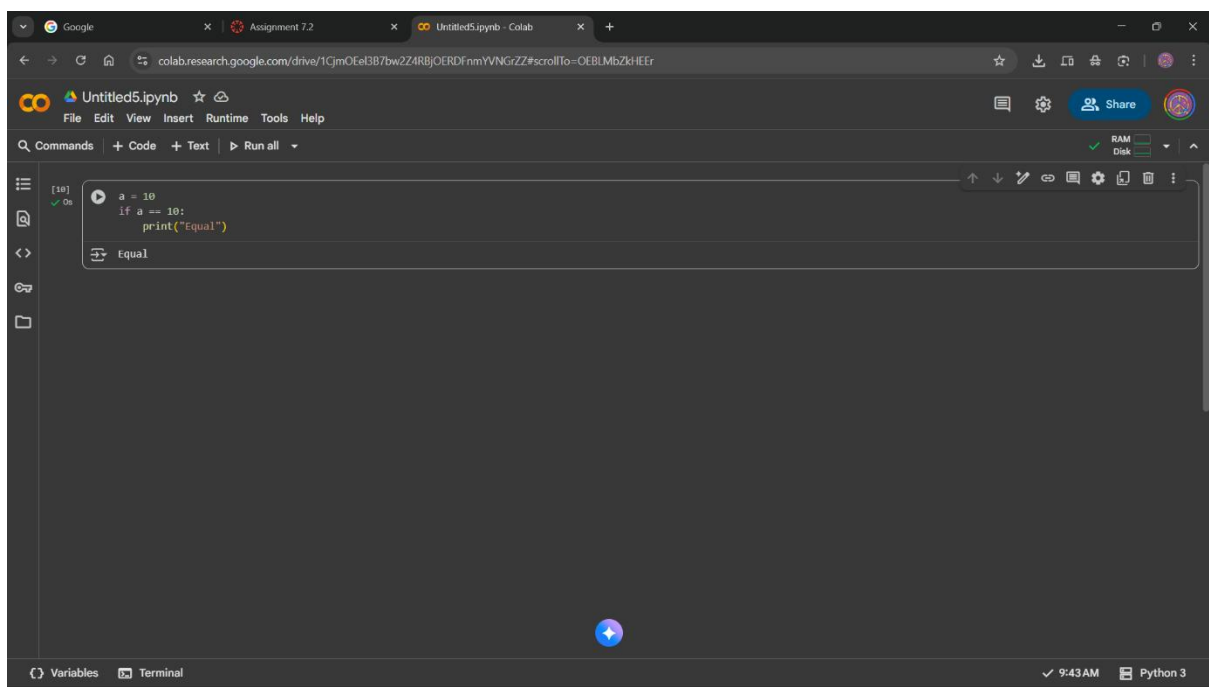
```
a = 10
if a = 10:
    print("Equal")
```

The code is highlighted in green, indicating it was executed. Below the code, a red error message is displayed:

```
File ~/tmp/ipython-input-3402872600.py, line 2
if a = 10:
    ^
SyntaxError: invalid syntax. Maybe you meant '==' or ':=' instead of '='?
```

Below the error message, there is a "Next steps:" section with a button labeled "Explain error". The bottom status bar shows "9:43 AM" and "Python 3".

## Corrected code and output :

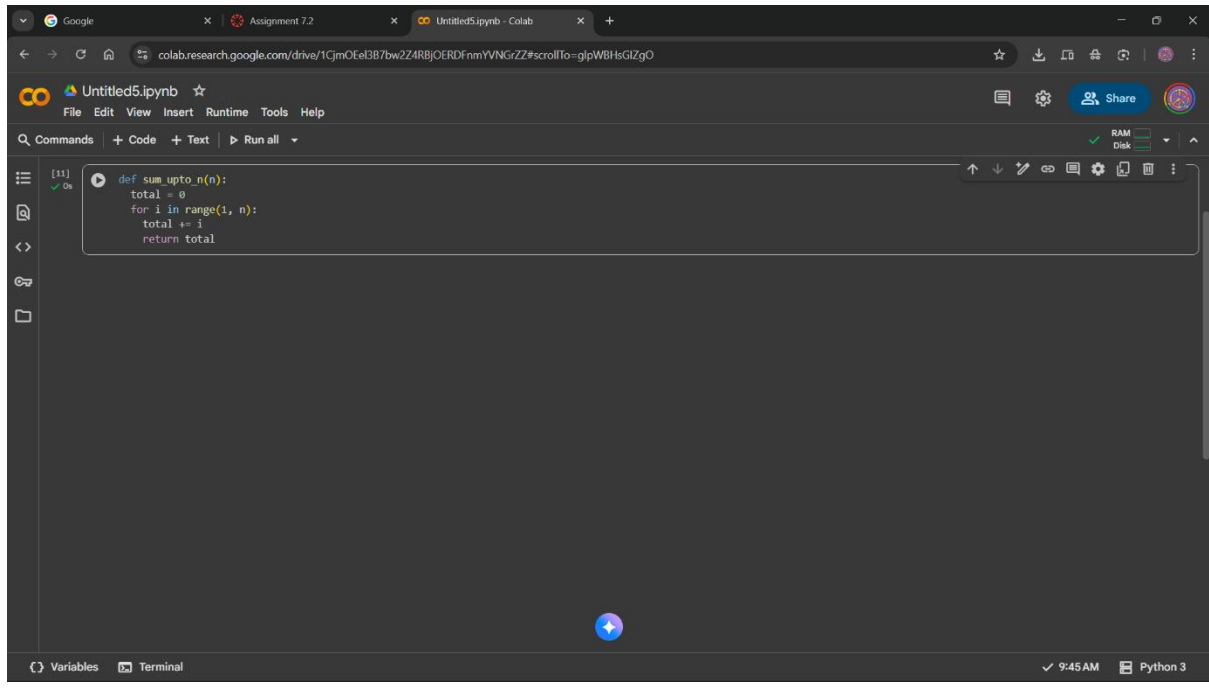


The image shows the same Google Colab notebook after the code has been corrected. The code cell now contains:

```
a = 10
if a == 10:
    print("Equal")
```

The code is highlighted in green, indicating it was executed successfully. Below the code, the output "Equal" is displayed. The bottom status bar shows "9:43 AM" and "Python 3".

## • Task #2 – Loop Off-By-One Error :

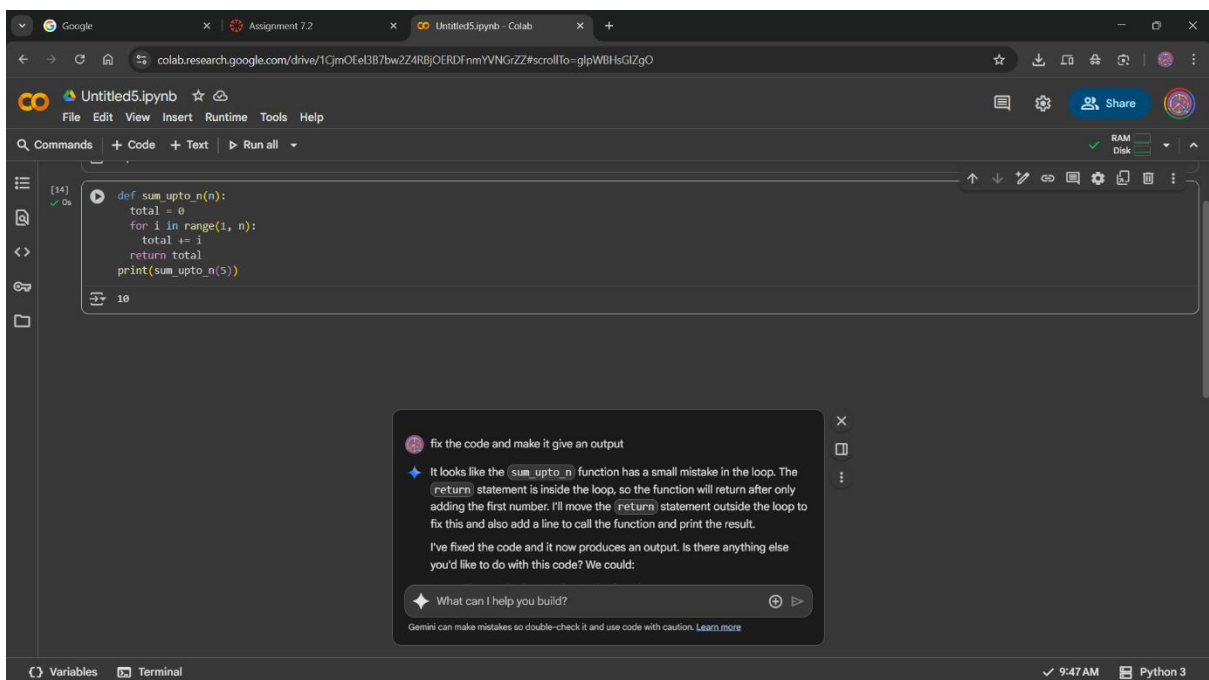


The image shows a Google Colab notebook titled 'Untitled5.ipynb'. The code in the first cell is as follows:

```
[11]: def sum_upto_n(n):  
      total = 0  
      for i in range(1, n):  
          total += i  
      return total
```

The code is syntactically correct but contains a logical error: the loop `for i in range(1, n):` does not include the value `n`, meaning the sum of numbers from 1 to `n` is not calculated. The notebook interface shows the code is ready to be run, with a 'Run all' button and a status bar at the bottom indicating 'Python 3'.

## Corrected code and output :



The image shows the same Colab notebook with the code corrected to include `n` in the range:

```
[14]: def sum_upto_n(n):  
      total = 0  
      for i in range(1, n):  
          total += i  
      return total  
      print(sum_upto_n(5))
```

The output of the cell is the number 10. Below the code cell, a Gemini AI assistant has provided a helpful message:

fix the code and make it give an output

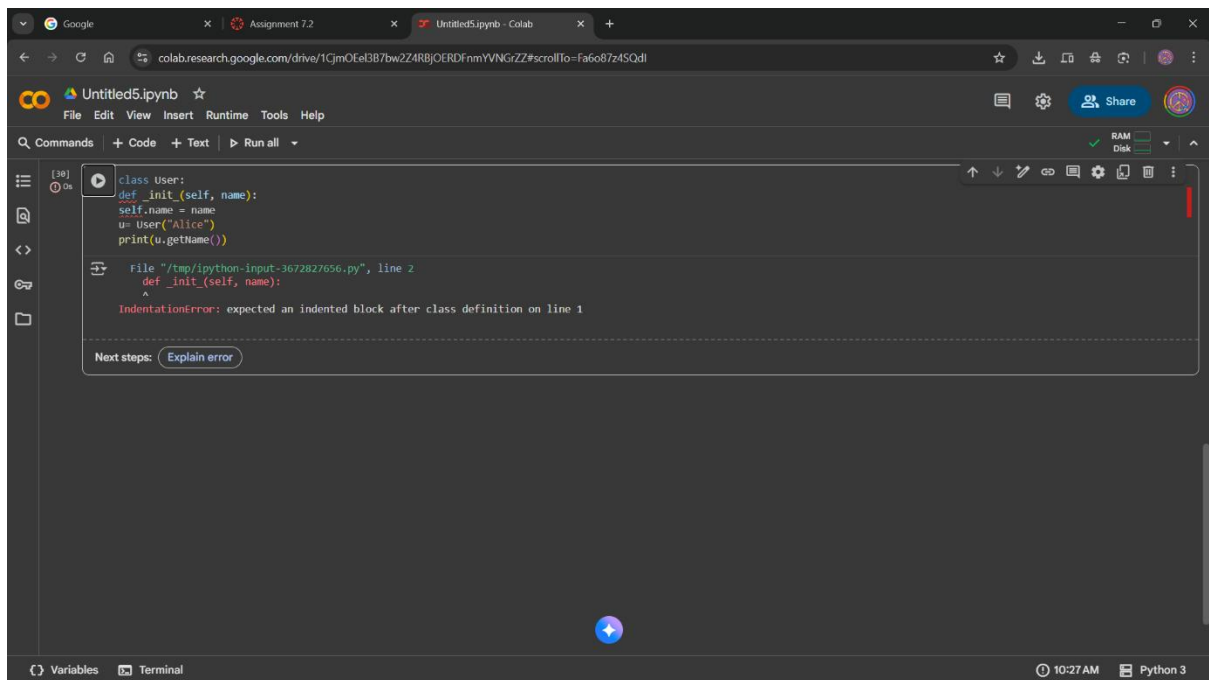
It looks like the `sum_upto_n` function has a small mistake in the loop. The `return` statement is inside the loop, so the function will return after only adding the first number. I'll move the `return` statement outside the loop to fix this and also add a line to call the function and print the result.

I've fixed the code and it now produces an output. Is there anything else you'd like to do with this code? We could:

What can I help you build?

Gemini can make mistakes so double-check it and use code with caution. [Learn more](#)

## Task #3 – Error : AttributeError :



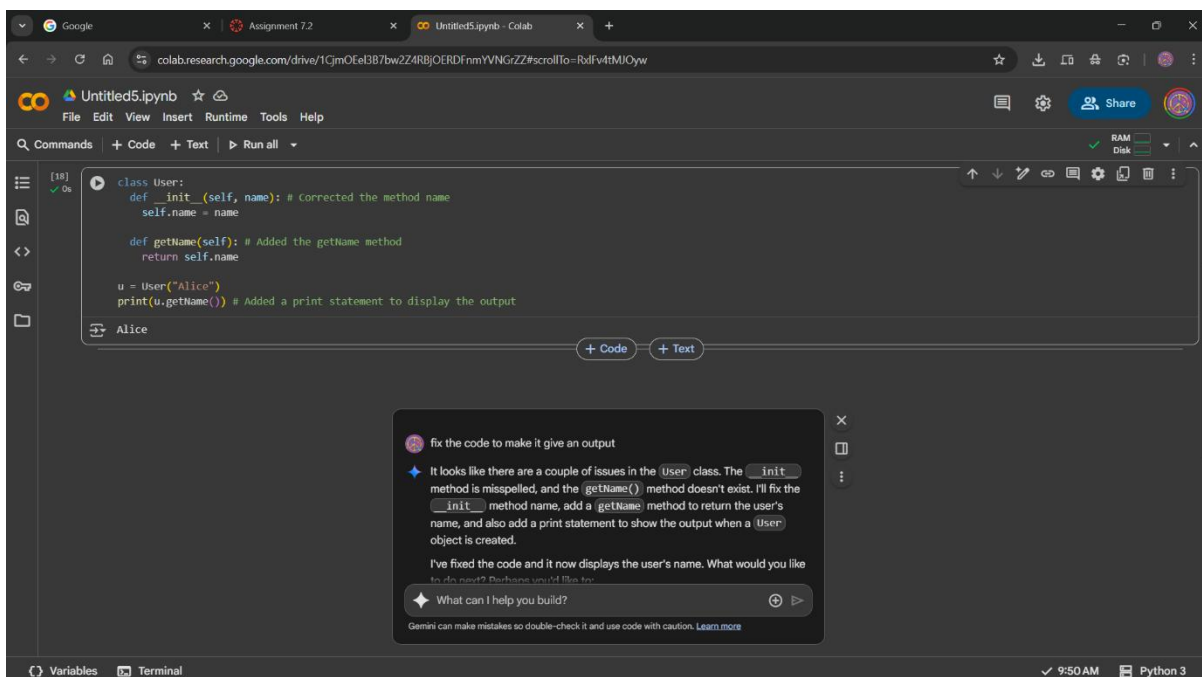
The screenshot shows a Jupyter Notebook interface with a code cell containing the following Python code:

```
class User:
    def __init__(self, name):
        self.name = name
        u = User("Alice")
        print(u.getName())

file "/tmp/ipython-input-3672827656.py", line 2
    def __init__(self, name):
        ^
IndentationError: expected an indented block after class definition on line 1
```

Below the code cell, there is a message box that says "Next steps: Explain error". The bottom status bar shows "Python 3" and the time "10:27 AM".

## Corrected code and output :



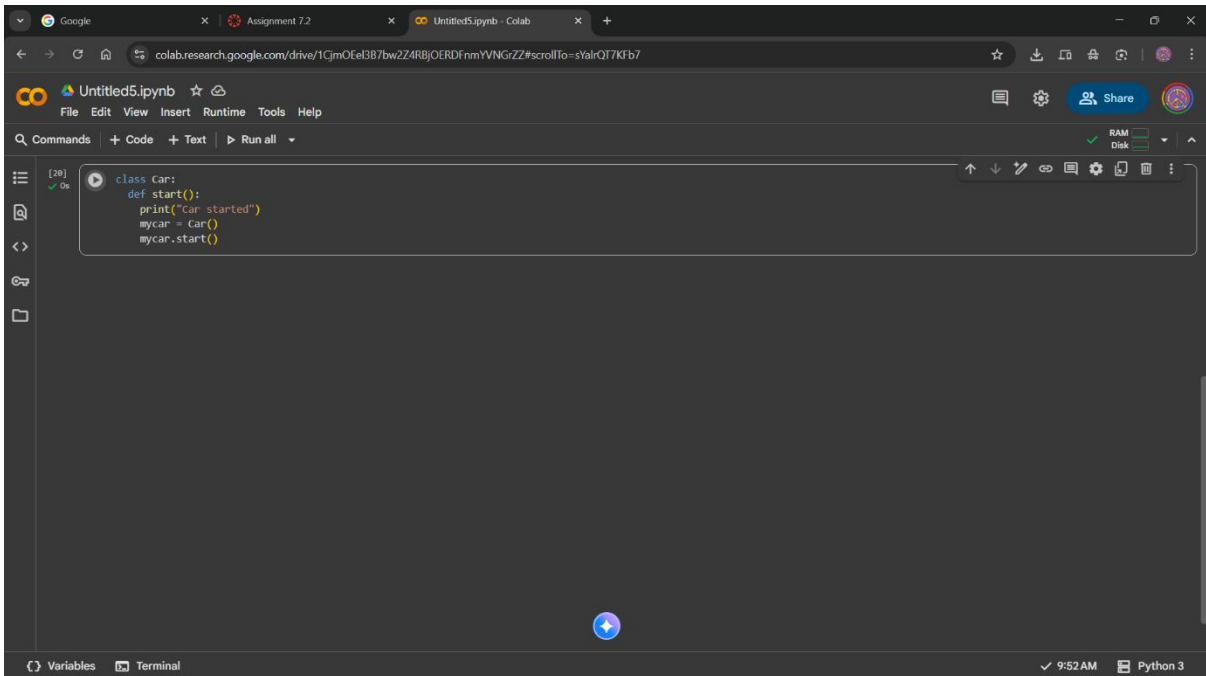
The screenshot shows the same Jupyter Notebook interface, but with the code corrected. The code cell now contains:

```
class User:
    def __init__(self, name): # Corrected the method name
        self.name = name

    def getName(self): # Added the getName method
        return self.name

u = User("Alice")
print(u.getName()) # Added a print statement to display the output
```

The output of the code cell is "Alice". Below the code cell, there is a message box that says "fix the code to make it give an output". The bottom status bar shows "Python 3" and the time "9:50 AM".



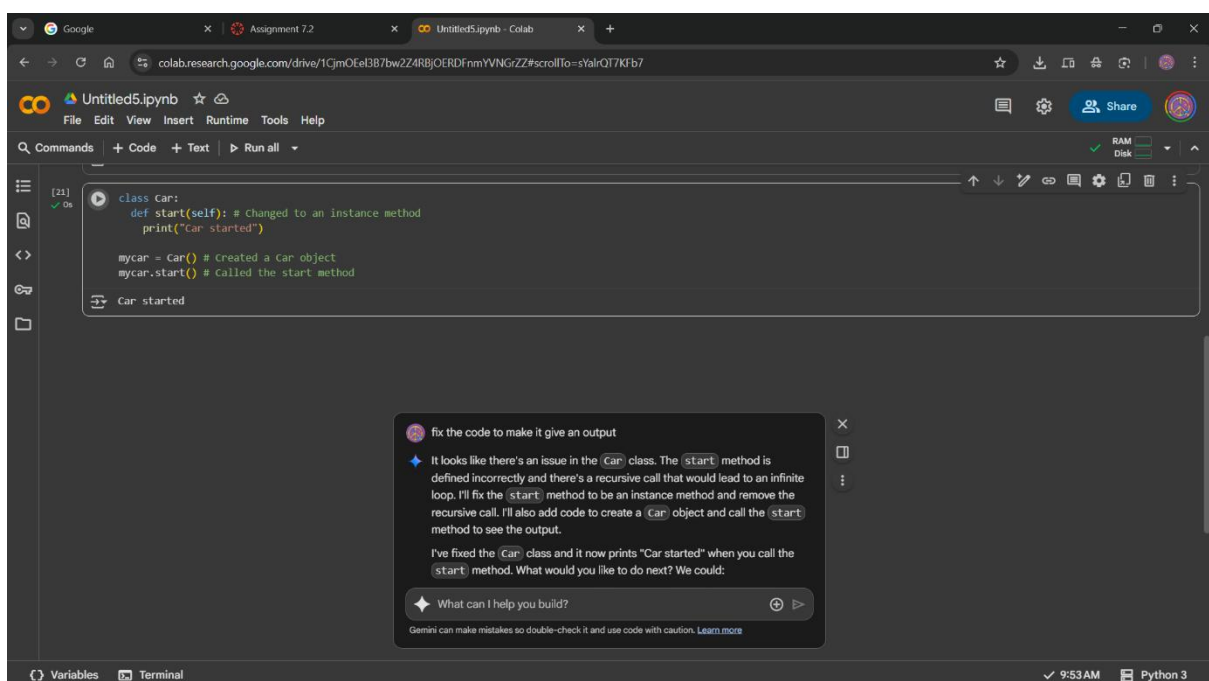
The image shows a Google Colab notebook titled 'Untitled5.ipynb'. The code in the first cell is as follows:

```
[20] class Car:
      def start():
        print("car started")
      mycar = Car()
      mycar.start()
```

The code is executed, and the output is 'car started'. The notebook interface shows the 'Commands' tab, a 'Run all' button, and a status bar at the bottom indicating '9:52 AM' and 'Python 3'.

## Task #4 – Incorrect Class Attribute Initialization

### Corrected code and output :



The image shows the same Google Colab notebook, but with the code corrected. The code in the first cell is as follows:

```
[21] class Car:
      def start(self): # Changed to an instance method
        print("car started")

      mycar = Car() # Created a Car object
      mycar.start() # Called the start method
```

The code is executed, and the output is 'car started'. A Gemini AI assistant overlay is visible, providing feedback on the code changes:

fix the code to make it give an output

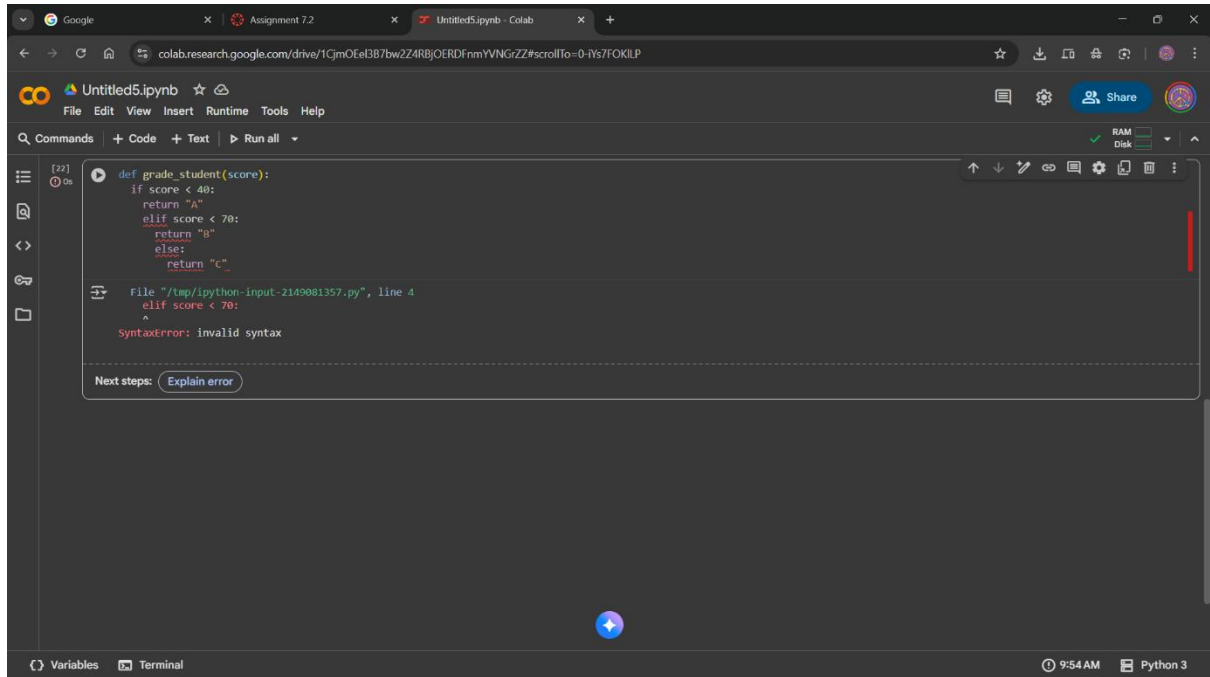
It looks like there's an issue in the `Car` class. The `start` method is defined incorrectly and there's a recursive call that would lead to an infinite loop. I'll fix the `start` method to be an instance method and remove the recursive call. I'll also add code to create a `Car` object and call the `start` method to see the output.

I've fixed the `Car` class and it now prints "Car started" when you call the `start` method. What would you like to do next? We could:

What can I help you build?

Gemini can make mistakes so double-check it and use code with caution. [Learn more](#)

# Task #5 – Conditional Logic Error in Grading



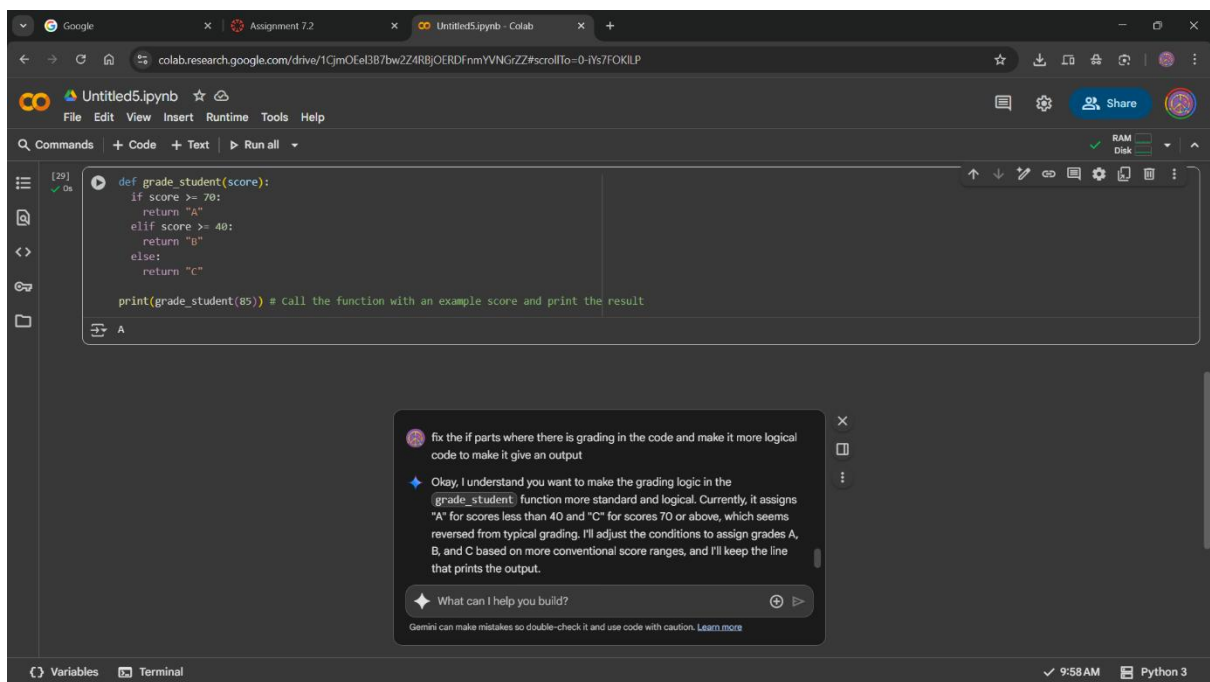
The screenshot shows a Google Colab notebook titled "Untitled5.ipynb". The code cell contains a Python function `def grade_student(score):` with a conditional logic error. The function is intended to return "A" for scores < 40, "B" for scores < 70, and "C" otherwise. However, there is a syntax error on line 4. The error message is: `SyntaxError: invalid syntax`. The error points to the line `elif score < 70:`. The "Next steps" section shows a button labeled "Explain error".

```
[22] def grade_student(score):  
      if score < 40:  
          return "A"  
      elif score < 70:  
          return "B"  
      else:  
          return "C"  
      File "/tmp/ipython-input-2149081357.py", line 4  
          ^  
SyntaxError: invalid syntax
```

Next steps: [Explain error](#)

## System

# Corrected code and Output :



The screenshot shows a Google Colab notebook interface. The browser tabs at the top include 'Google', 'Assignment 7.2', and 'Untitled5.ipynb - Colab'. The address bar shows the Colab URL. The notebook menu bar includes 'File', 'Edit', 'View', 'Insert', 'Runtime', 'Tools', and 'Help'. Below the menu is a search bar and tabs for 'Commands', 'Code', 'Text', and 'Run all'. The main code area contains a Python function `def grade_student(score):` with an `if` statement that returns 'A' for scores  $\geq 70$ , 'B' for scores  $\geq 40$ , and 'C' otherwise. A `print` statement calls the function with a score of 85. The output cell shows the letter 'A'. On the left sidebar, there are icons for file explorer, code editor, and runtime. At the bottom, there is a 'Variables' tab and a 'Terminal' tab. A Gemini AI chat window is open in the bottom right, with a message asking to fix the grading logic and a response explaining the current logic and offering to adjust it to be more standard.

```
[29] def grade_student(score):  
      if score >= 70:  
          return "A"  
      elif score >= 40:  
          return "B"  
      else:  
          return "C"  
  
      print(grade_student(85)) # call the function with an example score and print the result
```

A

fix the if parts where there is grading in the code and make it more logical code to make it give an output

Okay, I understand you want to make the grading logic in the `grade_student` function more standard and logical. Currently, it assigns "A" for scores less than 40 and "C" for scores 70 or above, which seems reversed from typical grading. I'll adjust the conditions to assign grades A, B, and C based on more conventional score ranges, and I'll keep the line that prints the output.

What can I help you build?

Gemini can make mistakes so double-check it and use code with caution. [Learn more](#)

Variables Terminal 9:58 AM Python 3