AI ASSISTED CODING ASSIGNMENT – 7.2

Lab 7:

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

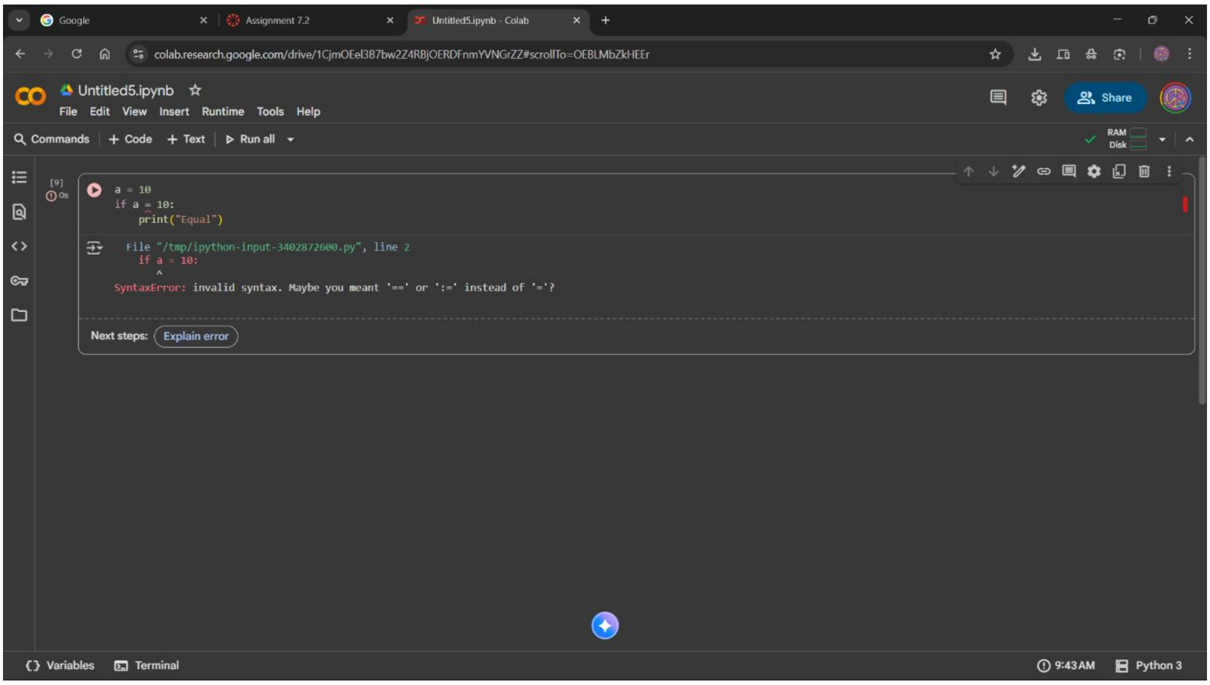
Lab Objectives:

- **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.**

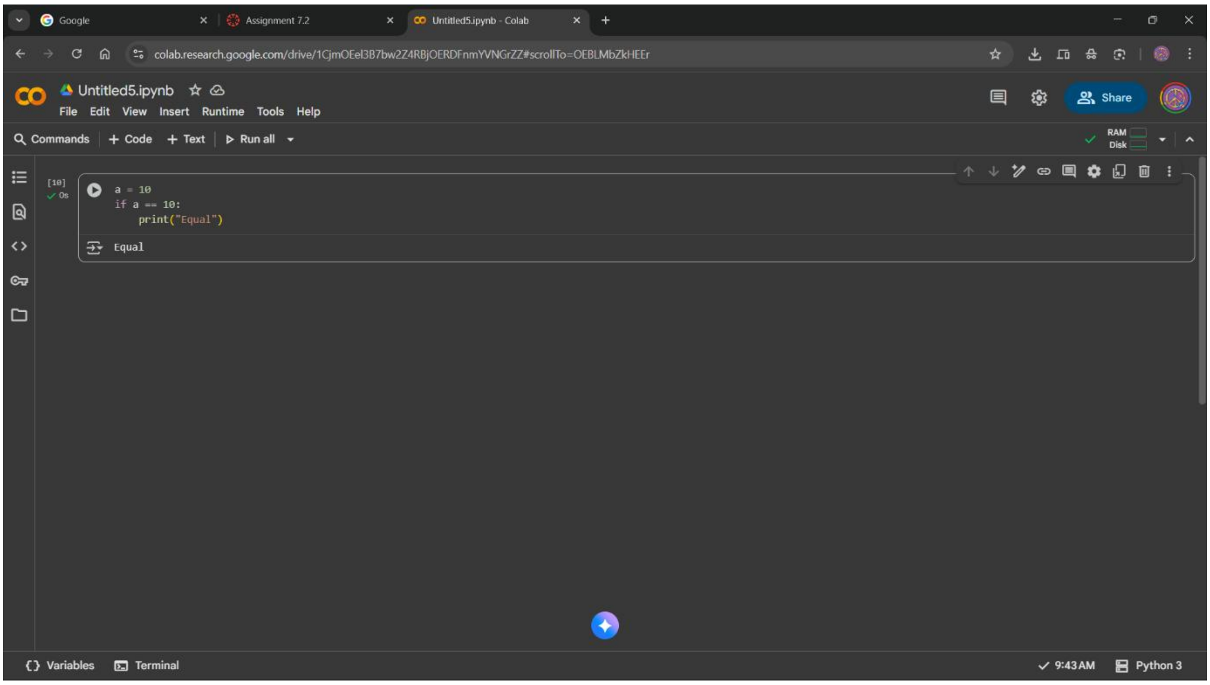
• Task #1 – Syntax Error in Conditionals :



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

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

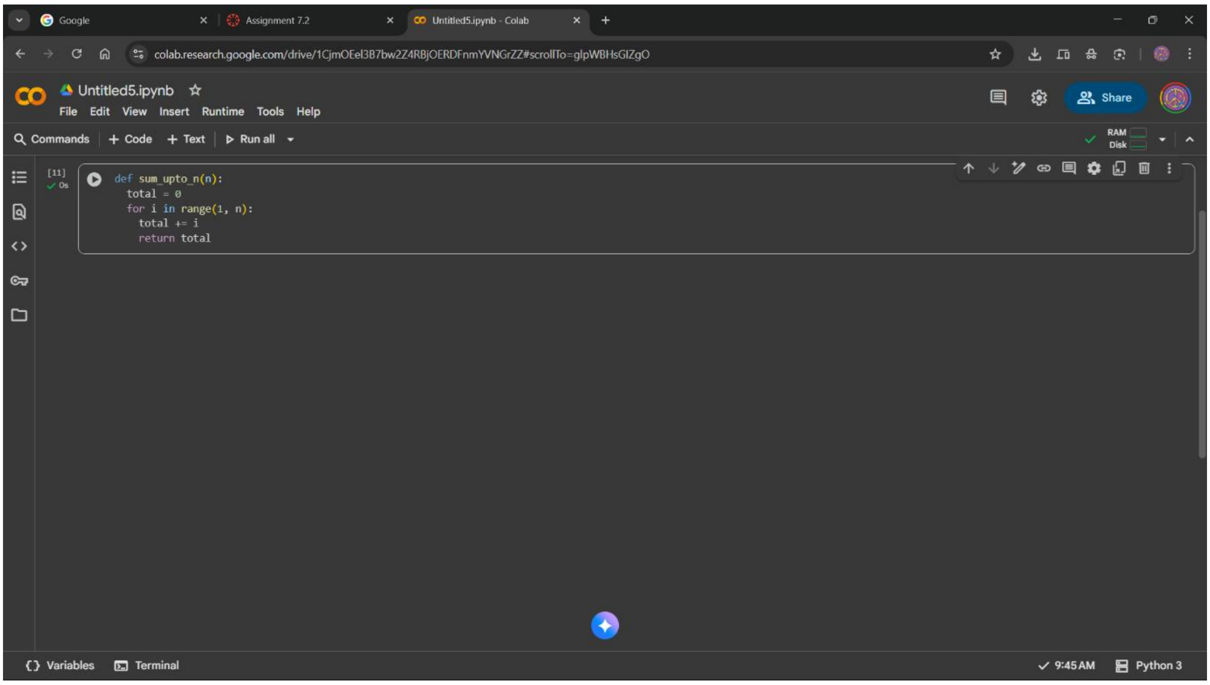
Corrected code and output :



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

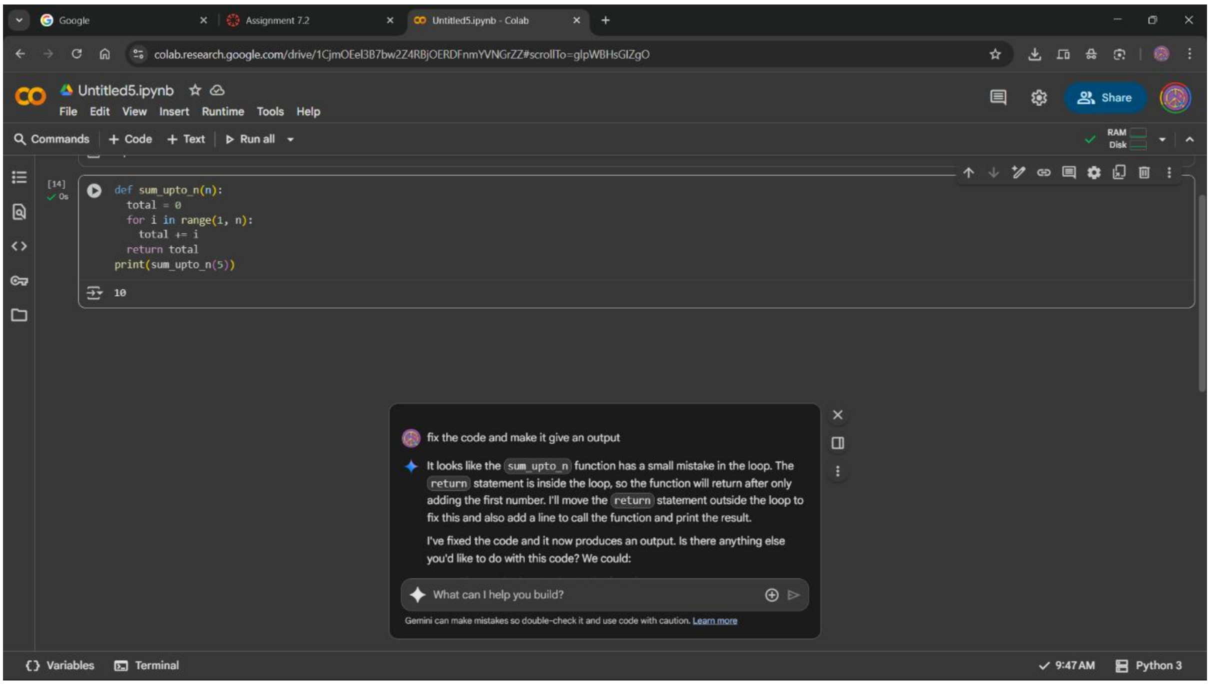
```
Equal
```

• Task #2 – Loop Off-By-One Error :



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

Corrected code and output :



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

```
10
```

fix the code and make it give an output

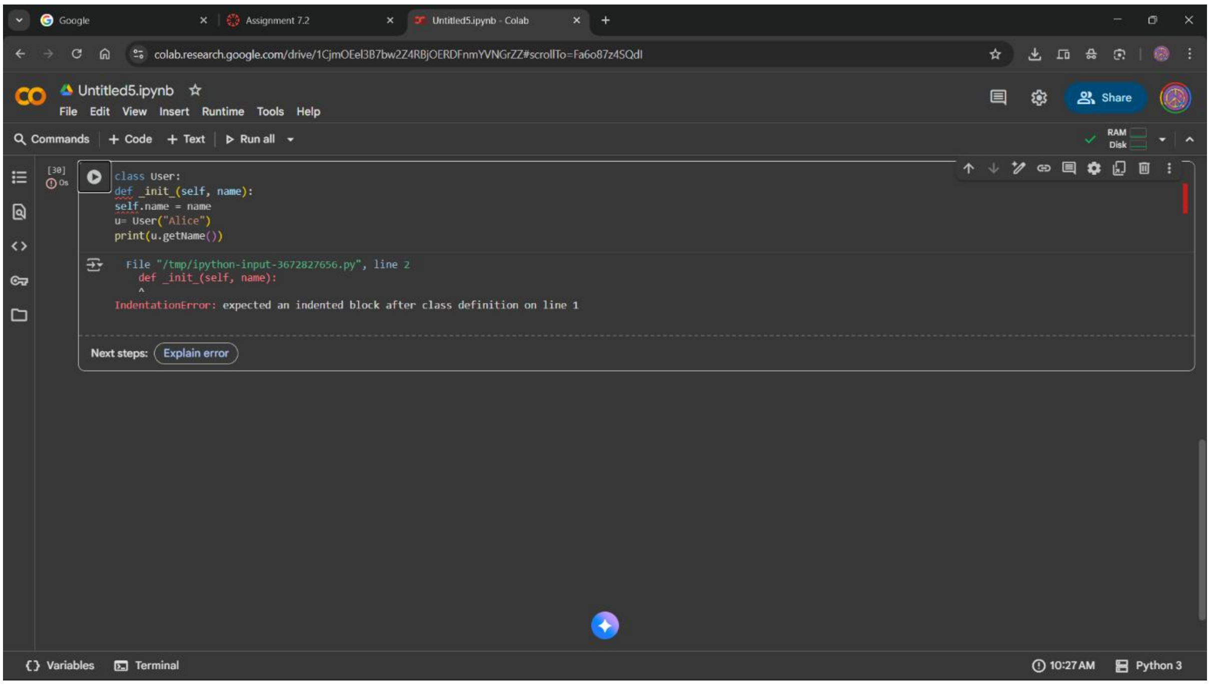
It looks like the `sum_upto_n` function has a small mistake in the loop. The `return` statement is inside the loop, so the function will return after only adding the first number. I'll move the `return` statement outside the loop to fix this and also add a line to call the function and print the result.

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

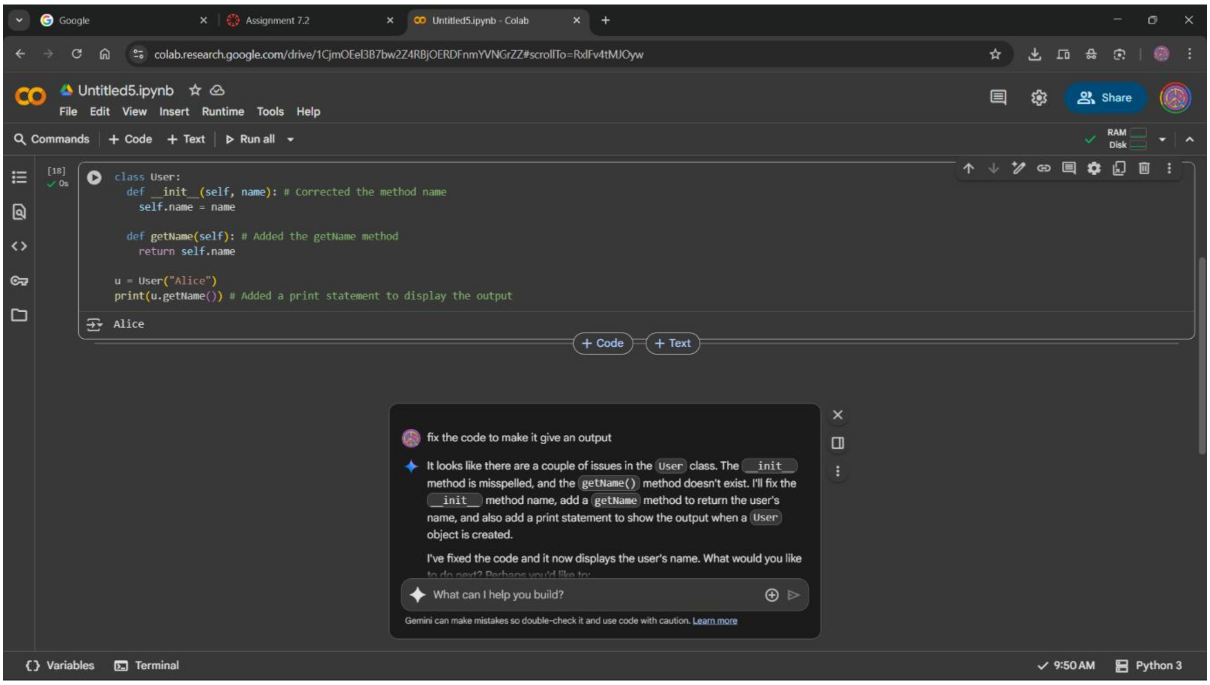
Task #3 – Error : AttributeError :



```
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
```

Corrected code and output :



```
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
```

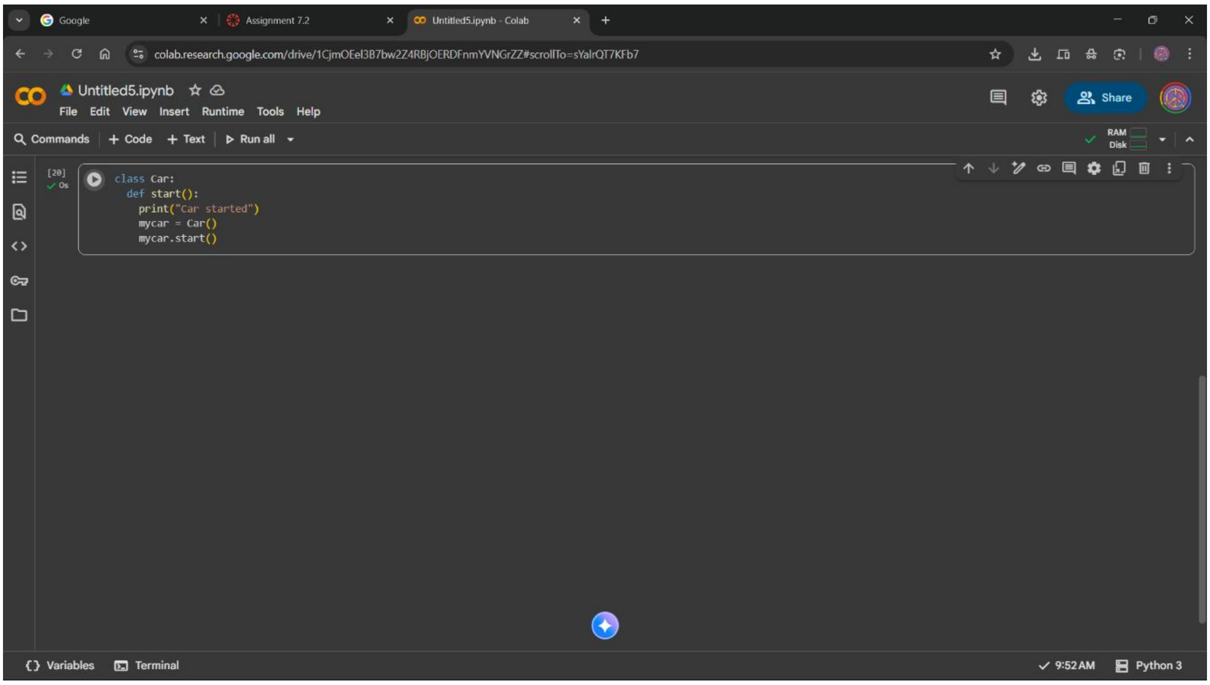
```
Alice
```

fix the code to make it give an output

It looks like there are a couple of issues in the `User` class. The `__init__` method is misspelled, and the `getName()` method doesn't exist. I'll fix the `__init__` method name, add a `getName` method to return the user's name, and also add a print statement to show the output when a `User` object is created.

I've fixed the code and it now displays the user's name. What would you like to do next? Perhaps you'd like to:

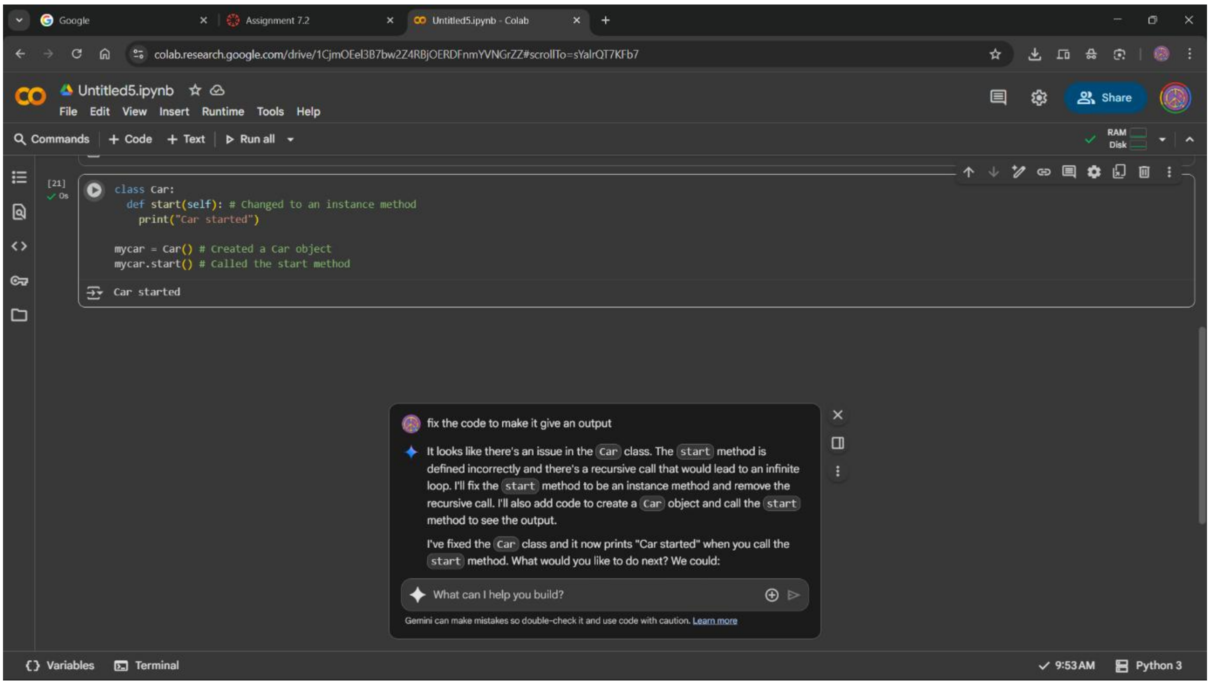
Gemini can make mistakes so double-check it and use code with caution. Learn more



```
class Car:
  def start():
    print("Car started")
    mycar = Car()
    mycar.start()
```

Task #4 – Incorrect Class Attribute Initialization

Corrected code and output :


Google
Assignment 7.2
Untitled5.ipynb - Colab
colab.research.google.com/drive/1CjmOEel3B7bw2Z4RBjOERDFnmYVNGrZZ#scrollTo=sYalrQT7KFb7
Untitled5.ipynb
File Edit View Insert Runtime Tools Help
Share
Commands
+ Code
+ Text
Run all
RAM
Disk
[21]
0s
What can I help you build?
Gemini can make mistakes so double-check it and use code with caution. Learn more
Variables
Terminal
9:53 AM
Python 3

```
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
```

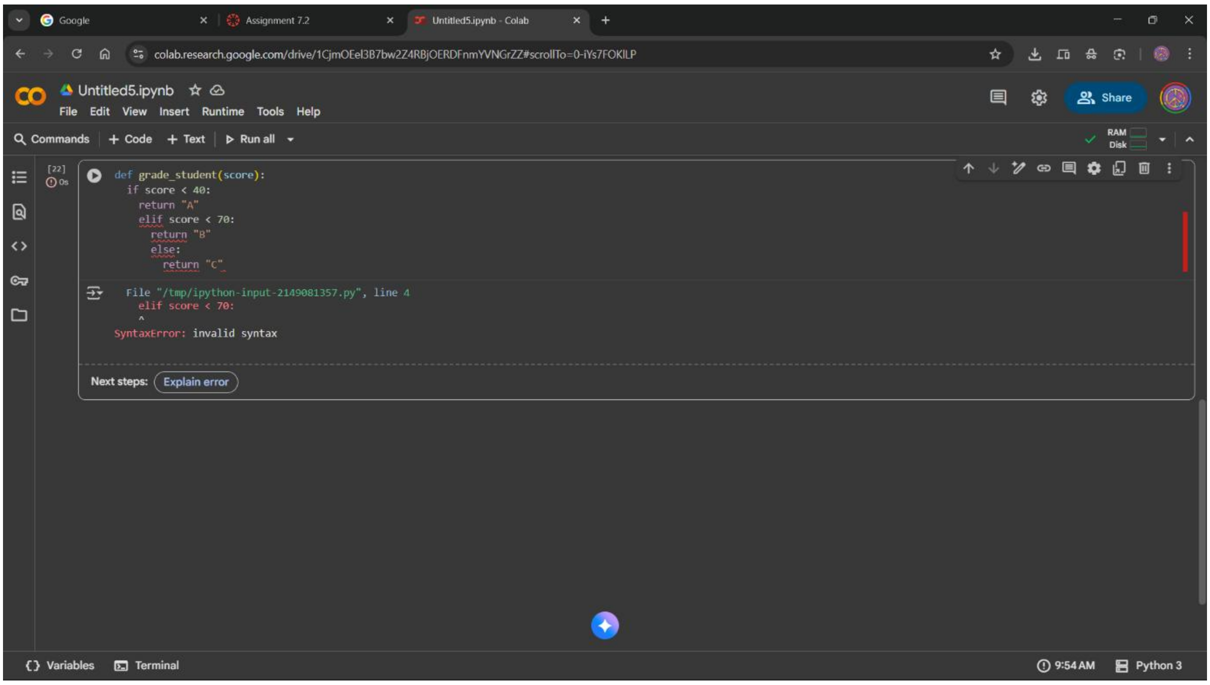
```
Car started
```

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:

Task #5 – Conditional Logic Error in Grading

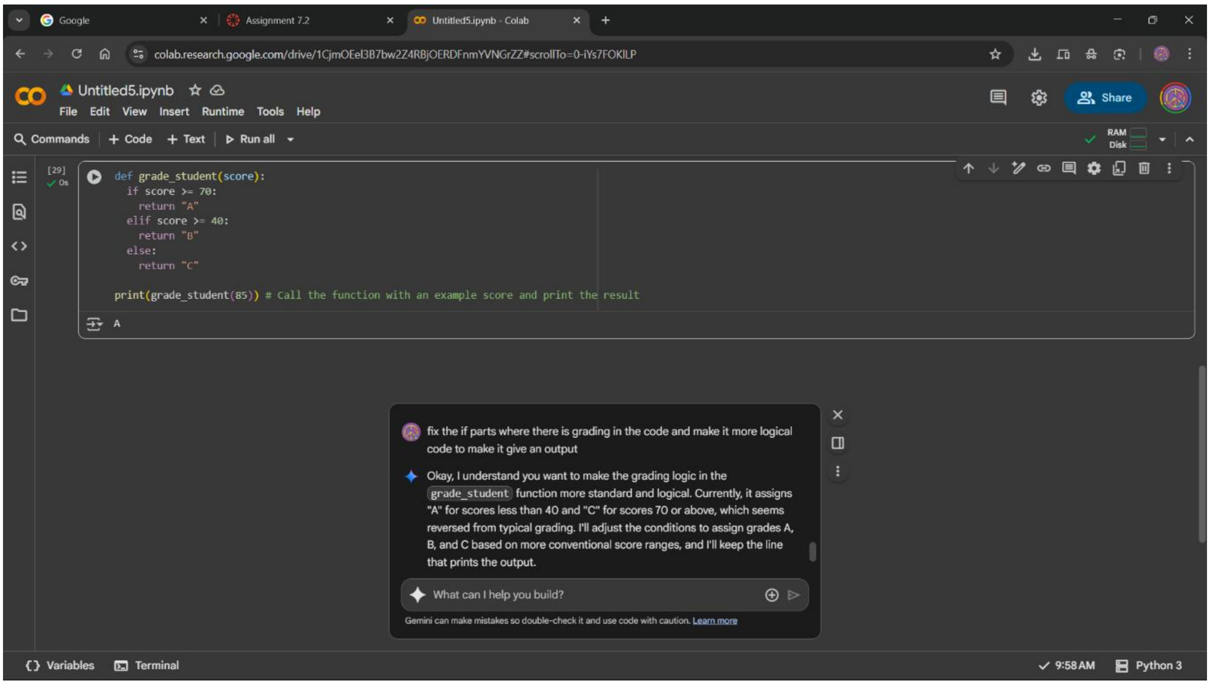


```
def grade_student(score):
  if score < 40:
    return "A"
    elif score < 70:
      return "B"
      else:
        return "C"
```

```
  File "/tmp/ipython-input-2149081357.py", line 4
    elif score < 70:
    ^
SyntaxError: invalid syntax
```

System

Corrected code and Output :



```
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