

Program	:B.tech(CSE)
Specialization	:AIML
Course Title	:AI Assisted Coding
Course Code	:24CS002PC215
Semester	:3 <sup>rd</sup> semester
Academic Session	:2025-2026
Name of Student	:Kurapati Akshitha
Enrollment No.	:2403A52021
Batch No.	:02
Date	:16/09/2025

---

## #LAB ASSIGNMENT

---

### #TASK DESCRIPTION-1:

- Task #1 – Syntax Error in Conditionals

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

### #PROMPT:

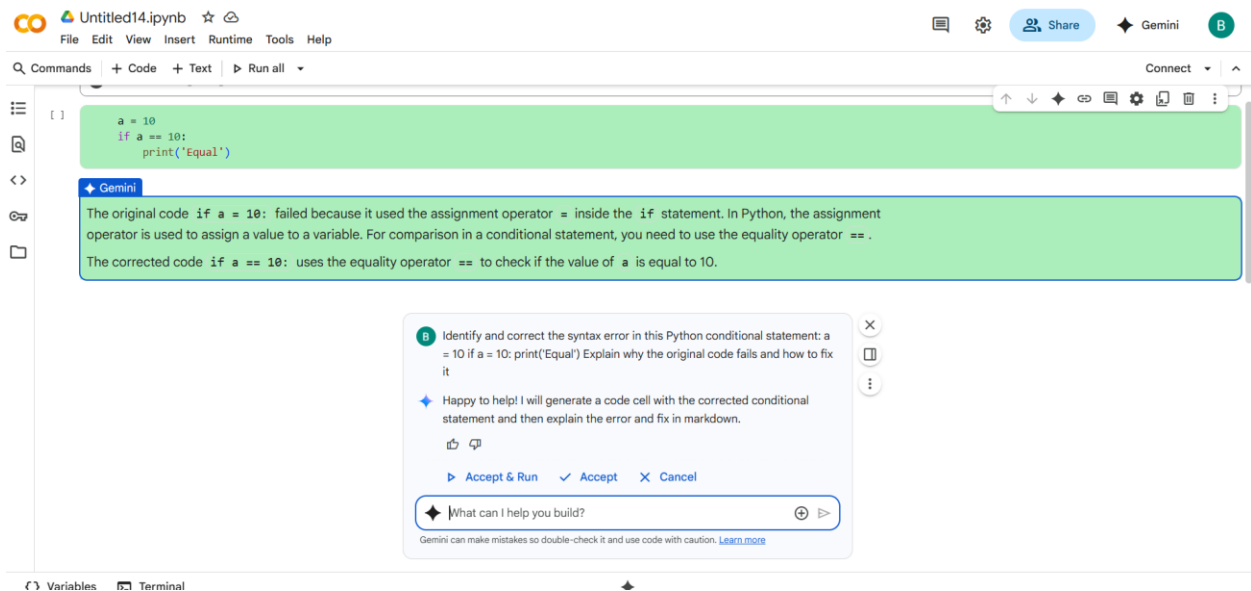
"Identify and correct the syntax error in this Python conditional statement:

CODE:

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

Explain why the original code fails and how to fix it.

## #QUESTION:



The screenshot shows a Jupyter Notebook titled "Untitled14.ipynb". The code cell contains the following Python code:

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

A Gemini AI chat window is open, displaying the following text:

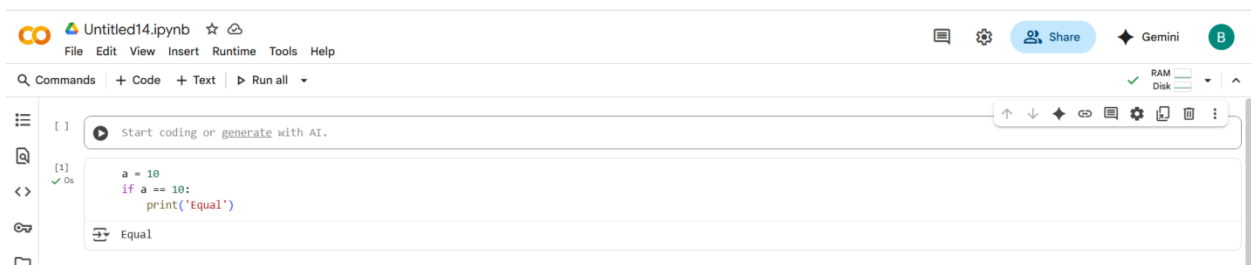
The original code `if a = 10:` failed because it used the assignment operator `=` inside the `if` statement. In Python, the assignment operator is used to assign a value to a variable. For comparison in a conditional statement, you need to use the equality operator `==`.

The corrected code `if a == 10:` uses the equality operator `==` to check if the value of `a` is equal to 10.

Below the chat window, there is a prompt box with the text: "Identify and correct the syntax error in this Python conditional statement: a = 10 if a = 10: print('Equal') Explain why the original code fails and how to fix it". The response from Gemini is: "Happy to help! I will generate a code cell with the corrected conditional statement and then explain the error and fix in markdown."

At the bottom of the chat window, there are buttons: "Accept & Run", "Accept", and "Cancel". There is also a search bar with the text "What can I help you build?" and a "Learn more" link.

## #CODE with OUTPUT:



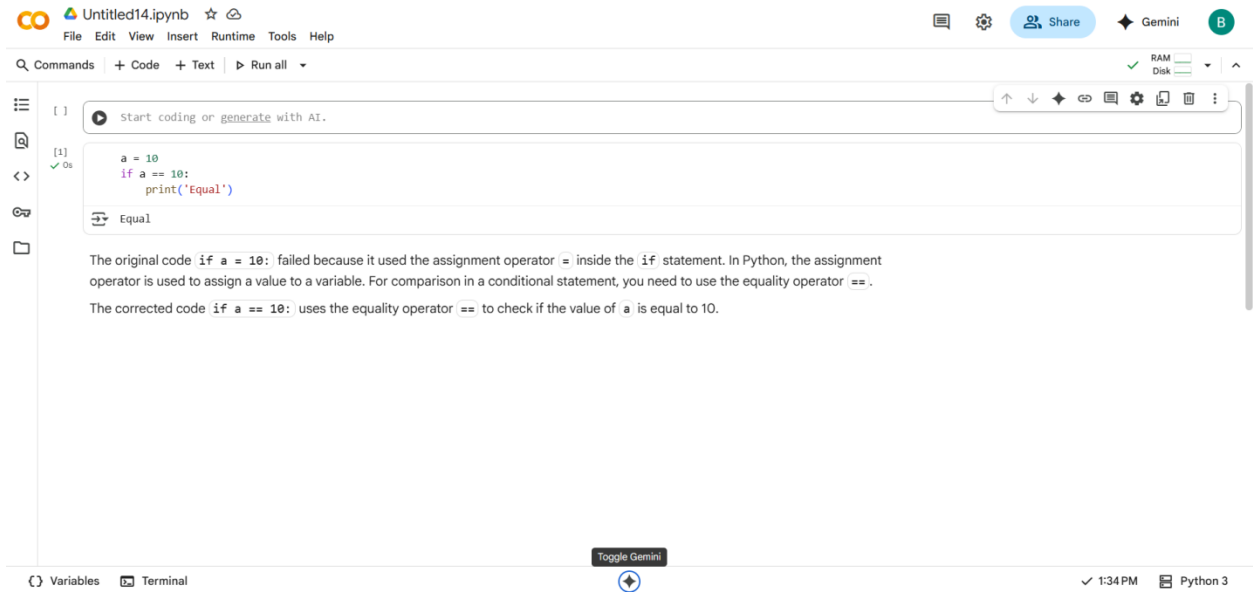
The screenshot shows the same Jupyter Notebook interface. The code cell now contains the corrected code:

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

The output of the code cell is displayed below the code:

```
Equal
```

## #EXPLANATION:



#What I did:

I correct the error without using AI, After giving this code to AI it will also did same correction.

## #TASK DESCRIPTION-2:

- Task #2 – Loop Off-By-One Error.

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

## #PROMPT:

Fix the loop so that this function correctly returns the sum of numbers from 1 to n:

```
def sum_upto_n(n):
```

```
    total = 0
```

```
for i in range(1, n):
```

```
    total += i
```

```
return total
```

## #QUESTION:

The screenshot shows a Jupyter Notebook titled 'Untitled14.ipynb'. The main code cell contains the following Python code:

```
The original code if a = 10: failed because it used the assignment operator = inside the if statement. In Python, the assignment operator is used to assign a value to a variable. For comparison in a conditional statement, you need to use the equality operator ==.
```

```
The corrected code if a == 10: uses the equality operator == to check if the value of a is equal to 10.
```

A Gemini chat window is open, displaying a tip:

**B** Fix the loop so that this function correctly returns the sum of numbers from 1 to n: `def sum_upto_n(n): total = 0 for i in range(1, n): total += i return total`

Working...

What can I help you build?

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

## #CODE with OUTPUT:

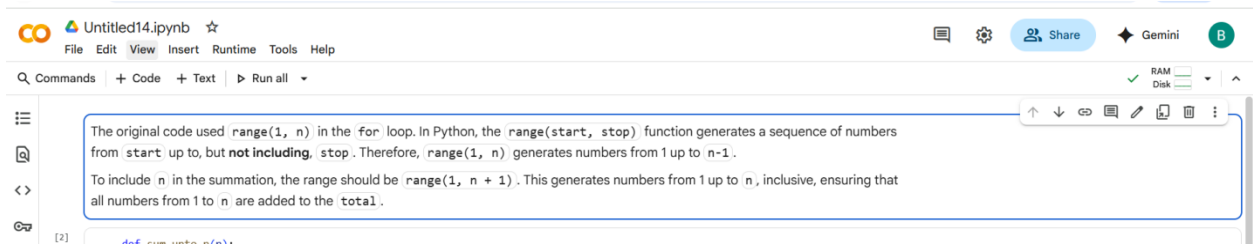
The screenshot shows the same Jupyter Notebook interface. The code cell now contains the corrected function and a test call:

```
[2] ✓ 0s ▶ def sum_upto_n(n):  
    total = 0  
    for i in range(1, n + 1): # Corrected range to include n  
        total += i  
    return total  
  
# Test the function  
print(sum_upto_n(5))
```

The output cell shows the result:

```
15
```

## #EXPLANATION:



## # What I did:

I correct this error in another way AI correction was different compare to me.

## #TASK DESCRIPTION-3:

- Error: AttributeError

```
class User:
    def __init__(self, name):
        self.name = name

u = User("Alice")
print(u.getName())
```

## #PROMPT:

Fix the AttributeError in this Python code:

```
class User:

    def __init__(self, name):

        self.name = name

u = User("Alice")

print(u.getName())
```

The goal is to correctly print the user's name.

# #QUESTION:

The screenshot shows a Jupyter Notebook titled 'Untitled15.ipynb'. The code cell contains the following Python code:

```
class User:
    def __init__(self, name):
        self.name = name

    def getName(self): # Define the method to get the name
        return self.name

u = User("Alice")
print(u.getName())
```

The output of the code is 'Alice'. Below the code cell, there is an 'Explanation of the Error:' section. It states: 'The original code produced an `AttributeError` because you were trying to call a method named `getName()` on an instance of the `User` class, but this method was not defined within the class. The corrected code adds the `getName()` method to the `User` class. This method simply returns the value of the `self.name` attribute, allowing you to access the user's name by calling `u.getName()`.'

Below the explanation, there is a chat window with the following text:

Fix the `AttributeError` in this Python code: `class User: def __init__(self, name): self.name = name u = User("Alice") print(u.getName())` The goal is to correctly print the user's name, explain the error which we solve

Sure! I can help you fix the `AttributeError` and explain the error.

Great! The `AttributeError` is resolved.

Would you like to explore more about:

- Adding more methods to the `User` class?
- Creating other classes and objects?

What can I help you build?

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

# #CODE with OUTPUT:

The screenshot shows the same Jupyter Notebook interface as the first screenshot, but without the AI-generated explanation. The code cell contains the same Python code:

```
class User:
    def __init__(self, name):
        self.name = name

    def getName(self): # Define the method to get the name
        return self.name

u = User("Alice")
print(u.getName())
```

The output of the code is 'Alice'.

# #EXPLANATION:

The screenshot shows the same Jupyter Notebook interface as the first screenshot, but without the AI-generated explanation. The code cell contains the same Python code:

```
class User:
    def __init__(self, name):
        self.name = name

    def getName(self): # Define the method to get the name
        return self.name

u = User("Alice")
print(u.getName())
```

The output of the code is 'Alice'.

# What I did:

Actually I am facing little bit of problem correction of this error finally I do almost correct code ,but AI did very well.

#### #TASK DESCRIPTION-4:

- **Incorrect Class Attribute Initialization**

```
class Car:
    def start():
        print("Car started")

mycar = Car()
mycar.start()
```

#### #PROMPT:

Fix the method definition so this Python class works correctly:

```
class Car:

    def start():

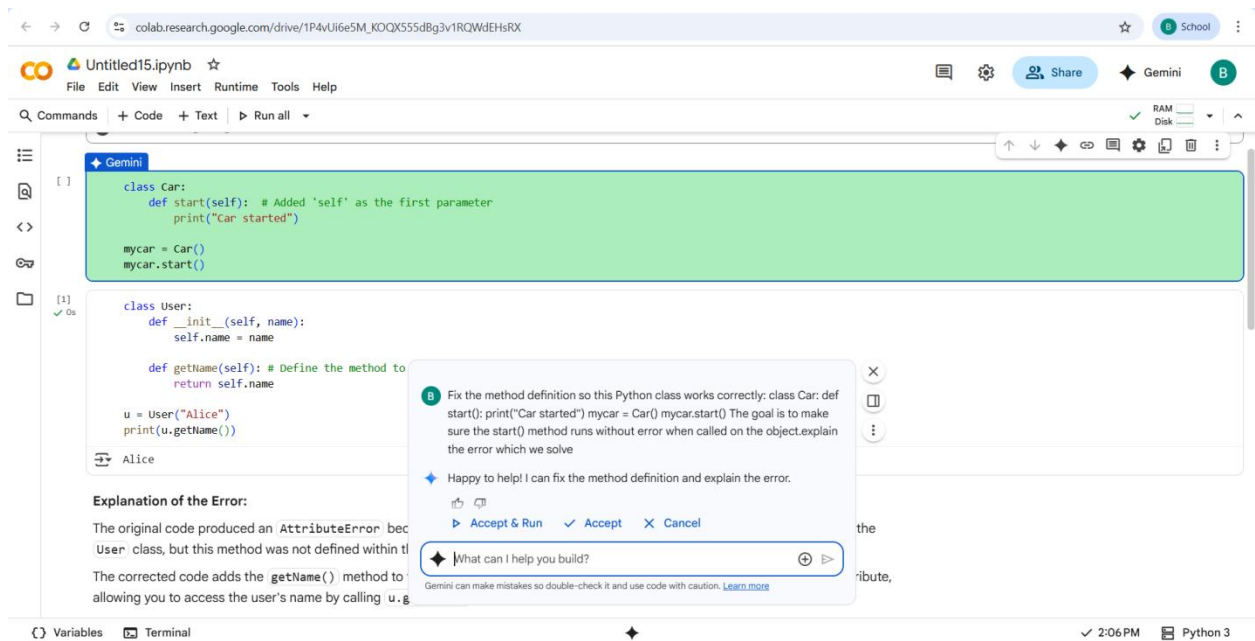
        print("Car started")
```

```
mycar = Car()
```

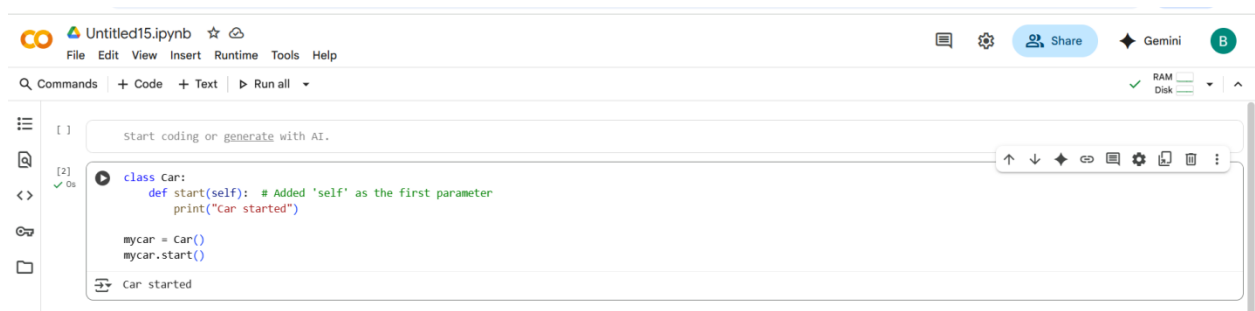
```
mycar.start()
```

The goal is to make sure the `start()` method runs without error when called on the object.

#### #QUESTION:

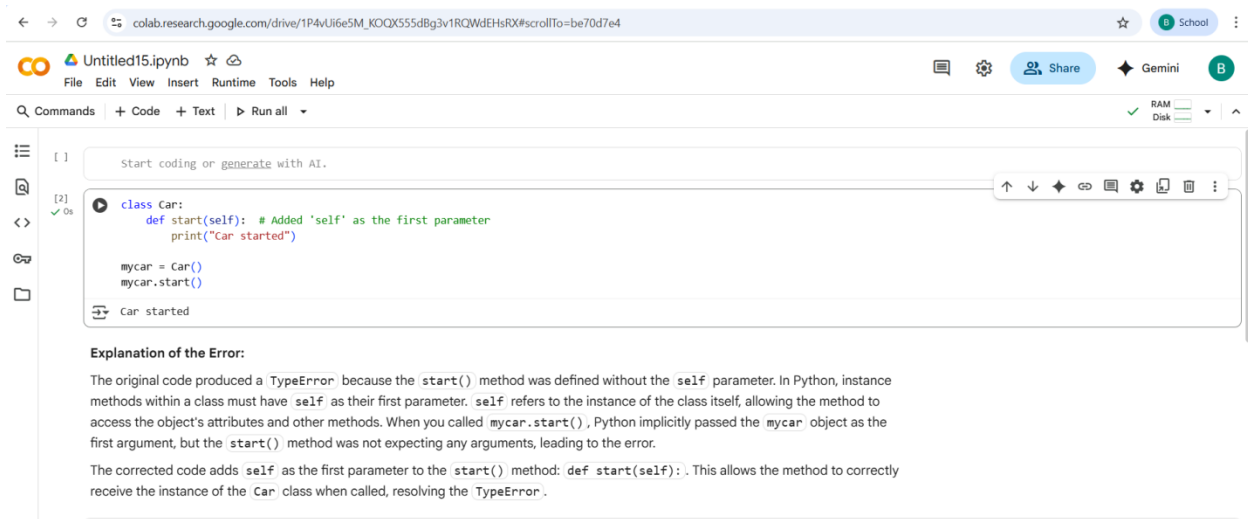


#CODE with OUTPUT:



#EXPLANATION:





# What I did:

I correct this error without using AI , after giving this code to AI it will also do same thing.

## #TASK DESCRIPTION-5:

- Conditional Logic Error in Grading System

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

#PROMPT:

Fix the grading logic in this function so that it assigns correct letter grades based on score:

```
def grade_student(score):
```

```
    if score < 40:
```

```
        return "A"
```

```
    elif score < 70:
```

```
        return "B"
```

```
    else:
```

```
        return "C"
```

The goal is to make sure high scores get higher grades and low scores get lower grades.

## #QUESTION:

The screenshot shows a Jupyter Notebook titled 'Untitled15.ipynb'. The code in the cell is as follows:

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

# Test cases
print(grade_student(85)) # Expected: A
print(grade_student(55)) # Expected: B
print(grade_student(30)) # Expected: C
```

A Gemini AI assistant is providing an explanation of the error and a corrected version of the code. The explanation states: 'The original code assigned grades in reverse order. A score of 85 received a "C". This is the opposite of how grading works. The corrected code reverses the logic. It first checks if the score is greater than or equal to 70 and assigns a "A". The next range (score >= 40) and assigns a "B". Finally, scores below 40 get a "C".'

The corrected code is:

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

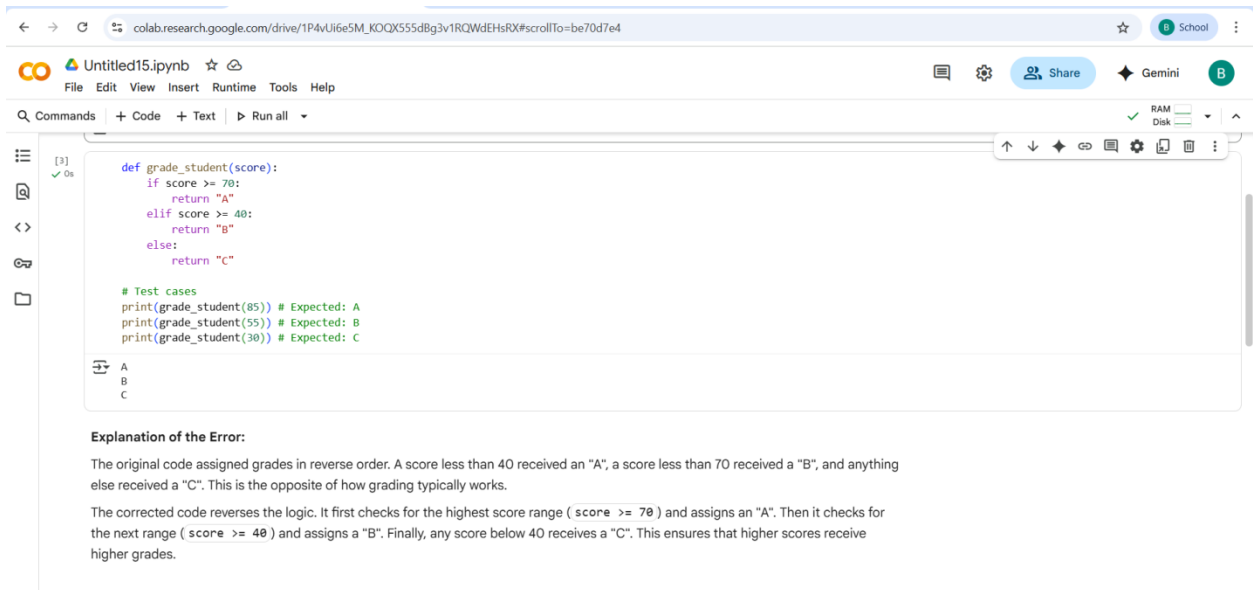
The Gemini assistant also provides a prompt: 'What can I help you build?' with a search icon and a 'Go' button.

## #CODE with OUTPUT:

The screenshot shows the same Jupyter Notebook with the corrected code. The output of the code is displayed below the cell:

```
A
B
C
```

# #EXPLANATION:



```
def grade_student(score):  
    if score >= 70:  
        return "A"  
    elif score >= 40:  
        return "B"  
    else:  
        return "C"  
  
# Test cases  
print(grade_student(85)) # Expected: A  
print(grade_student(55)) # Expected: B  
print(grade_student(30)) # Expected: C
```

**Explanation of the Error:**

The original code assigned grades in reverse order. A score less than 40 received an "A", a score less than 70 received a "B", and anything else received a "C". This is the opposite of how grading typically works.

The corrected code reverses the logic. It first checks for the highest score range ( `score >= 70` ) and assigns an "A". Then it checks for the next range ( `score >= 40` ) and assigns a "B". Finally, any score below 40 receives a "C". This ensures that higher scores receive higher grades.

## # What I did:

I corrected this error without using the AI , after giving to AI some differences compare to AI code and My code.

- THANK YOU-

