AI-Assisted Coding

Name:Peravena Navadeep

Htno:2403A52019

Task Description#1

- Task #1 – Syntax Error in Conditionals a=10 if a=10:

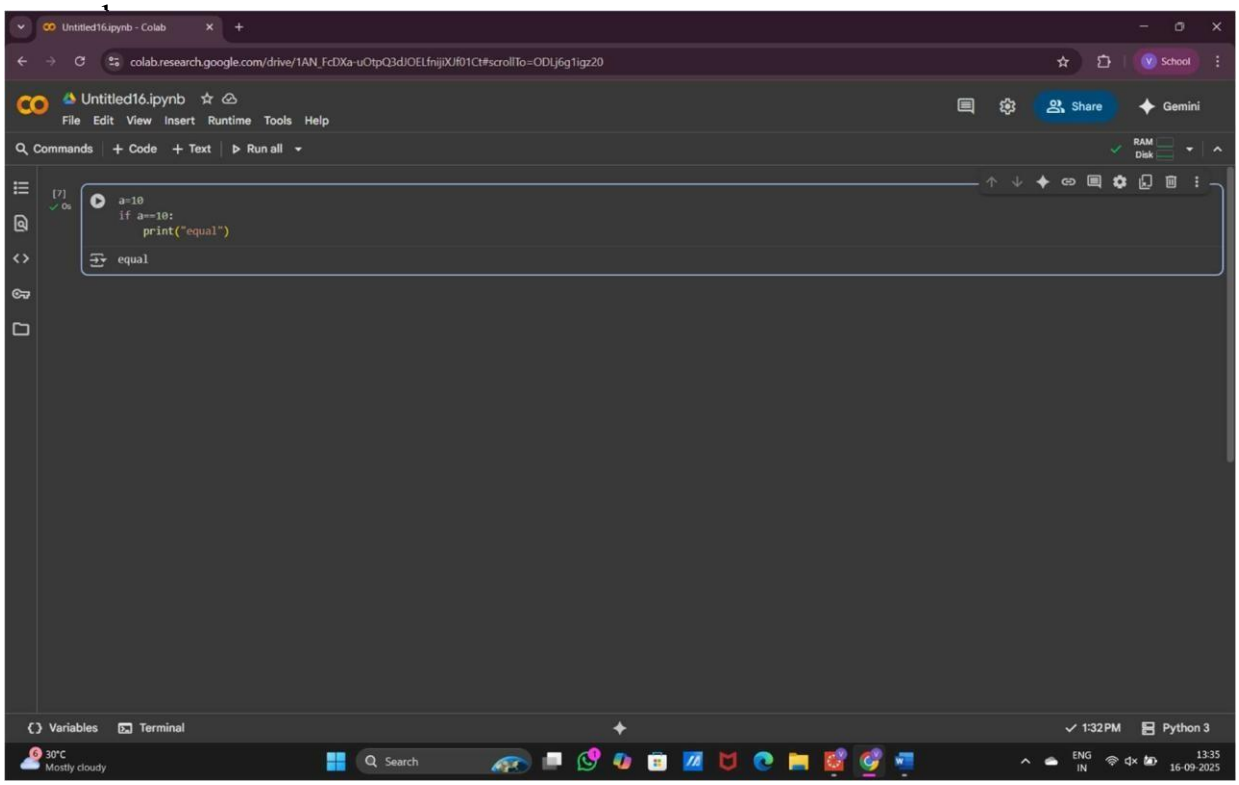
print(“equal’)

Expected Output#1

- Corrected function with syntax fix

PROMPT:

a=10 if a=10: print("equal").fix this code without any errors and give the



OBSERVATION:

This code Is run successfully using the assignment operator = inside an if statement condition. In Python, to check for equality, you should use the comparison operator ==.I will fixed this error by changing = to == in the if statement.

Task Description#2

- Task #2 – Loop O -By-One Error.

Def_sum_upto_n(n):

Total=0

For i in range(1,n):

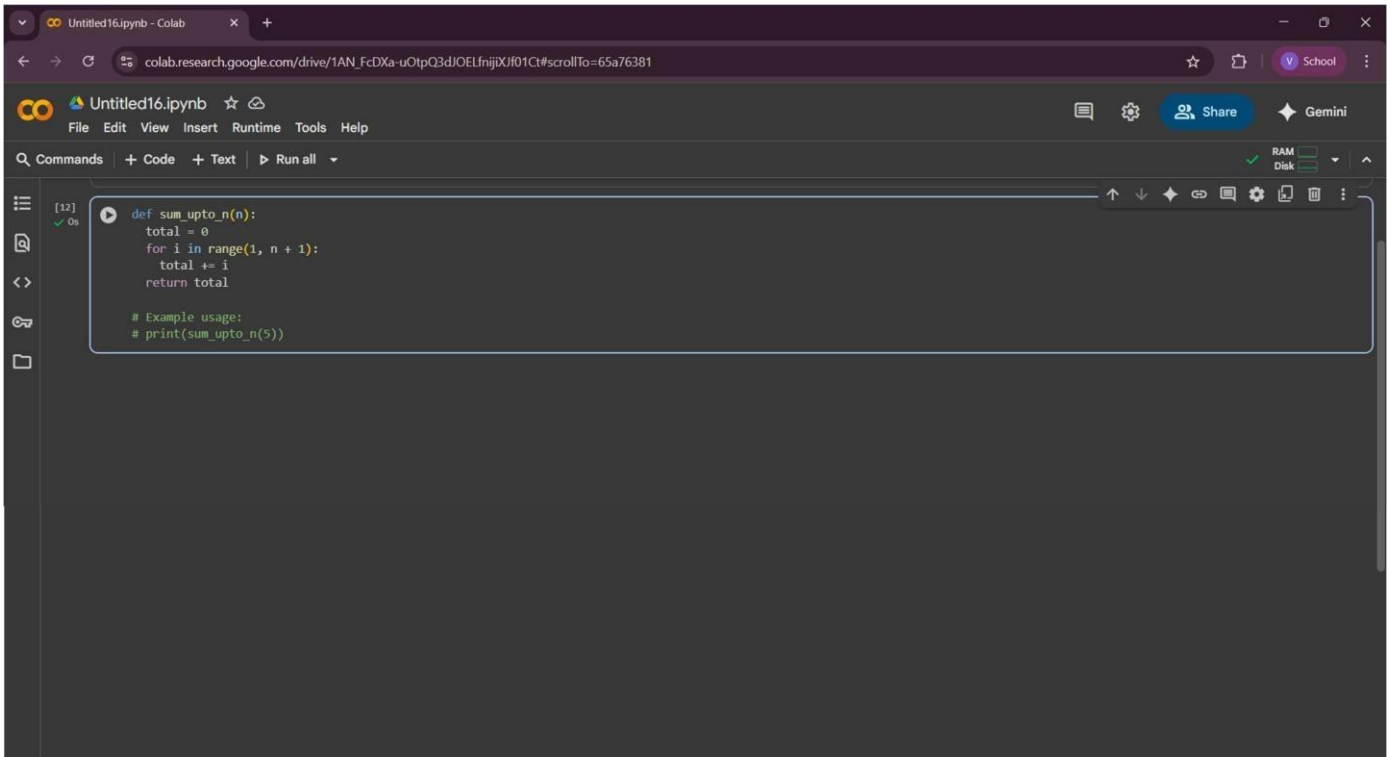
Total +=i Return
total

Expected Output#2

- AI fixes increment/decrement error

PROMPT:

Def_sum_upto_n(n): Total=0 For i in range(1,n): Total +=i Return total.Fix this code without any errors.



OBSERVATION:

This code defines a function called sum_upto_n that takes one input, n. It calculates the sum of all whole numbers starting from 1 up to and including n. It does this by starting a total at 0, then looping through each number from 1 to n and adding it to the total. Finally, it gives back the final calculated total.

Task Description#3

- Error: AttributeError Class user:

```
Def __init__(self,name):

  Self.name=name

U=user(“Alice”)

Print(u.getName())
```

Expected Output#3

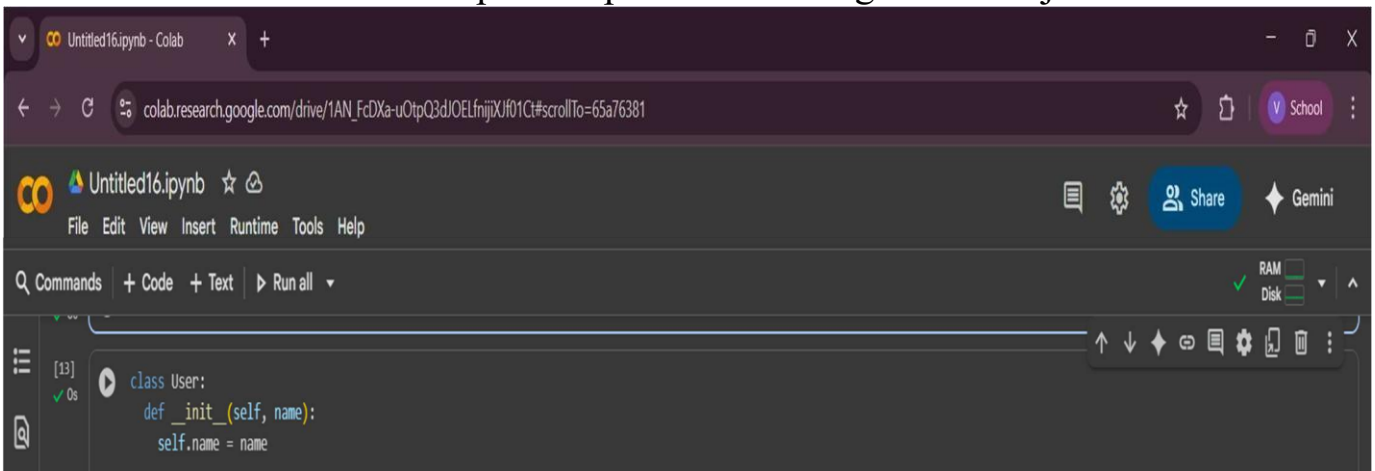
- Identify the missing method and correct the code.

PROMPT:

Class user: Def __init__(self,name): Self.name=name
U=user(“Alice”) Print(u.getName()) fix this code without any errors

OBSERVATION:

This code defines a simple blueprint for creating "User" objects.



When you create a User object, you give it a name.
The __init__ method is like a setup process that stores this name inside the object. After creating a User object named u with the name "Alice", the code then prints out the name that's stored within that u object, which is "Alice".

Task Description#4

Incorrect Class Attribute

Initialization class car: def start():

print(“car started”) mycar =car()

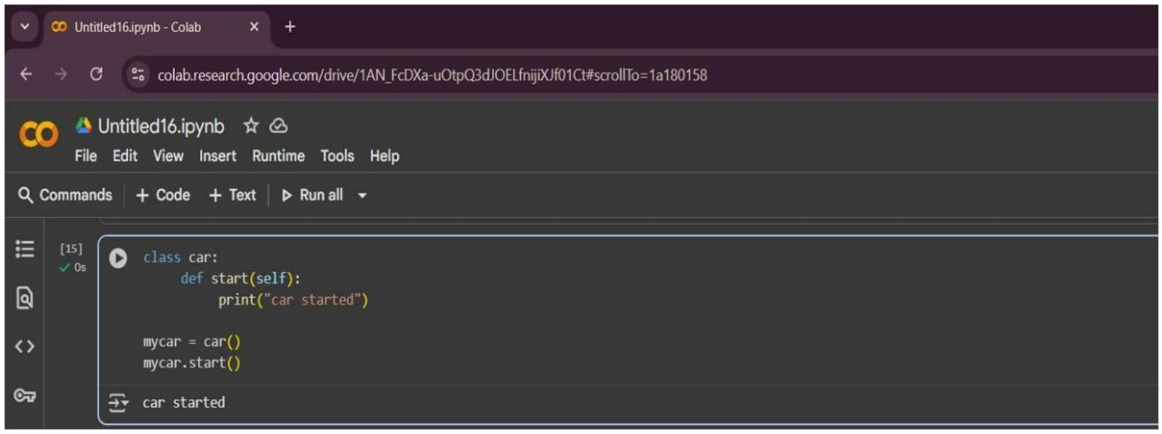
mycar.start()

Expected Output#4

• Detect missing self and initialize attributes properly.

PROMPT:

class car: def start(): print(“car started”) mycar =car() mycar.start() fix this code without any errors



OBSERVATION:

This code defines a basic blueprint for a car. Inside this blueprint, there's a function called start. When you create a specific car based on this blueprint (like mycar), you can then tell that specific car to

start(). The start function simply prints "car started". So, when you run this code, it creates a car object and then tells it to start, resulting in "car started" being printed.

Task Description#5

- Conditional Logic Error in Grading System Def grade_student(score):

If score < 40:

Return "A" If

score < 70:

Return "B"

Else:

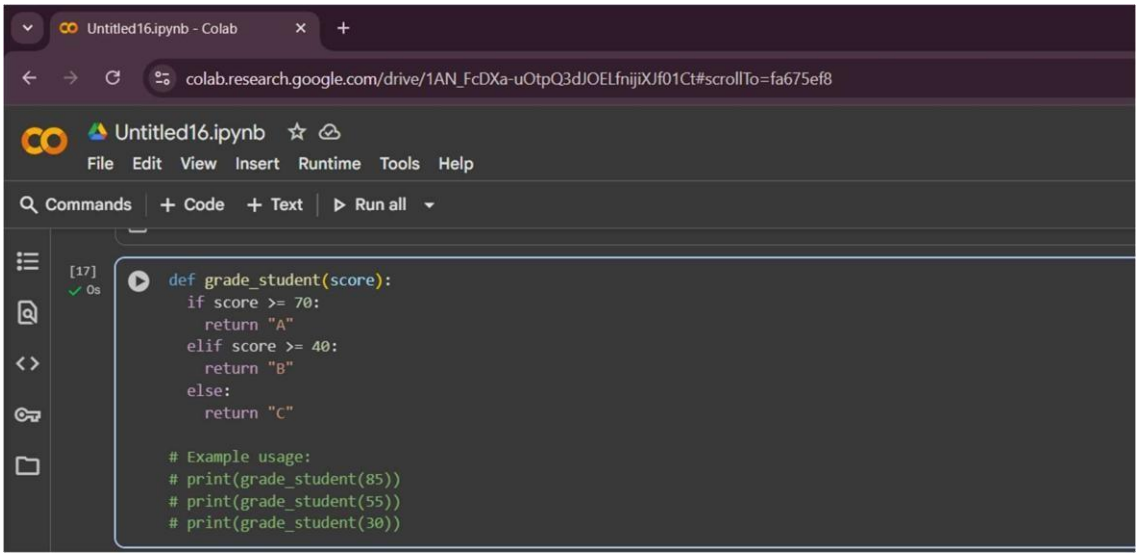
Return "c"

Expected Output#5

- Detect illogical grading and correct the grade levels

PROMPT:

Def grade_student(score): If score < 40: Return "A" If score < 70: Return "B" Else: Return "c" fix this code without any errors



OBSERVATION:

This function grade_student takes a student's score as input and figures out their letter grade based on that score. It first checks if the score is 70 or higher; if it is, they get an "A". If not, it then checks if the score is 40 or higher; if it is, they get a "B". If the score isn't 70 or above and isn't 40 or above, then it must be below 40, and in that case, the student gets a "C". So, it's a straightforward way to assign grades based on di erent score ranges.