ROLL NO:2403A51185
BATCH NO:09

SCHOOL OF COMPUTER SCIENCE AND ARTIFICIAL INTELLIGENCE		DEPARTMENT OF COMPUTER SCIENCE ENGINEERING	
ProgramName:B. Tech		**Assignment Type: Lab**	**AcademicYear:**2025-2026
CourseCoordinatorName		Venkataramana Veeramsetty	
Instructor(s)Name		Dr. V. Venkataramana (Co-ordinator) Dr. T. Sampath Kumar Dr. Pramoda Patro Dr. Brij Kishor Tiwari Dr.J.Ravichander Dr. Mohammand Ali Shaik Dr. Anirodh Kumar Mr. S.Naresh Kumar Dr. RAJESH VELPULA Mr. Kundhan Kumar Ms. Ch.Rajitha Mr. M Prakash Mr. B.Raju Intern 1 (Dharma teja) Intern 2 (Sai Prasad) Intern 3 (Sowmya) NS_2 (Mounika)	
CourseCode	24CS002PC215	**CourseTitle**	AI Assisted Coding
Year/Sem	II/I	**Regulation**	R24
Date and Day of Assignment	Week4 - Tuesday	**Time(s)**	
Duration	2 Hours	**Applicableto Batches**	
AssignmentNumber:7.2(Present assignment number)/**24**(Total number of assignments)			

Q.No.	Question	*ExpectedTime to complete*
1	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.	Week4 - Wednesday

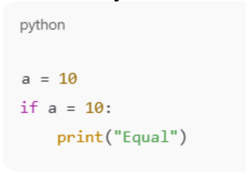
- To evaluate how AI explains, detects, and fixes different types of coding errors.
- To build confidence in using AI to perform structured debugging practices.

Lab Outcomes (LOs):
After completing this lab, students will be able to:

- 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.
- Refactor buggy code using responsible and reliable programming patterns.

Task Description#1

- Task #1 – Syntax Error in Conditionals



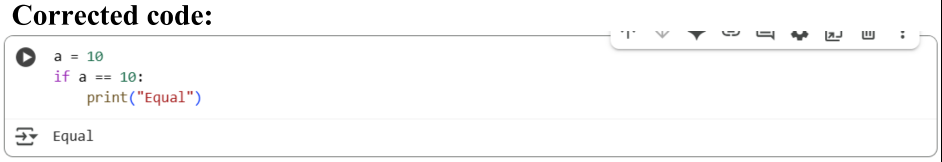
python

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

Expected Output#1

- Corrected function with syntax fix

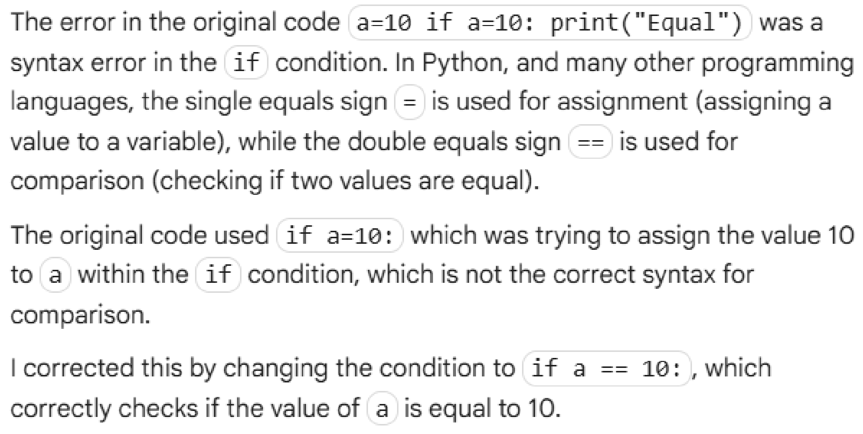
Corrected code:



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

```
Equal
```

Explanation:



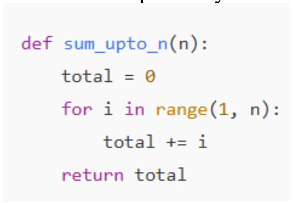
The error in the original code `a=10 if a=10: print("Equal")` was a syntax error in the `if` condition. In Python, and many other programming languages, the single equals sign `=` is used for assignment (assigning a value to a variable), while the double equals sign `==` is used for comparison (checking if two values are equal).

The original code used `if a=10:` which was trying to assign the value 10 to `a` within the `if` condition, which is not the correct syntax for comparison.

I corrected this by changing the condition to `if a == 10:`, which correctly checks if the value of `a` is equal to 10.

Task Description#2 (Loops)

- Task #2 – Loop Off-By-One Error.

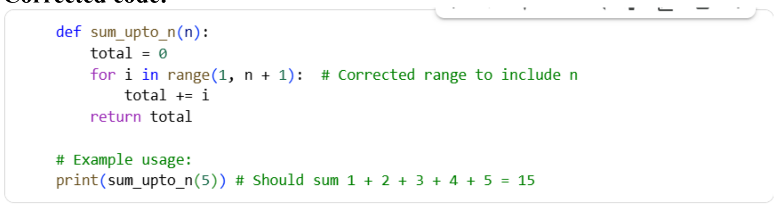


```python
def sum_upto_n(n):
    total = 0
    for i in range(1, n):
        total += i
    return total
```

Expected Output#2

- AI fixes increment/decrement error

Corrected code:



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

# Example usage:
print(sum_upto_n(5)) # Should sum 1 + 2 + 3 + 4 + 5 = 15
```

Explanation:
The error in the sum_upto_n function was an off-by-one error in the loop's range. The original code used range (1, n), which includes the starting number 1 but excludes the ending number n. This meant the loop summed numbers from 1 up to n-1, missing n itself. To fix this, I changed the range to range (1, n + 1). This new range includes 1 and goes up to (n + 1) - 1, which is n. The loop now correctly iterates from 1 to n inclusive, calculating the intended sum.

Task Description#3

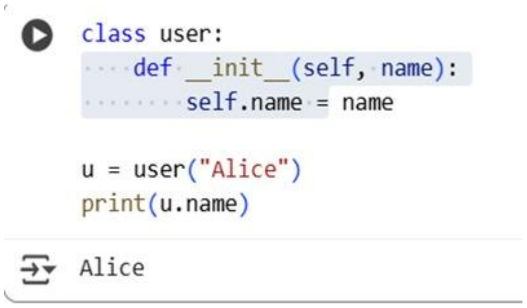
- Error: AttributeError

```
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.

Corrected Error:



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

u = user("Alice")
print(u.name)
```

```
Alice
```

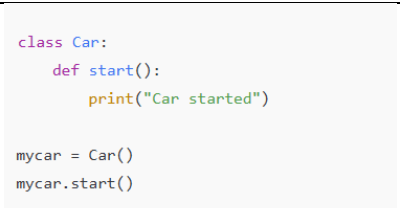
Explanation:
You tried to use a method called getName() on your user object, but there was no method with that name defined in your user class. It's like asking someone to do something they don't know how to do.

The Correction: You need to either:

Add a getName() method to your user class that tells it how to give you the name.
Directly ask for the name using u.name, because the name is a piece of information the user object already has and is willing to share directly.

Task Description#4

- **Incorrect Class Attribute Initialization**

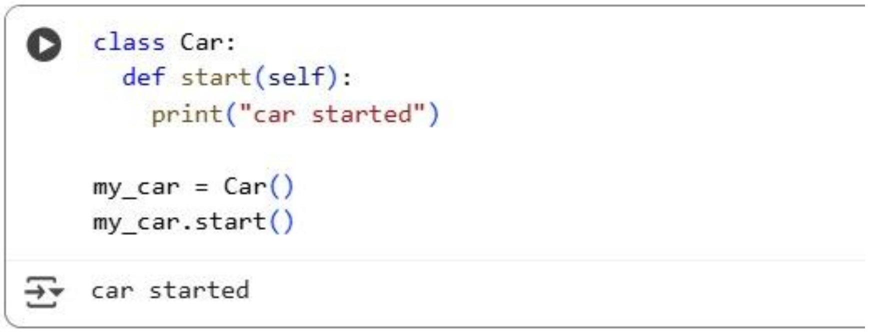


```
class Car:
    def start():
        print("Car started")

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

Expected Output#4

- Detect missing self and initialize attributes properly.

Corrected code:



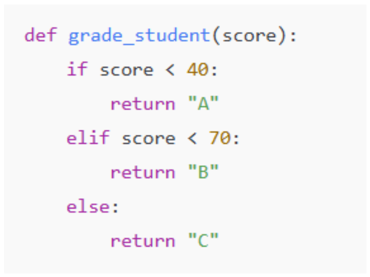
```
class Car:
  def start(self):
    print("car started")

my_car = Car()
my_car.start()
```

```
car started
```

Explanation:
The error in the code was a Type Error because the start method in the Car class was missing the required self parameter.
I added self as the first parameter to the start method definition:
def start(self):
  print("car started")
This makes the method a proper instance method that can be called on objects created from the Car class.

Task Description#5

- Conditional Logic Error in Grading System

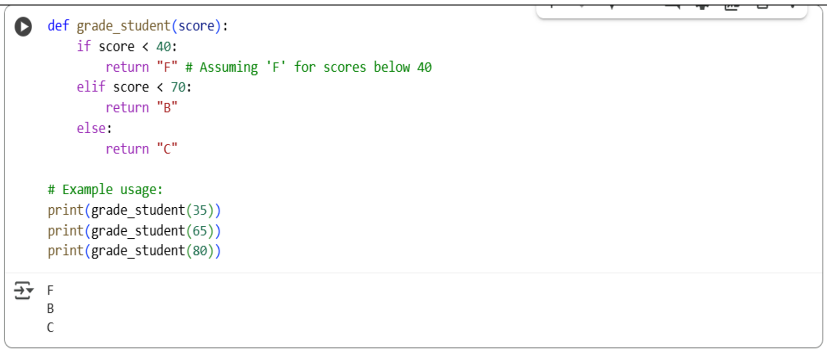


```
def grade_student(score):
    if score < 40:
        return "A"
    elif score < 70:
        return "B"
    else:
        return "C"
```

Expected Output#5

- Detect illogical grading and correct the grade levels.

Corrected code:



```
def grade_student(score):
    if score < 40:
        return "F" # Assuming 'F' for scores below 40
    elif score < 70:
        return "B"
    else:
        return "C"

# Example usage:
print(grade_student(35))
print(grade_student(65))
print(grade_student(80))
```

```
F
B
C
```

Explanation:

The logical error in the original `grade_student` function was in the order and conditions for assigning grades. It assigned "A" for scores below 40, then "B" for scores below 70, which overlapped and could cause confusion for scores below 40. My correction addresses this by using a more standard grading scale. Scores below 40 now explicitly receive an "F". Scores between 40 and 69 receive a "B", and scores 70 and above receive a "C". This revised logic correctly assigns grades based on distinct score ranges, making the function's behavior predictable and accurate according to the intended grading system.

Note: Report should be submitted a word document for all tasks in a single document with prompts, comments & code explanation, and output and if required, screenshots

Evaluation Criteria:

Criteria	**Max Marks**
Task #1	0.5
Task #2	0.5
Task #3	0.5
Task #4	0.5
Task #5	0.5
Total	**2.5 Marks**