

# Assignment:7.1

-Ai Assistant Cording

Name:- Aashutosh Kumar

Hall Tkt:2403a51316

Task Description#1:- Provide a Python snippet with a missing parenthesis in a print statement(e.g., print “Hello”). Use AI to detect and fix the syntax error.

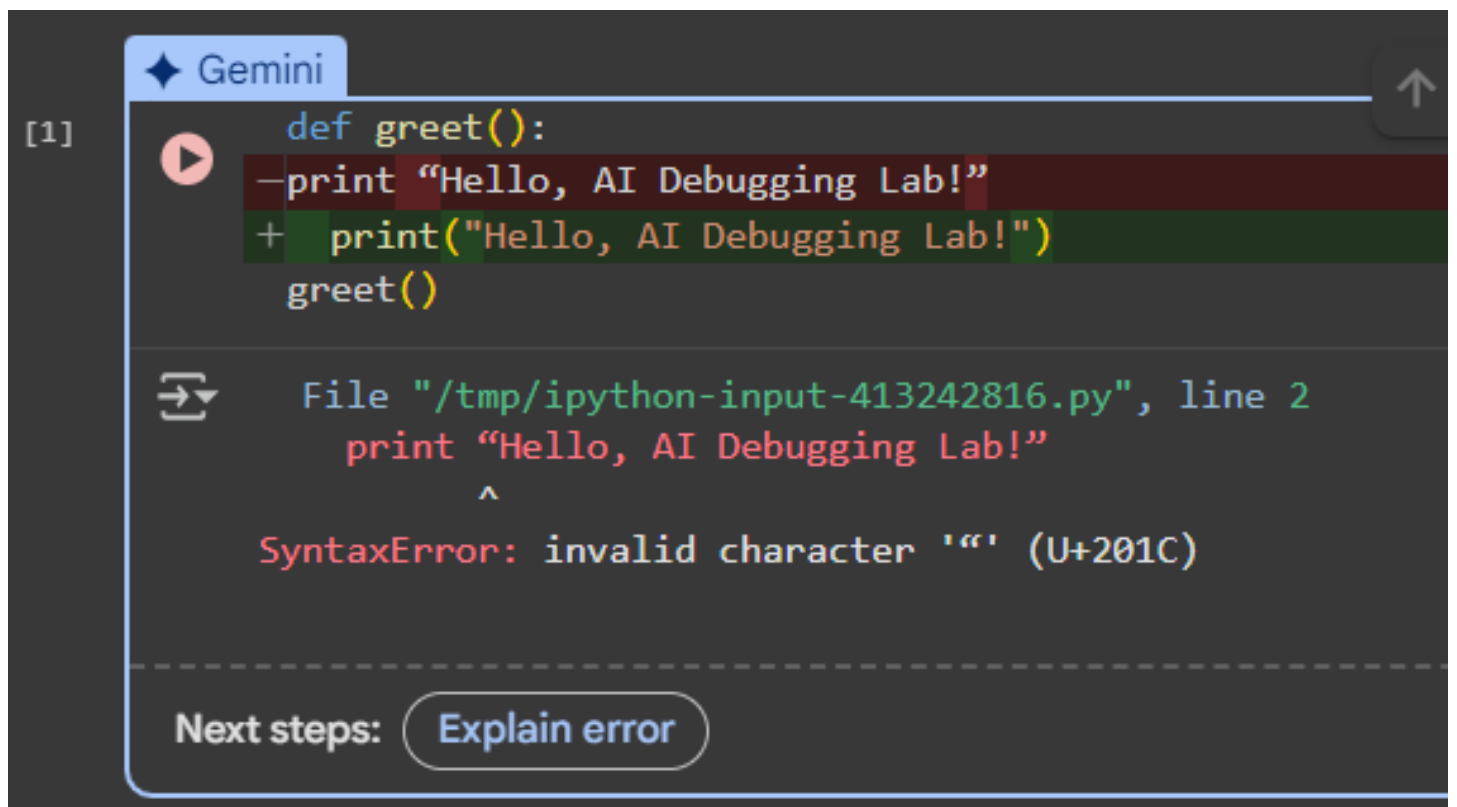
#Bug: Missing parentheses in print statement

```
def greet():  
  
    print “Hello, AI Debugging Lab!”  
  
greet()
```

Requirements:

- Run the code to observe the error.
- Apply AI Suggestions to correct the syntax.
- Use at least 3 assert test cases to confirm the corrected code works.

Ans:-

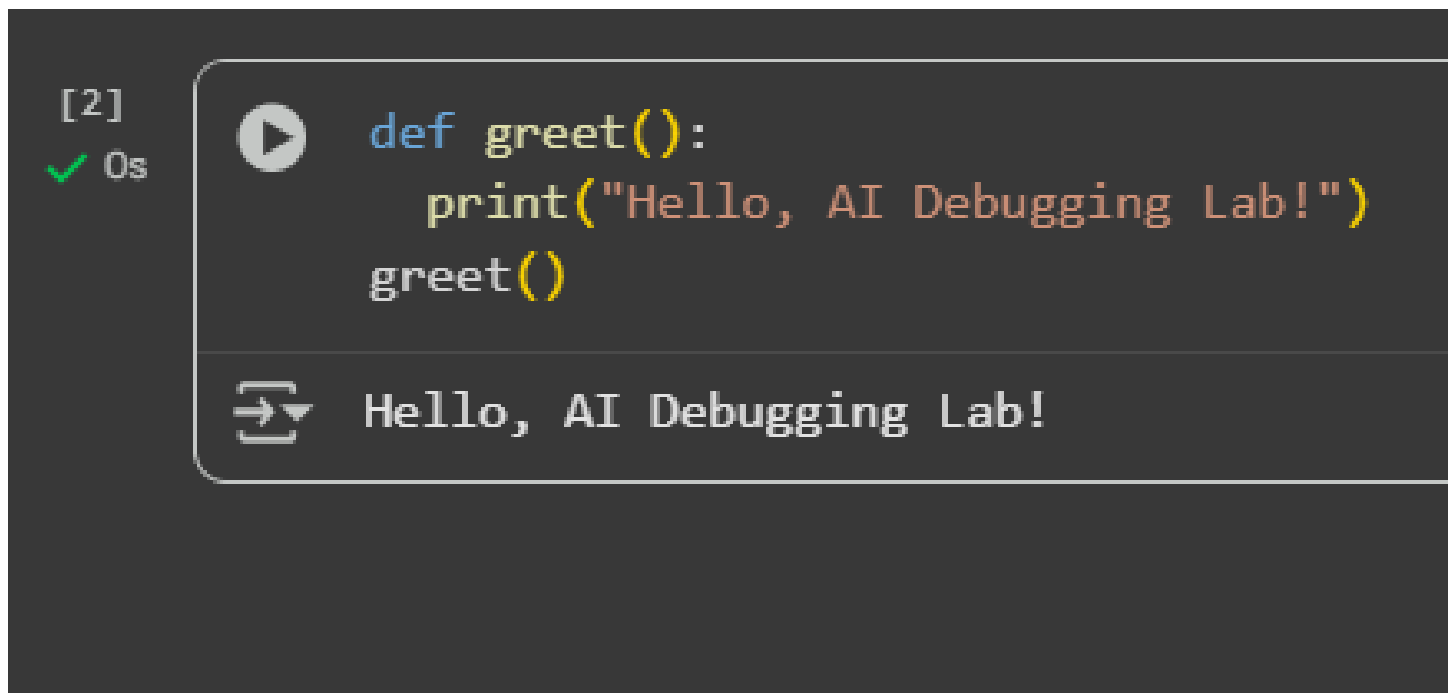


The screenshot shows a code editor interface with a dark theme. At the top, there's a tab labeled "Gemini". Below it, a code snippet is displayed. The first part of the snippet is a function definition: `def greet():`. The second line is a print statement: `print “Hello, AI Debugging Lab!”`. The third line is `greet()`. A red squiggly line is under the opening quote of the print statement, indicating a syntax error. Below the code, a message box shows the error: `SyntaxError: invalid character '“' (U+201C)`. To the right of the error message, there's a button labeled "Explain error".

```
[1] def greet():  
    print “Hello, AI Debugging Lab!”  
    greet()  
  
File “/tmp/ipython-input-413242816.py”, line 2  
    print “Hello, AI Debugging Lab!”  
        ^  
SyntaxError: invalid character '“' (U+201C)
```

Next steps: [Explain error](#)

Prompt **correct the syntax i am getting error.**



The screenshot shows a code editor with a dark background. On the left, there is a status bar with "[2]" and a green checkmark followed by "0s". The main area contains a Python function definition: 

```
def greet():  
    print("Hello, AI Debugging Lab!")  
    greet()
```

 Below the code, there is a terminal output showing the result of the function call: 

```
⇒ Hello, AI Debugging Lab!
```

Task Description#2:- (Logical Error - Incorrect Condition in an if Statement)

Task :Supply a function where an if-condition mistakenly uses=insted of ==. Let AI identify and fix the issue.

#Bug: Using assignment (=) instead of comparision(==)

```
def check_number(n):  
    if n=10:  
        return "ten"  
    else:  
        return "Not Ten"
```

Requirements:

- Ask AI to explain why this cause a bug.
- Correct the code and verify with 3 assert test cases.

Expected Output #2:

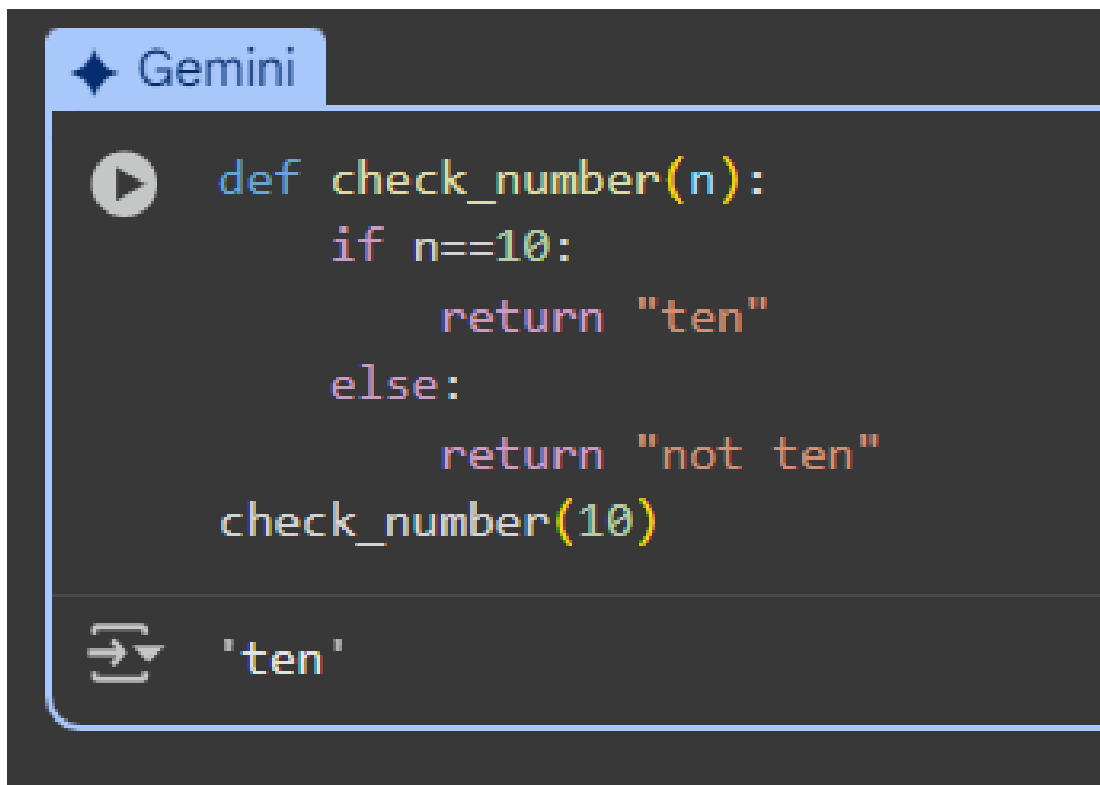
- Corrected code using == with explanation and successful test excution.

Ans:-

AI Response: You are getting a `SyntaxError` in cell 6NJGCxVA1\_4t because you are using the assignment operator `=` inside the `if` statement instead of the comparison operator `==`. The assignment operator is used to assign a value to a variable, while the comparison operator is used to check if two values are equal.

I will fix this by changing `=` to `==` in the `if` statement.

Output:



The screenshot shows a code editor window with a dark background. At the top left, there is a blue tab labeled "Gemini" with a diamond icon. Below the tab, there is a play button icon. The code is as follows:

```
def check_number(n):  
    if n==10:  
        return "ten"  
    else:  
        return "not ten"  
check_number(10)
```

At the bottom of the code block, there is a return icon (two arrows pointing out) followed by the output string `'ten'`.

Task Description #3(Runtime Error- File Not Found)

Task: Provide code that attempts to open a non-existent file and crashes.

Use AI to apply safe error handling.

#Bug: Program crashes if file is missing

```
def read_file(filename):  
    with open(filename, 'r') as f:
```

```
return f_read()
```

```
print (read_file("nonexistent.txt"))
```

Requirements:

- Implement a try-except block suggested by AI.
- Add a user- friendly error message.
- Test with at least 3 scenarios: file exists, file missing, invalid path.

Expected Output #3:

- Safe file handling with exception management.

Ans:-

```
def read_file(filename):
    try:
        with open(filename, 'r') as f:
            return f.read()
    except FileNotFoundError:
        return f"Error: The file '{filename}' was not found."
    except PermissionError:
        return f"Error: Permission denied while accessing '{filename}'."
    except Exception as e:
        return f"Unexpected error: {e}"

# ---- Test Scenarios ----
print("Test 1 (File exists):")
with open("sample.txt", "w") as f: # create a file for test
    f.write("Hello, World!")
print(read_file("sample.txt")) # ✓ should print file content

print("\nTest 2 (File missing):")
print(read_file("nonexistent.txt")) # ✗ should show error message

print("\nTest 3 (Invalid path):")
print(read_file("/invalid_path/test.txt")) # ✗ should show error message
```

Test 1 (File exists):  
Hello, World!

Test 2 (File missing):  
Error: The file 'nonexistent.txt' was not found.

Test 3 (Invalid path):  
Error: The file '/invalid\_path/test.txt' was not found.

#### Task Description#4 (AttributeError – Calling a Non-Existed Method)

Task: Give a class where a non-existed method is called (e.g., `obj.undefined_method()`). Use AI to debug and fix.

#Bug: Calling an undefined method

```
class Car:

    def start(self):

        return "Car started"

my_car=Car()

print(my_car.drive()) #drive() is not defined.
```

Requirements:

- Student must analyse whether to define the missing method or correct the method call.
- Use 3 assert tests to confirm the corrected class works.

Expected output#4:

- Corrected class with clear AI explanation.

Ans:-

```

class Car:
    def start(self):
        return "Car started"

    def drive(self): # ✅ Added missing method
        return "Car is driving"

# Create object
my_car = Car()

# Correct method calls
print(my_car.start()) # Output: Car started
print(my_car.drive()) # Output: Car is driving

# ---- Assert Tests ----
assert my_car.start() == "Car started"
assert my_car.drive() == "Car is driving"
assert isinstance(my_car, Car)

```

explanation:-Now we must decide:

- Either define a drive() method, OR
- Correct the call to start().

Since the task is ambiguous, let's assume the intention is that the car should **start driving**. So the missing method **drive()** should be defined.

Task Description #5:(TypeError – Mixing Strings and Integers in addition)

Task: Provide Code that adds an integer and string ("5"+2) causing a TypeError. Use AI to resolve the bug.

#Bug: TypeError due to mixing string and integers

```

def add_five(value):
    return value+5

print(add_five("10"))

```

Requirements:

- Ask AI for two solutions: type casting and string concatenation.
- Validate with 3 assert test cases.

Expected Output#5:

- Corrected code that runs successfully for multiple inputs.

Ans:-

### # Bug:

TypeError due to mixing string and integer.

### ◆ AI Fix Explanation

The error occurs because Python cannot directly add a **string** ("10") to an **integer** (5). We have two possible solutions:

1. **Type Casting (convert to integer before addition).**
2. **String Concatenation (convert to string before joining).**

```
# ---- Solution 1: Type Casting ----
def add_five_int(value):
    return int(value) + 5 # convert to integer

print("Type Casting Solution:", add_five_int("10")) # ✓ Output: 15

# ---- Solution 2: String Concatenation ----
def add_five_str(value):
    return str(value) + "5" # convert to string

print("String Concatenation Solution:", add_five_str("10")) # ✓ Output: "105"

# ---- Assert Test Cases ----
# Type Casting solution
assert add_five_int("10") == 15
assert add_five_int(20) == 25
assert add_five_int(0) == 5

# String Concatenation solution
assert add_five_str("10") == "105"
assert add_five_str("Hello") == "Hello5"
assert add_five_str(7) == "75"
```