ASSIGNMENT-7.1

HT NO:-2303A52292

BATCH:-43

Task Description #1 (Syntax Errors – Missing Parentheses in Print Statement)

Task: 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:

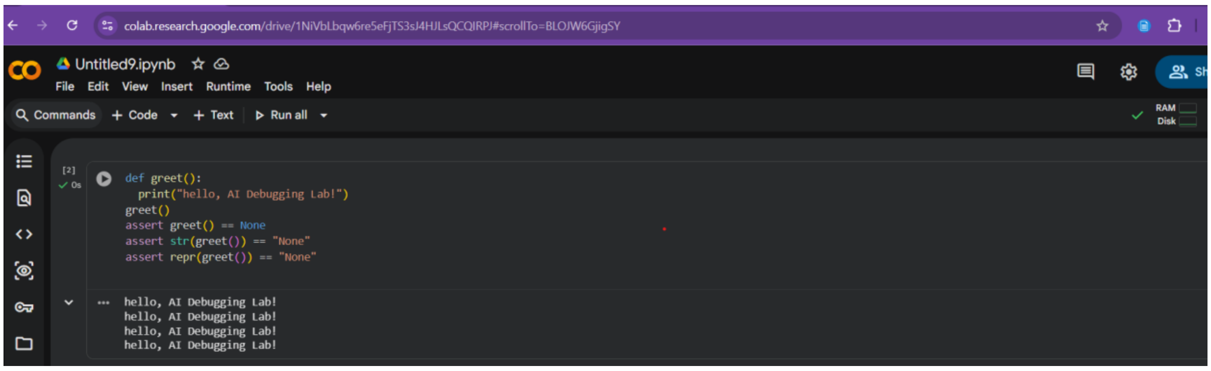
• Run the given code to observe the error.

• Apply AI suggestions to correct the syntax.

• Use at least 3 assert test cases to confirm the corrected code works.

Expected Output #1:

• Corrected code with proper syntax and AI explanation.


colab.research.google.com/drive/1NiVbLbqw6re5eFjTS3sJ4HJLsQCQIRPJ#scrollTo=BLOJW6GjigSY
Untitled9.ipynb
File Edit View Insert Runtime Tools Help
Commands + Code + Text Run all

```python
def greet():
  print("hello, AI Debugging Lab!")
greet()
assert greet() == None
assert str(greet()) == "None"
assert repr(greet()) == "None"
```

```
hello, AI Debugging Lab!
hello, AI Debugging Lab!
hello, AI Debugging Lab!
hello, AI Debugging Lab!
```

Task Description #2 (Incorrect condition in an If Statement)

Task: Supply a function where an if-condition mistakenly uses =

instead of ==. Let AI identify and fix the issue.

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

def check_number(n):

if n = 10:

return "Ten"

else:

return "Not Ten"

Requirements:

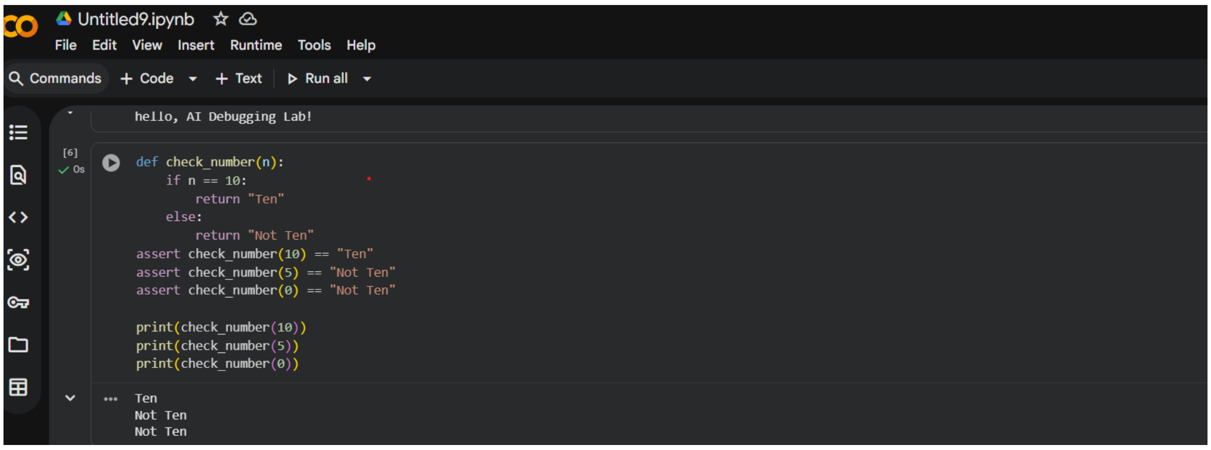
• Ask AI to explain why this causes a bug.

• Correct the code and verify with 3 assert test cases.

Expected Output #2:

• Corrected code using == with explanation and successful test

execution.



Untitled9.ipynb
File Edit View Insert Runtime Tools Help
Commands + Code + Text ▷ Run all

```
hello, AI Debugging Lab!
```

```
[6]
def check_number(n):
    if n == 10:
        return "Ten"
    else:
        return "Not Ten"
assert check_number(10) == "Ten"
assert check_number(5) == "Not Ten"
assert check_number(0) == "Not Ten"

print(check_number(10))
print(check_number(5))
print(check_number(0))
```

```
Ten
Not Ten
Not 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.

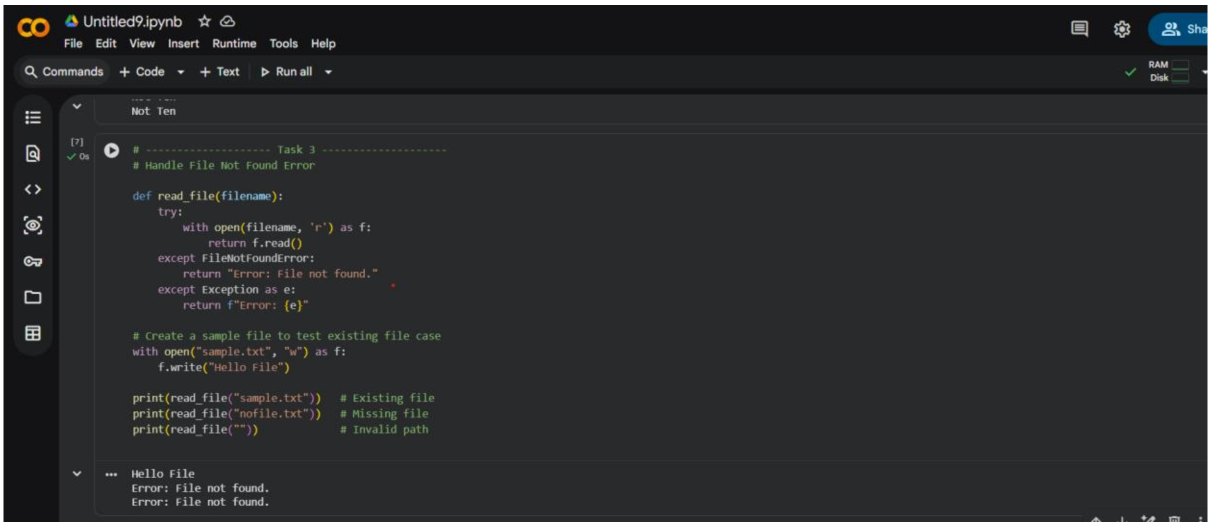
• Add a user-friendly error message.

• Test with at least 3 scenarios: file exists, file missing, invalid

path.

Expected Output #3:

• Safe file handling with exception management.



```
Not Ten
```

```
# ------------------- Task 3 -------------------
# Handle File Not Found Error

def read_file(filename):
    try:
        with open(filename, 'r') as f:
            return f.read()
    except FileNotFoundError:
        return "Error: File not found."
    except Exception as e:
        return f"Error: {e}"

# Create a sample file to test existing file case
with open("sample.txt", "w") as f:
    f.write("Hello File")

print(read_file("sample.txt"))   # Existing file
print(read_file("nofile.txt"))   # Missing file
print(read_file(""))             # Invalid path
```

```
Hello File
Error: File not found.
Error: File not found.
```

Task Description #4 (Calling a Non-Existent Method)

Task: Give a class where a non-existent 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:

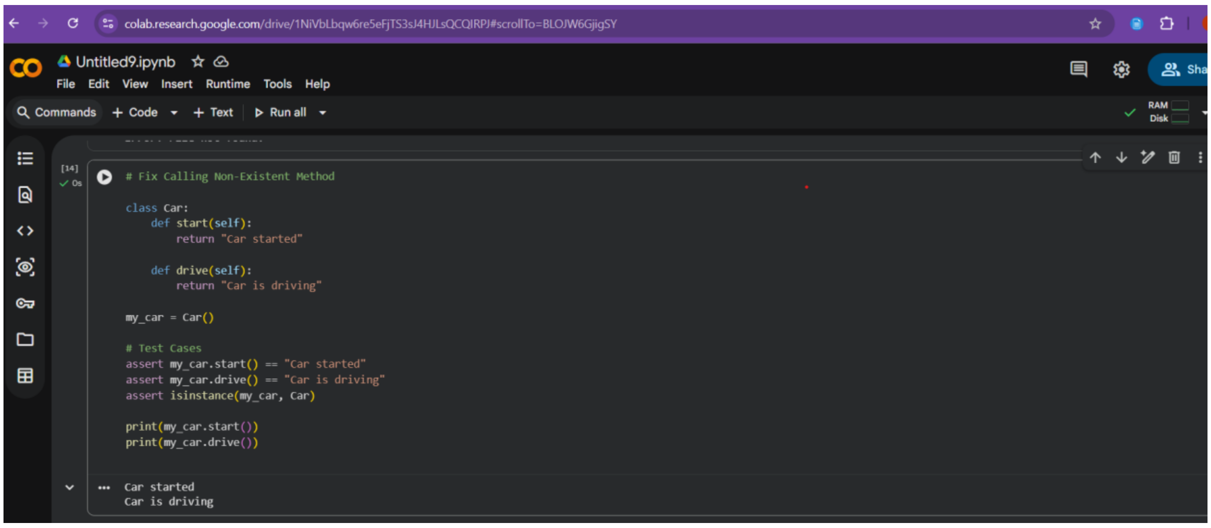
• Students must analyze whether to define the missing method

or correct the method call.

• Use 3 assert tests to confirm the corrected class works.

Expected Output #4:

• Corrected class with clear AI explanation.


colab.research.google.com/drive/1NiVbLbqw6re5eFjTS3sJ4HJLsQCQIRPJ#scrollTo=BLOJW6GjigSY
Untitled9.ipynb
File Edit View Insert Runtime Tools Help
Commands + Code + Text Run all
RAM Disk
[14]
0s

```python
# Fix Calling Non-Existent Method

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

    def drive(self):
        return "Car is driving"

my_car = Car()

# Test Cases
assert my_car.start() == "Car started"
assert my_car.drive() == "Car is driving"
assert isinstance(my_car, Car)

print(my_car.start())
print(my_car.drive())
```

```
Car started
Car is driving
```

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 integer

def add_five(value):

return value + 5

print(add_five("10"))

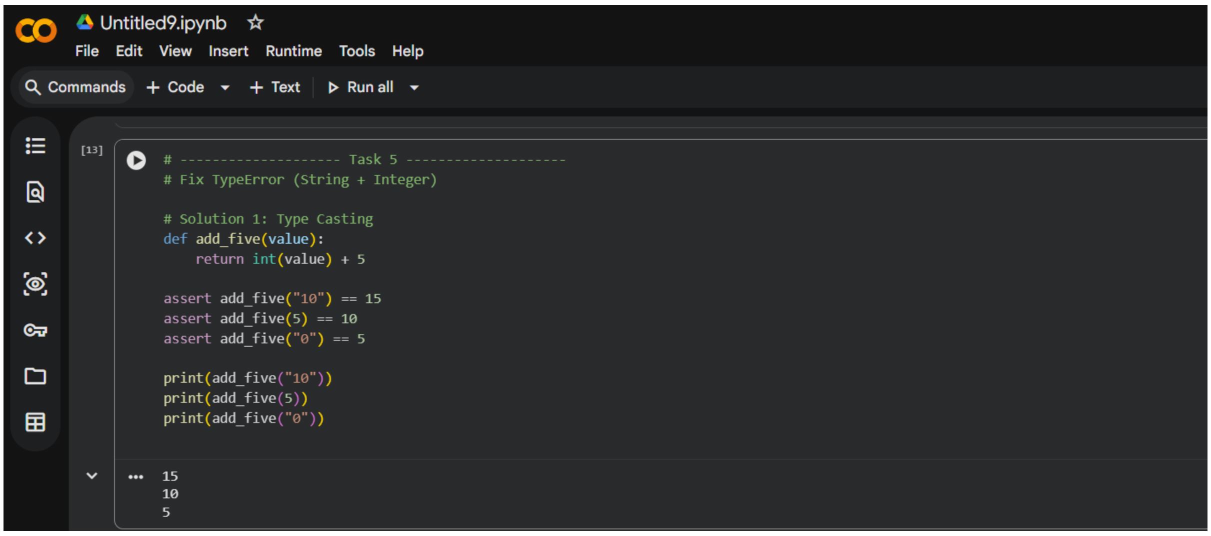
Requirements:

• 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.


Untitled9.ipynb
File Edit View Insert Runtime Tools Help
Commands + Code + Text Run all
[13]

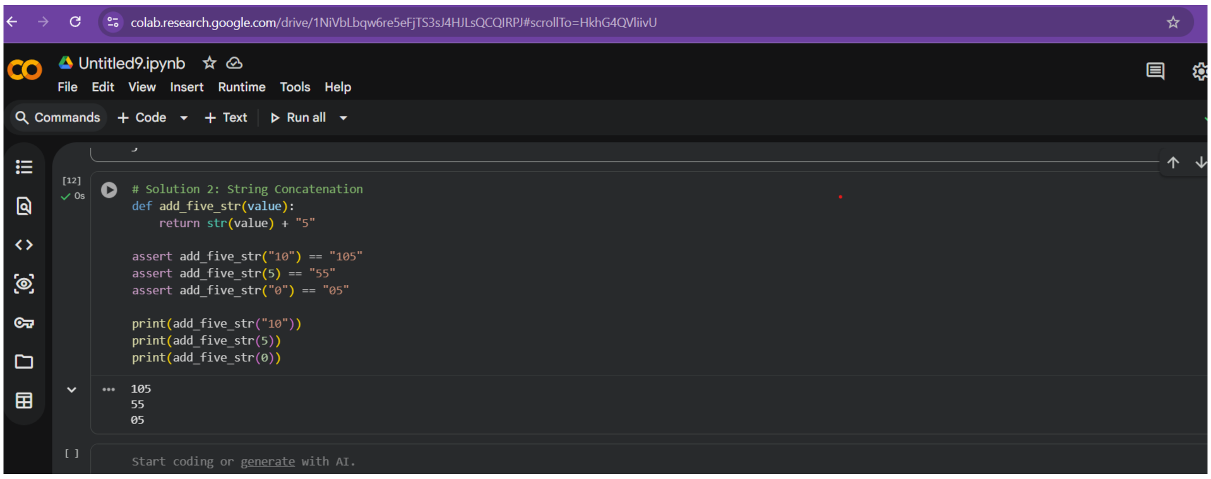
```python
# ------------------- Task 5 -------------------
# Fix TypeError (String + Integer)

# Solution 1: Type Casting
def add_five(value):
    return int(value) + 5

assert add_five("10") == 15
assert add_five(5) == 10
assert add_five("0") == 5

print(add_five("10"))
print(add_five(5))
print(add_five("0"))
```

```
15
10
5
```


colab.research.google.com/drive/1NiVbLbqw6re5eFjTS3sJ4HJLsQCQIRPJ#scrollTo=HkhG4QVliivU
Untitled9.ipynb
File Edit View Insert Runtime Tools Help
Commands + Code + Text Run all
[12]
0s

```python
# Solution 2: String Concatenation
def add_five_str(value):
    return str(value) + "5"

assert add_five_str("10") == "105"
assert add_five_str(5) == "55"
assert add_five_str("0") == "05"

print(add_five_str("10"))
print(add_five_str(5))
print(add_five_str(0))
```

```
105
55
05
```

[ ]
Start coding or generate with AI.