

# AI Assisted Coding (III Year) Assignment

Name: B. Rithwik

HT NO: 2303A52330

Batch: 35

**Lab 7:** Error Debugging with AI – Systematic Approaches to Finding and Fixing Bugs

Week 4 – Monday

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

## Task 1: Syntax Error – Missing Parentheses in Print Statement

Buggy Code

The screenshot shows a dark-themed code editor window. At the top, there's a play button icon and the letter 's'. Below the editor area, there's a status bar with icons for file operations. The code itself is as follows:

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

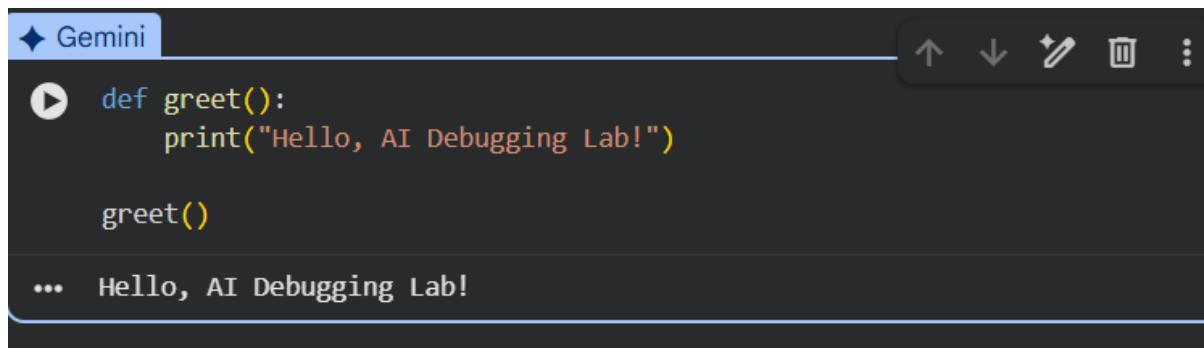
greet()
```

After the code, there's an ellipsis (...), followed by the path to the file: File "/tmp/ipython-input-3503158804.py", line 2. On line 2, there's a line of code: print "Hello, AI Debugging Lab!". A small yellow arrow points to the first character of the word "print". Below this, the error message is displayed in red: IndentationError: expected an indented block after function definition on line 1. At the bottom of the editor, there's a dashed line separating the code from the footer. In the footer, the text "Next steps:" is followed by a button labeled "Explain error".

## Observed Error

- SyntaxError occurs because Python 3 requires parentheses in print().

AI Fix (Corrected Code):



The screenshot shows the Gemini AI interface with a dark theme. At the top, it says "Gemini". Below that is a toolbar with icons for up, down, edit, and more. The main area has a play button icon followed by the code:

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

greet()
...
```

The output below the code shows the result of running the script:

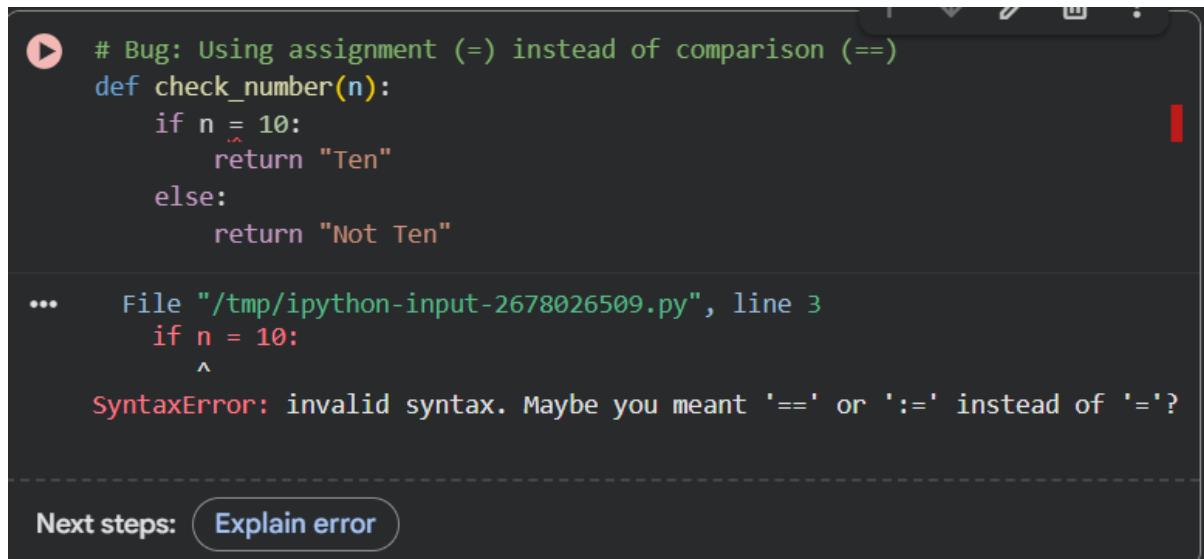
```
Hello, AI Debugging Lab!
```

## Explanation

- In Python 3, print is a function, so parentheses are mandatory.
- Indentation was also corrected.

## Task 2: Logic Error – Incorrect Condition in If Statement

### Buggy Code



The screenshot shows the Gemini AI interface with a dark theme. At the top, it says "Gemini". Below that is a toolbar with icons for up, down, edit, and more. The main area has a play button icon followed by the code:

```
# Bug: Using assignment (=) instead of comparison (==)
def check_number(n):
    if n = 10:
        return "Ten"
    else:
        return "Not Ten"

...
File "/tmp/ipython-input-2678026509.py", line 3
    if n = 10:
        ^
SyntaxError: invalid syntax. Maybe you meant '==' or ':=' instead of '='?
```

At the bottom, there is a "Next steps:" section with a button labeled "Explain error".

### Why This Causes a Bug

- = is used for assignment, not comparison.
- Conditions require ==.

## AI Fix (Corrected Code) :

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

+ Code + Text

## Task 3: Runtime Error – File Not Found

### Buggy Code

```
▶ # 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"))

...
-
FileNotFoundException                                     Traceback (most recent call
last)
/tmp/ipython-input-3903500015.py in <cell line: 0>()
      4         return f.read()
      5
----> 6 print(read_file("nonexistent.txt"))

/tmp/ipython-input-3903500015.py in read_file(filename)
      1 # Bug: Program crashes if file is missing
      2 def read_file(filename):
----> 3     with open(filename, 'r') as f:
      4         return f.read()
      5

FileNotFoundException: [Errno 2] No such file or directory: 'nonexistent.txt'
```

Next steps: [Explain error](#)

### Observed Error

- FileNotFoundError occurs when file does not exist.

## AI Fix (Safe Code with Try-Except):

```
▶ def read_file(filename):
    try:
        with open(filename, 'r') as f:
            return f.read()
    except FileNotFoundError:
        return "Error: File not found."
    except Exception:
        return "Error: Invalid file path or access issue."
```

## Task 4: Calling a Non-Existent Method

### Buggy Code:

```
▶ # Bug: Calling an undefined method
  class Car:
      def start(self):
          return "Car started"

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

... -----

```
AttributeError                                Traceback (most recent call
last)
/tmp/ipython-input-566315127.py in <cell line: 0>()
      5
      6 my_car = Car()
----> 7 print(my_car.drive()) # drive() is not defined

AttributeError: 'Car' object has no attribute 'drive'
```

Next steps: [Explain error](#)

### Problem

- `drive()` method does not exist, so `AttributeError` occurs.

## AI Fix: Correct the Method Call:

```
▶ class Car:  
    def start(self):  
        return "Car started"  
  
    my_car = Car()  
    print(my_car.start())  
  
... Car started
```

## Task 5: TypeError – Mixing Strings and Integers in Addition

### Buggy Code:

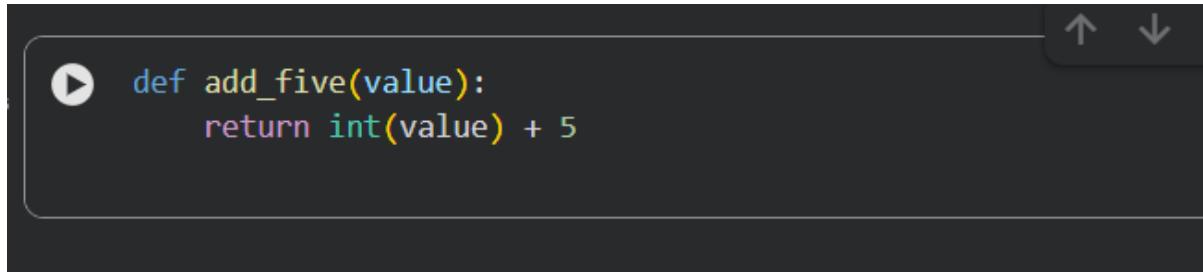
```
▶ # Bug: TypeError due to mixing string and integer  
def add_five(value):  
    return value + 5  
  
print(add_five("10"))  
  
...  
-  
TypeError                                         Traceback (most recent call  
last)  
/tmp/ipython-input-3441793644.py in <cell line: 0>()  
      3     return value + 5  
      4  
----> 5 print(add_five("10"))  
      6  
  
/tmp/ipython-input-3441793644.py in add_five(value)  
      1 # Bug: TypeError due to mixing string and integer  
      2 def add_five(value):  
----> 3     return value + 5  
      4  
      5 print(add_five("10"))  
  
TypeError: can only concatenate str (not "int") to str
```

Next steps: [Explain error](#)

### Observed Error

- TypeError occurs because "10" is a string and cannot be added to integer 5.

**AI Correction:** String Concatenation:



```
def add_five(value):
    return int(value) + 5
```

## Final Conclusion

This lab demonstrated how AI tools help in debugging different types of errors:

- Syntax Errors (missing parentheses, indentation)
- Logic Errors (wrong operators in conditions)
- Runtime Errors (missing files, invalid paths)
- Attribute Errors (undefined method calls)
- Type Errors (mixing incompatible data types)

AI-assisted debugging improves productivity, but human understanding is necessary to validate fixes and write reliable code.