

## ASSIGNMENT-7.1

**Name:** M .Raja

**Hall No:**2303A52277

**Batch:**36

**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()
```

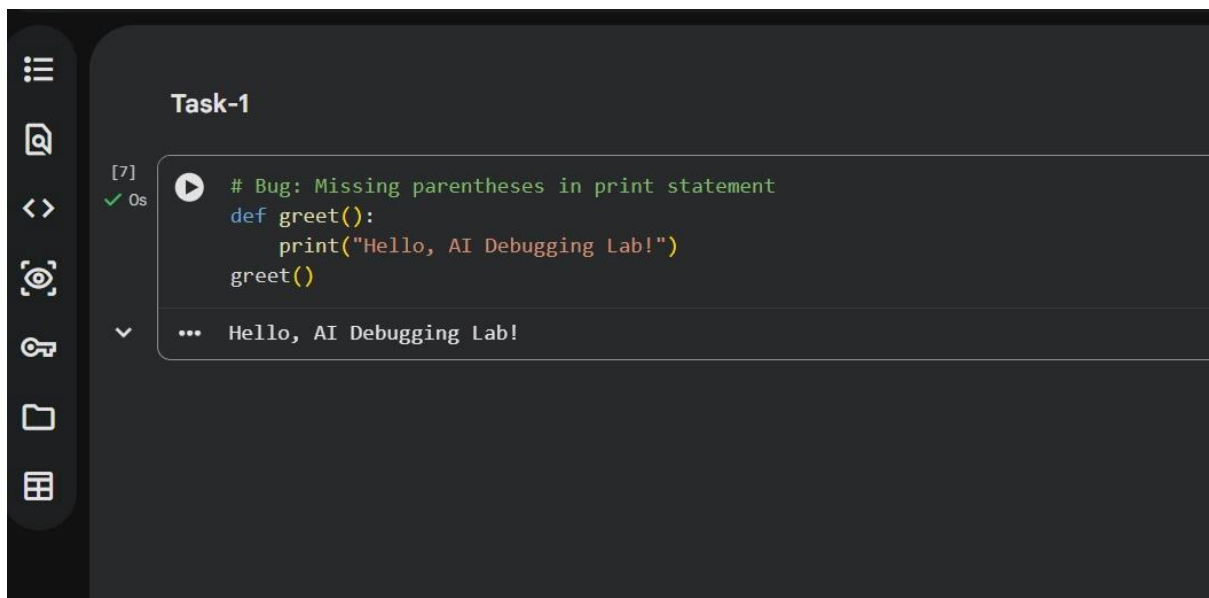
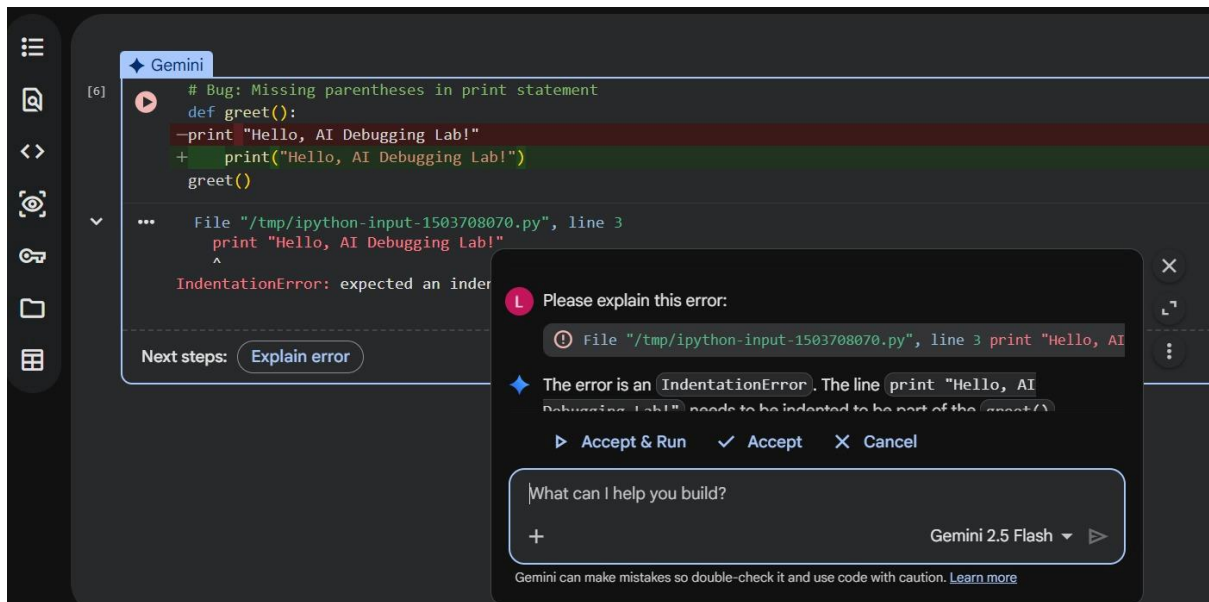
Requirements:

- 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.

Output:



## 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"
```

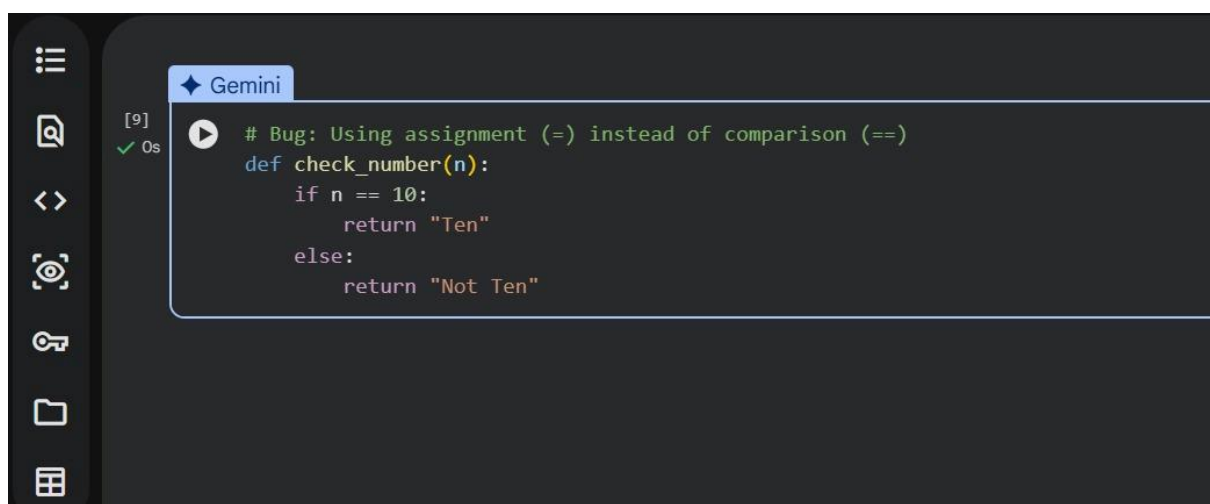
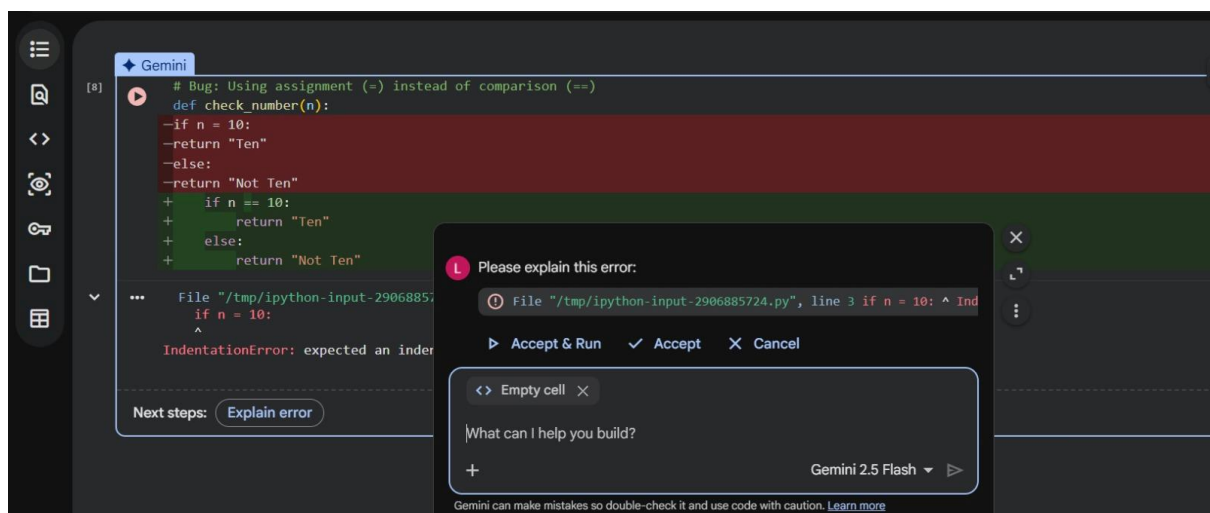
Requirements:

- 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.

Output:



### 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"))
```

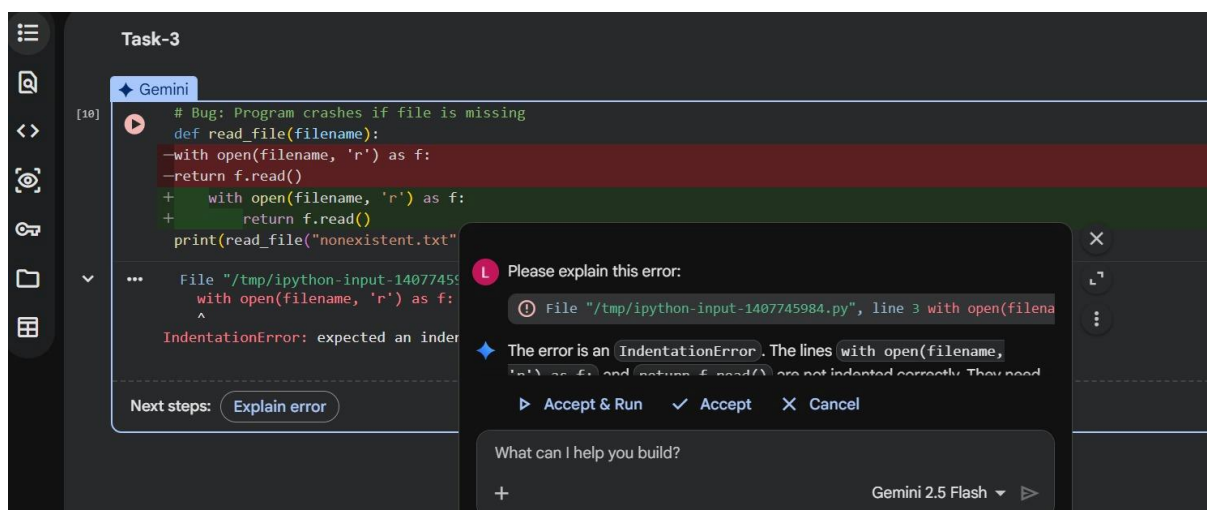
Requirements:

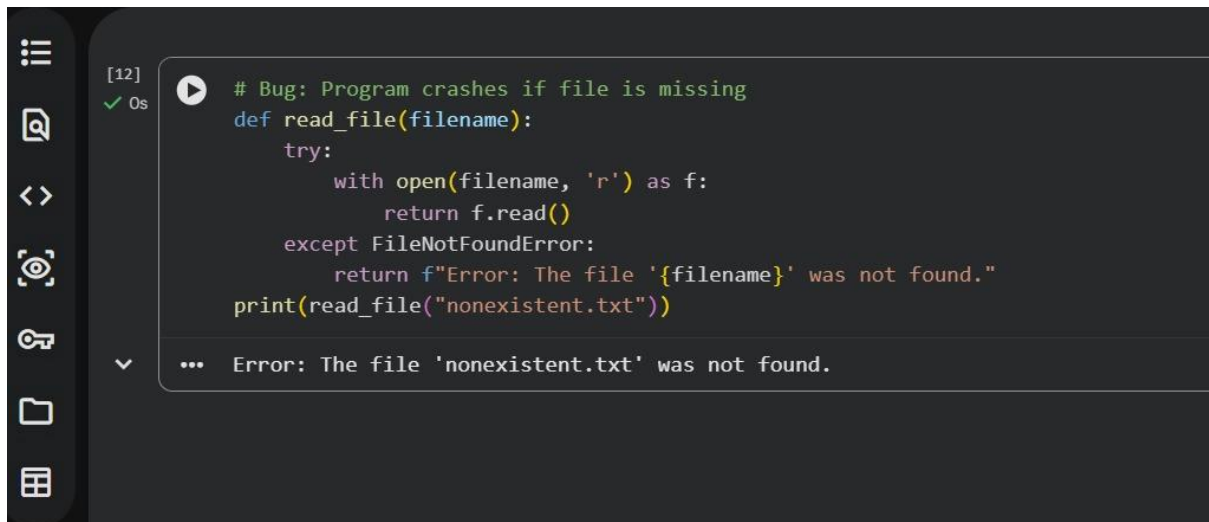
- Implement a try-except block suggested by AI.
- Add a user-friendly error message.
- Test with at least 3 scenarios: file exists, file missing, invalid path.

Expected Output #3:

- Safe file handling with exception management.

Output:



A screenshot of a code editor interface. On the left is a dark sidebar with icons for file explorer, search, and other IDE features. The main area has a dark background. At the top left of the editor, it says "[12] ✓ Os". Below this is a Python code snippet:

```
# Bug: Program crashes if file is missing
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."
print(read_file("nonexistent.txt"))
```

Below the code, there is a console output area showing the result of the execution:

```
... Error: The file 'nonexistent.txt' was 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
```

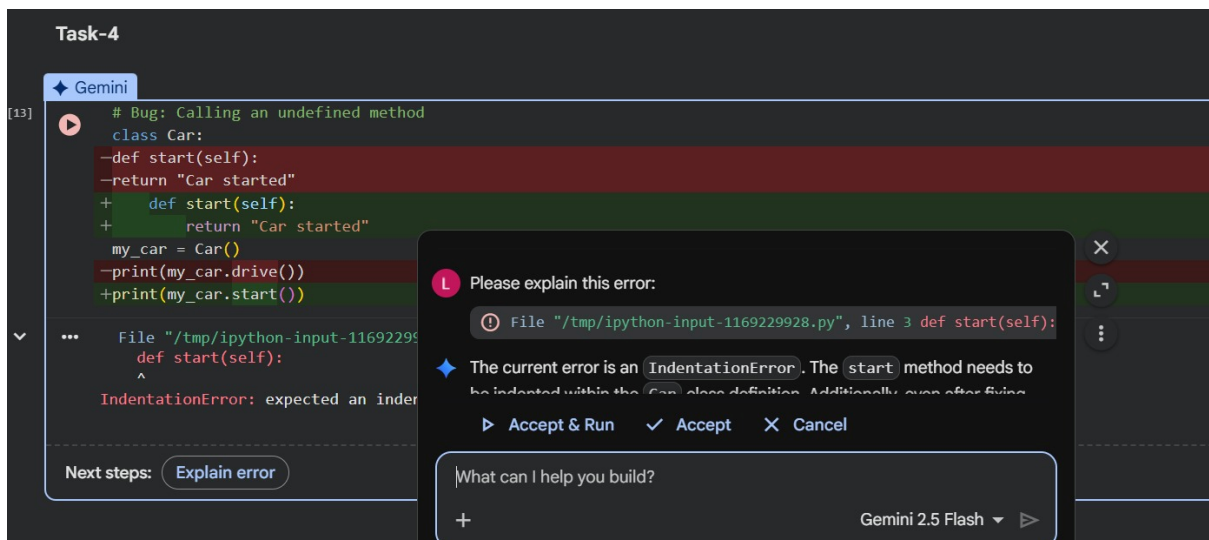
Requirements:

- 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.

Output:



## 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.

Output:

The screenshot shows a code editor window titled "Task-5". The code is as follows:

```
[15] # Bug: TypeError due to mixing string and integer
def add_five(value):
    -return value + 5
    + return int(value) + 5
    print(add_five("10"))
```

The line `-return value + 5` is highlighted in red, indicating an error. Below the code, an error message is displayed: `IndentationError: expected an indented block`. To the right of the code, a panel titled "Please explain this error:" provides a detailed explanation: "The current error is an `IndentationError`. The line `return value + 5` needs to be indented under the `add_five` function. After fixing this, there are no more errors." Below the explanation, there are buttons for "Accept & Run", "Accept", and "Cancel". At the bottom of the panel, there is a text input field with the placeholder "What can I help you build?" and a "Gemini 2.5 Flash" dropdown menu.

The screenshot shows the same code editor window after the error has been corrected. The code is now:

```
[16] # Bug: TypeError due to mixing string and integer
def add_five(value):
    return int(value) + 5
    print(add_five("10"))
```

The line `return int(value) + 5` is now correctly indented. Below the code, the output is displayed: `15`. The output is preceded by a green checkmark and the text "0s", indicating that the code ran successfully.