

ASSIGNMENT-7.1

Name: G.Vignesh

HT.No:2303A52359

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:

The screenshot shows the Gemini AI interface. A code editor window displays the following Python code:

```
[6] # Bug: Missing parentheses in print statement
def greet():
    -print "Hello, AI Debugging Lab!"
    + print("Hello, AI Debugging Lab!")
    greet()

... File "/tmp/ipython-input-1503708070.py", line 3
        print "Hello, AI Debugging Lab!"
        ^
IndentationError: expected an indent
```

A tooltip box is open over the error line, containing the message: "L Please explain this error: The error is an IndentationError. The line print "Hello, AI Debugging Lab!" needs to be indented to be part of the greet() function." Below the tooltip are three buttons: "Accept & Run", "Accept", and "Cancel". At the bottom of the tooltip box is a text input field asking "What can I help you build?" and a "Gemini 2.5 Flash" button.

The screenshot shows the Gemini AI interface with the code editor window titled "Task-1". The code has been corrected:

```
[7] ✓ 0s
[7] # Bug: Missing parentheses in print statement
def greet():
    print("Hello, AI Debugging Lab!")
    greet()

... 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.

Output:

The screenshot shows a Jupyter Notebook cell with the following Python code:

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

... File "/tmp/ipython-input-2906885724.py", line 3 if n = 10: ^ IndentationError: expected an inden
```

An AI pop-up from Gemini asks: "Please explain this error." It includes a link to the file and line number, and buttons for "Accept & Run", "Accept", and "Cancel". Below the pop-up, there's a "Next steps" button labeled "Explain error". A sidebar on the right says "Gemini 2.5 Flash".

The screenshot shows the same Jupyter Notebook cell after the code has been corrected. The output cell now shows a green checkmark and "0s" indicating successful execution. The code is now correctly written using == instead of =.

```
# Bug: Using assignment (=) instead of comparison (==)
def check_number(n):
    if n == 10:
        return "Ten"
    else:
        return "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.

Output:

The screenshot shows the Gemini 2.5 Flash interface with two code editor panes. The top pane displays the following Python code with an indentation error:

```
[10]  # Bug: Program crashes if file is missing  
def read_file(filename):  
    with open(filename, 'r') as f:  
        return f.read()  
    +     with open(filename, 'r') as f:  
    +         return f.read()  
    print(read_file("nonexistent.txt"))
```

An error message is shown in the status bar: "IndentationError: expected an indent". A modal window titled "Please explain this error:" provides details about the error and offers options to "Accept & Run", "Accept", or "Cancel".

The bottom pane shows the corrected code using a try-except block to handle the FileNotFoundError:

```
[12] ✓ 0s  # 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"))
```

The status bar in the bottom pane indicates "Gemini 2.5 Flash".

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-4

```
[13] Gemini
  # Bug: Calling an undefined method
  class Car:
    -def start(self):
    -return "Car started"
    + def start(self):
    +     return "Car started"
    my_car = Car()
    -print(my_car.drive())
    +print(my_car.start())

  ... File "/tmp/ipython-input-1169229928.py", line 3 def start(self):
        ^
IndentationError: expected an indent
```

Please explain this error:

The current error is an `IndentationError`. The `start` method needs to be indented within the `Car` class definition. Additionally, even after fixing the indentation, there is a syntax error in the line `-print(my_car.drive())` which should be `+print(my_car.start())`.

Next steps: Explain error

What can I help you build?

Gemini 2.5 Flash ▶

Task-4

```
[14] Gemini
  # Bug: Calling an undefined method
  class Car:
    def start(self):
      return "Car started"
    my_car = Car()
    print(my_car.start())

  ... Car started
```

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:

Task-5

Gemini

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

...
File "/tmp/ipython-input-958511054.py", line 3
    return value + 5
          ^
IndentationError: expected an indent
```

Next steps: Explain error

L Please explain this error:
① File "/tmp/ipython-input-958511054.py", line 3 return value + 5

◆ The current error is an `IndentationError`. The line `return value + 5` needs to be indented under the `add_five` function. After fixing this there

▶ Accept & Run ✓ Accept ✖ Cancel

| What can I help you build?
+ Gemini 2.5 Flash ▶

Gemini can make mistakes so double-check it and use code with caution. Learn more.

Task-5

Gemini

```
[16] ✓ 0s # Bug: TypeError due to mixing string and integer
def add_five(value):
    return int(value) + 5
print(add_five("10"))

...
15
```