

ASSIGNMENT-7.1

Name: G.Laxmi Pranav

Hall No:2303A52403

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 a Python file with the following code:

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

An error message is shown: `IndentationError: expected an indent`. Below the code editor, a sidebar says "Next steps: Explain error". A modal window titled "L Please explain this error:" contains the error message and a suggestion: "The error is an `IndentationError`. The line `print "Hello, AI Debugging Lab!"` needs to be indented to be part of the `greet()`". It includes "Accept & Run", "Accept", and "Cancel" buttons. At the bottom right of the modal is "Gemini 2.5 Flash".

The screenshot shows the Gemini AI interface after the bug was fixed. The code editor window now displays the corrected code:

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

The output window shows the result: `Hello, AI Debugging Lab!`. The sidebar still says "Next steps: Explain error".

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"
```

An error message is displayed: `IndentationError: expected an indent`. A tooltip from Gemini asks for an explanation of the error. The tooltip also includes options to accept and run the code or cancel, and a link to Gemini's help documentation.

The screenshot shows the same Jupyter Notebook cell, but the code has been corrected to use the correct comparison operator (`==`). The output cell now displays the corrected code:

```
# 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 AI interface with two code snippets. The top snippet is titled 'Task-3' and shows a Python script with a bug. The code defines a function `read_file` that attempts to open a file named 'nonexistent.txt' and read its contents. The AI has suggested a fix where the code uses a context manager (`with`) to handle the file. A tooltip from the AI asks for an explanation of the error, which it identifies as an `IndentationError`. The bottom snippet shows the corrected code, which includes a try-except block to catch `FileNotFoundException` and return an error message if the file is not found.

```
[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"))

...
File "/tmp/ipython-input-1407745984.py", line 3 with open(filename,
    ^
IndentationError: expected an indent

Next steps: Explain error
```

L Please explain this error:
 ⓘ File "/tmp/ipython-input-1407745984.py", line 3 with open(filename,
 ^)
 ◆ The error is an `IndentationError`. The lines `with open(filename,`
`'r')` as `f:` and `return f.read()` are not indented correctly. They need
 ▶ Accept & Run ✓ Accept ✕ Cancel

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

```
[12] ✓ os
# 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"))

...
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-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-11692298.py", line 3 def start(self):
^
IndentationError: expected an indent

L Please explain this error:
① File "/tmp/ipython-input-11692298.py", line 3 def start(self):
◆ The current error is an IndentationError. The start method needs to
be indented within the Car class definition. Additionally, even after fixing
the indentation, the method name is misspelled as start instead of __init__.
▶ Accept & Run ✓ Accept ✖ Cancel

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:

The screenshot shows the Gemini interface with a dark theme. A code cell titled "Task-5" contains Python code with a syntax error. The code is:

```
# 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-95851105.py"
        return value + 5
        ^
IndentationError: expected an indent
```

A tooltip from Gemini provides an explanation of the error:

L Please explain this error:
 ⓘ File "/tmp/ipython-input-95851105.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.

The screenshot shows the Gemini interface with a dark theme. A code cell titled "Task-5" contains Python code that has been corrected by Gemini. The code is:

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

... 15
```