

ASSIGNMENT-7.1

Name: L.Srinivas

Hall No:2303A52274

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 Python code with a red error indicator:

```
# 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, asking for an explanation:

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.

Next steps: Explain error

Accept & Run Accept Cancel

What can I help you build?

Gemini can make mistakes so double-check it and use code with caution. [Learn more](#)

The screenshot shows the Gemini AI interface after the error has been fixed. The code editor window displays the corrected Python code:

```
# 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 the Gemini AI interface. A code editor window displays a Python script with a syntax error. The code is as follows:

```
# 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 at the bottom indicates: `IndentationError: expected an indented line here`. A modal dialog titled "Please explain this error:" is open, showing the error details and options: "Accept & Run", "Accept", and "Cancel". Below the modal, a text input field asks "What can I help you build?" and a "Gemini 2.5 Flash" button is visible.

The screenshot shows the Gemini AI interface after the code has been corrected. The code editor now displays the following corrected Python script:

```
# 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 a dark theme. On the left is a sidebar with various icons. The main area is titled "Task-3". In the code editor, there is a file named "Gemini" containing the following Python code:

```
# 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"))
```

A red horizontal bar highlights the second "with" statement. Below the code editor, there is a "Next steps" button with the option "Explain error". A modal window is open, asking "Please explain this error:" and providing a detailed explanation of an IndentationError. It says: "The error is an IndentationError. The lines with open(filename, 'r') and return f.read() are not indented correctly. They need to be indented under the previous line." At the bottom of the modal are buttons for "Accept & Run", "Accept", and "Cancel".

The screenshot shows a code editor interface with a dark theme. On the left is a sidebar with various icons: three horizontal lines, a magnifying glass, a double arrow, an eye, a lock, a folder, and a square. The main area has a status bar at the top with '[12]' and '0s'. Below that is a code editor window containing the following Python code:

```
# 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.
```

The code defines a function `read_file` that attempts to open a file and return its contents. If the file does not exist, it catches a `FileNotFoundError` and returns an error message. When the code is run, it prints the error message to the console.

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:

① File "/tmp/ipython-input-1169229928.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, run after fixing.

▷ Accept & Run ✓ Accept X 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 AI interface with a dark theme. A code completion pop-up is open over a code editor window titled "Task-5". The code editor contains the following Python code:

```
[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"
      return value + 5
      ^
IndentationError: expected an inden
```

The pop-up has a title "L Please explain this error:" and a message: "The current error is an `IndentationError`. The line `return value + 5` needs to be indented under the `add_five` function. After fixing this there". It includes buttons for "Accept & Run", "Accept", and "Cancel". Below the message is a text input field "What can I help you build?" with a plus sign button. At the bottom right of the pop-up is the text "Gemini 2.5 Flash".

The screenshot shows the Gemini AI interface with a dark theme. The code editor window titled "Task-5" now displays the corrected code:

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

... 15
```

The status bar at the top left shows "0s" next to a green checkmark icon, indicating the code has been successfully run or checked.