

## **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:

The screenshot shows the Gemini AI interface. On the left is a sidebar with various icons. The main area displays a code editor with a dark theme. A tooltip is open over a line of code:

```
# 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
```

The tooltip contains the following text:

L Please explain this error:

File "/tmp/ipython-input-1503708070.py", line 3 print "Hello, AI Debugging Lab!"

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](#)

Gemini 2.5 Flash

The screenshot shows the Gemini AI interface. The sidebar icons are visible on the left. The main area displays a code editor with a dark theme. The code editor shows the following Python script:

```
[7] ✓ 0s
# 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. On the left, there's a sidebar with various icons. In the main area, a code editor window titled 'Gemini' shows a Python script with a bug. The code is:`# 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 says: 'IndentationError: expected an indented line'. Below the code, there's a button 'Explain error'. A modal window titled 'Please explain this error:' appears, asking 'What can I help you build?'. It has three buttons: 'Accept & Run', 'Accept', and 'Cancel'. The 'Accept & Run' button is highlighted. At the bottom right of the modal, it says 'Gemini 2.5 Flash'.

The screenshot shows the Gemini AI interface again. The sidebar is visible on the left. The main area now displays the corrected Python 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 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 Python script with a syntax error highlighted in red. The error message is: "IndentationError: expected an indent". A modal window titled "Please explain this error:" is open, containing the following text:  
File "/tmp/ipython-input-1407745984.py", line 3  
with open(filename, 'r') as f:  
^  
The error is an IndentationError. The lines with open(filename,  
'r') as f and return f.read() are not indented correctly. They need  
to have an extra indentation level.  
Next steps: Explain error

At the bottom of the modal, there are three buttons: "Accept & Run", "Accept", and "Cancel". Below the modal, there is a text input field asking "What can I help you build?" and a "Gemini 2.5 Flash" button.

The screenshot shows a code editor interface with a dark theme. On the left is a vertical toolbar with icons for file operations like new, open, save, and search. The main area displays a Python script. The code defines a function `read_file` that attempts to open a file and return its content. It handles a `FileNotFoundException` by returning an error message. A call to `read_file("nonexistent.txt")` results in the output "Error: The file 'nonexistent.txt' was not found.".

```
[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"))

... 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-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, even after fixing  
the indentation, you will still receive an `AttributeError` because the `drive` method  
is not defined.

▶ 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 interface with a code cell containing Python code. The code defines a function `add_five` that adds 5 to its input. A tooltip from Gemini explains a `IndentationError` that occurred because the line `return value + 5` was not properly indented under the function definition. The tooltip provides a fix: "The current error is an `IndentationError`. The line `return value + 5` needs to be indented under the `add_five` function. After fixing this there will be no errors." There are buttons to "Accept & Run", "Accept", or "Cancel".

```
[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-95851105.py", line 3
    return value + 5
          ^
IndentationError: expected an indent
```

Next steps: Explain 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 will be no errors.  
▶ Accept & Run ✓ Accept ✖ Cancel

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

The screenshot shows the Gemini interface with the same code cell, but now it is marked as successful with a green checkmark and the text "0s". The code has been corrected to use `int(value)` instead of `value + 5`. The output of the code, `15`, is visible below the code cell.

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

...
15
```