

ASSIGNMENT- 7.1

Name : T.Rakshitha

H.T.NO: 2303A51172

Batch : 18

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.

Promt :

```
#The following Python code has a syntax error related to a print statement.
```

```
# Bug Fix: Added parentheses to the print statement (Python 3 syntax)
```

```
# Call the function
```

```
# Assert test cases
```

INPUT:

```
File Edit Selection View Go Run ... Q AI
EXPLORER ...
AI
age and email using conditional statements.py
ai (6.3 task - 5).py
ai_lab(2.1).py
ai_lab(6.1).py
ai(6.3 task - 2).py
ai(6.3 task-3).py
ai(6.3 task-4).py
ai(6.3).py
ai(7.1) task-1.py
ai2.py
Allab(2.1).py
class Student with attributes.py
list are even and odd.py
simple bank account system.py
ai(7.1) task-1.py > ...
1 #The following Python code has a syntax error related to a print statement.
2 def greet():
3     print("Hello, AI Debugging Lab!")
4 greet()
5 # Bug Fix: Added parentheses to the print statement (Python 3 syntax)
6
7 def greet():
8     return "Hello, AI Debugging Lab!"
9
10 # Call the function
11 print(greet())
12
13 # Assert test cases
14 assert greet() == "Hello, AI Debugging Lab!"
15 assert type(greet()) == str
16 assert greet().startswith("Hello")
17 assert greet().endswith("Lab!")
```

OUTPUT:

```
PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS POSTMAN CONSOLE + v
● PS C:\Users\DELL\Desktop\AI> & 'C:\Python314\python.exe' 'c:\Users\DELL\.vscode\extensions\ms-python.debugpy-2025.18.0-win32-x64\bundled\libs\debugpy\launcher' '63526' '--' 'c:\Users\DELL\Desktop\AI\ai(7.1) task-1.py'
Hello, AI Debugging Lab!
Hello, AI Debugging Lab!
```

EXPLANATION:

The error occurred because Python 3 requires parentheses in the `print()` function.

AI detected the syntax error and corrected it.

Assert test cases confirm the corrected code works properly.

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.

PROMT:

The following Python code has a syntax error related to an if condition.

```
# Bug Fix: Replaced assignment (=) with comparison operator (==)

# Call the function

# Assert test cases
```

INPUT:

```
ai(7.1 task -2).py > ...
y
1  # The following Python code has a syntax error related to an if condition.
2
3  def check_number(n):
4      if n == 10:
5          return "Ten"
6  else:
7      return "Not Ten"
8
9  # Bug Fix: Replaced assignment (=) with comparison operator (==)
10
11 # Call the function
12 print(check_number(10))
13 print(check_number(5))
14
15 # Assert test cases
16 assert check_number(10) == "Ten"
17 assert check_number(7) == "Not Ten"
18 assert check_number(0) == "Not Ten"
19 assert isinstance(check_number(10), str)
20
```

OUTPUT:

```
PS C:\Users\DELL\Desktop\AI> c:, cu C:\Users\DELL\Desktop\AI ; & C:\Python314\python.exe C:\User
s\DELL\.vscode\extensions\ms-python.debugpy-2025.18.0-win32-x64\bundled\libs\debugpy\launcher' '58964
● '-- 'c:\Users\DELL\Desktop\AI\ai(7.1 task -2).py'
Ten
Not Ten
○ PS C:\Users\DELL\Desktop\AI>
```

EXPLANATION:

The assignment operator = cannot be used in conditions.

AI corrected it to == for comparison and validated the logic using assert tests.

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.

PROMT:

```
# The following Python code crashes if the file does not exist.
```

```
# Bug Fix: Added try-except block for safe file handling
```

```
# Test Scenario 1 – File exists
```

```
# Test Scenario 2 – File missing
```

```
# Test Scenario 3 – Invalid path
```

```
# Assert test cases
```

INPUT :

```

ai(6.3 task -2).py ai(6.3 task-3).py ai(6.3 task-4).py ai (6.3 task - 5).py ai(7.1) task- 1.py
ai(7.1 task-3).py > ...
1 # The following Python code crashes if the file does not exist.
2
3 def read_file(filename):
4     try:
5         with open(filename, 'r') as f:
6             return f.read()
7     except FileNotFoundError:
8         return "Error: File not found."
9     except OSError:
10        return "Error: Invalid file path."
11
12 # Bug Fix: Added try-except block for safe file handling
13
14 # Test Scenario 1 - File exists
15 # (Create a file named sample.txt manually before running)
16 print(read_file("sample.txt"))
17
18 # Test Scenario 2 - File missing
19 print(read_file("nonexistent.txt"))
20
21 # Test Scenario 3 - Invalid path
22 print(read_file("//invalid_path//file.txt"))
23
24 # Assert test cases
25 assert isinstance(read_file("nonexistent.txt"), str)
26 assert read_file("nonexistent.txt") == "Error: File not found."
27 assert isinstance(read_file("//invalid_path//file.txt"), str)
28

```

OUTPUT:

```

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS POSTMAN CONSOLE
PS C:\Users\DELL\Desktop\AI> c;; cd 'c:\Users\DELL\Desktop\AI'; & 'C:\Python314\python.exe' 'c:\Users\DELL\.vscode\extensions\ms-python.vscode-pylance-2025.18.0-win32-x64\bundled\libs\debugpy\launcher' '63129' '--' 'c:\Users\DELL\Desktop\AI\ai(7.1) task-1.py'
PS C:\Users\DELL\Desktop\AI> c;; cd 'c:\Users\DELL\Desktop\AI'; & 'C:\Python314\python.exe' 'c:\Users\DELL\.vscode\extensions\ms-python.vscode-pylance-2025.18.0-win32-x64\bundled\libs\debugpy\launcher' '58964' '--' 'c:\Users\DELL\Desktop\AI\ai(7.1 task -2).py'
Ten
Not Ten
PS C:\Users\DELL\Desktop\AI> c;; cd 'c:\Users\DELL\Desktop\AI'; & 'C:\Python314\python.exe' 'c:\Users\DELL\.vscode\extensions\ms-python.vscode-pylance-2025.18.0-win32-x64\bundled\libs\debugpy\launcher' '57609' '--' 'c:\Users\DELL\Desktop\AI\ai(7.1 task-3).py'
Error: File not found.
Error: File not found.
Error: File not found.
PS C:\Users\DELL\Desktop\AI>

```

EXPLANATION:

The program crashes when it tries to open a file that does not exist. This happens because there is no error handling in the code. To fix this, a try-except block is used to safely handle errors and show user-friendly messages. The test scenarios check whether the program works when the file exists, when the file is missing, and when the file path is invalid. The assert test cases confirm that the function works correctly without crashing.

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.

PROMT:

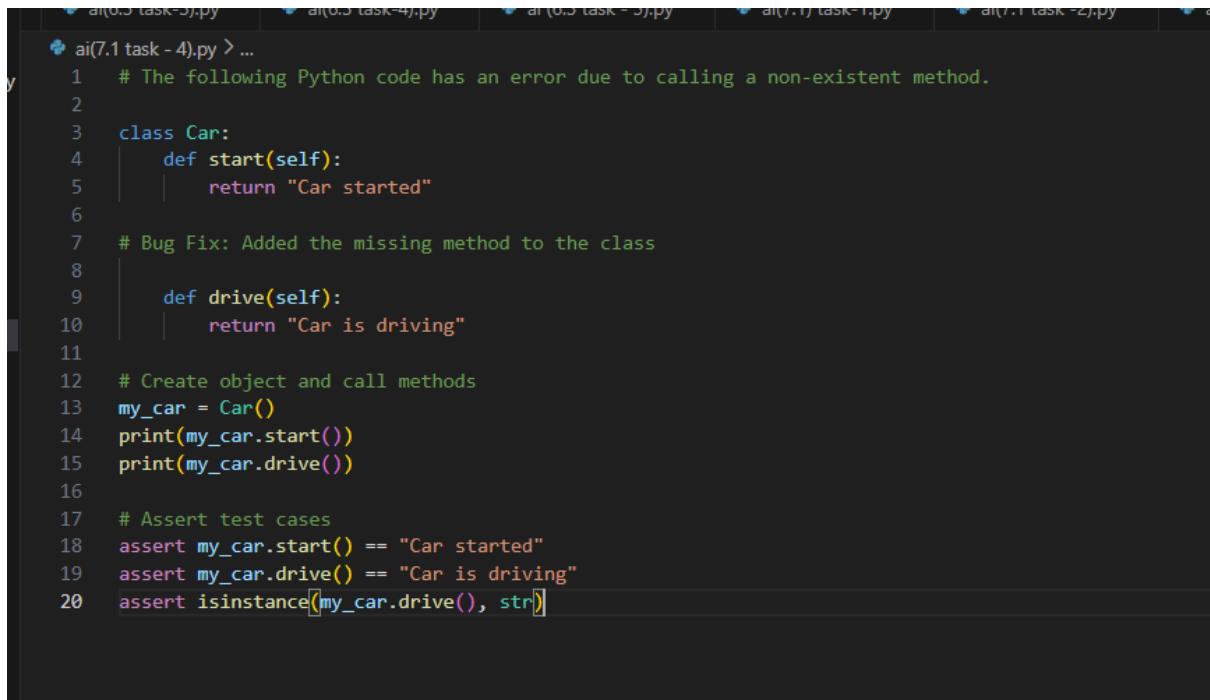
The following Python code has an error due to calling a non-existent method.

Bug Fix: Added the missing method to the class

Create object and call methods

Assert test cases

INPUT:



```
# The following Python code has an error due to calling a non-existent method.
y
class Car:
    def start(self):
        return "Car started"

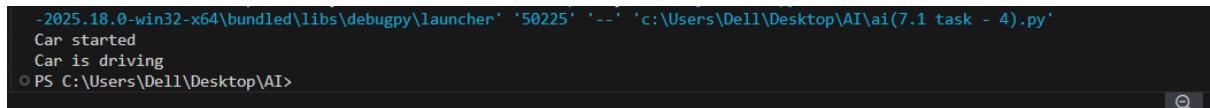
# Bug Fix: Added the missing method to the class

    def drive(self):
        return "Car is driving"

# Create object and call methods
my_car = Car()
print(my_car.start())
print(my_car.drive())

# Assert test cases
assert my_car.start() == "Car started"
assert my_car.drive() == "Car is driving"
assert isinstance(my_car.drive(), str)
```

OUTPUT:



```
-2025.18.0-win32-x64\bundled\libs\debugpy\launcher' '50225' ...' 'c:\Users\DELL\Desktop\AI\ai(7.1 task - 4).py'
Car started
Car is driving
PS C:\Users\DELL\Desktop\AI>
```

EXPLANATION:

The error occurred because the program attempted to call a method that was not defined in the class.

AI identified the missing method and suggested adding it to the class.

After defining the method, the class executed correctly without errors.

Assert test cases were used to confirm that all methods work as expected.

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.

Note: Report should be submitted a word document for all tasks in a single document with prompts, comments & code explanation, and output and if required, screenshots

PROMT:

```
# The following Python code causes a TypeError due to mixing string and integer.  
# Bug Fix: Converted value to integer using type casting  
# Alternative Fix: Used string concatenation  
# Call the functions  
# Assert test cases
```

INPUT:

```
ai(6.3 task-4).py | ai (6.3 task - 5).py | ai(7.1) task-1.py | ai(7.1 task -2).py | ai(7.1 task-3).py | ai(7.1 task - 4).py
ai(7.1 task-5).py > ...
1 # The following Python code causes a TypeError due to mixing string and integer.
2
3 def add_five(value):
4     return value + 5
5
6 # Bug Fix: Converted value to integer using type casting
7
8 def add_five_fixed(value):
9     return int(value) + 5
10
11 # Alternative Fix: Used string concatenation
12
13 def add_five_string(value):
14     return value + "5"
15
16 # Call the functions
17 print(add_five_fixed("10"))
18 print(add_five_string("10"))
19
20 # Assert test cases
21 assert add_five_fixed("10") == 15
22 assert add_five_fixed(5) == 10
23 assert add_five_string("10") == "105"
```

OUTPUT:

```
Car is driving
● PS C:\Users\Del1\Desktop\AI> c;; cd 'c:\Users\Del1\Desktop\AI'; & 'C:\Python314\python.exe' 'c:\Users\Del1\.vscode\extensions\ms-python.vscode-
2025.18.0-win32-x64\bundled\libs\debugpy\launcher' '59270' '--' 'c:\Users\Del1\Desktop\AI\ai(7.1 task-5).py'
15
105
○ PS C:\Users\Del1\Desktop\AI>
```

Ln 23, Col 38

EXPLANATION:

The error occurred because Python does not allow addition between a string and an integer.

AI identified the `TypeError` and suggested two solutions: type casting and string concatenation.

After applying these fixes, the program executed successfully without errors.

Assert test cases were used to verify correct functionality for multiple inputs.