

ASSIGNMENT-7.1

D.Nithya
2303A52457
B-35

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.

The screenshot shows a code editor interface with a dark theme. On the left is a sidebar with various icons. The main area displays a file named 'ai.py' containing the following code:

```

1 # Bug: Missing parentheses in print statement
2 def greet():
3     print "Hello, AI Debugging Lab!"
4
5 greet()
6 # Fixed code with parentheses in print statement
7 def greet():
8     print("Hello, AI Debugging Lab!")

```

A red squiggly underline is under the word 'print' in line 3, indicating a bug. A tooltip above the word 'print' says '# Bug: Missing parentheses in print statement'. Line 7 shows the corrected code where 'print' is now enclosed in parentheses. To the right of the code editor, there is a sidebar titled 'Build with Agent'. It contains a message stating 'AI responses may be inaccurate.' and a link 'Generate Agent Instructions to onboard AI onto your codebase.' Below this is a small input field with placeholder text 'Describe what to bu...' and several small icons.

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.

```

1 # Bug: Using assignment (=) instead of comparison (==)
2 def check_number(n):
3     if n = 10:
4         return "Ten"
5     else:
6         return "Not Ten"
7 # Corrected code
8 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.

The screenshot shows a code editor interface with the following details:

- Title Bar:** ai.py — assignments-ai code — 1 problem in this file
- File List:** Welcome, ai.py (marked with a red dot)
- Code Area:**

```
1  # Bug: Program crashes if file is missing
2  def read_file(filename):
3      with open(filename, 'r') as f:
4          return f.read()
5
6  print(read_file("nonexist"))
7 # Fixed: Added error han
8 def read_file(filename):
    try:
        with open
```

A tooltip above the code area says: "Accept Tab Accept Word ...".
- Right Panel:**
 - Build with Agent:** A section with a speech bubble icon. It says: "AI responses may be inaccurate." and "Generate Agent Instructions to onboard AI onto your codebase."
 - Build Panel:** A sidebar with a search bar: "Describe what to bu" and a list of items: "ai.py +", "Screen Reader Optimized", "Ln 8, Col 25", "Spaces: 4", "UTF-8", "LF", "Python 3.14.2", "Go Live", and a bell icon.
- Bottom Bar:** Includes icons for file operations, a refresh button, and a status bar showing "Screen Reader Optimized", "Ln 8, Col 25", "Spaces: 4", "UTF-8", "LF", "Python 3.14.2", "Go Live", and a bell icon.

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.

```
# Bug: Calling an undefined method
class Car:
    def start(self):
        return "Car started"

my_car = Car()
print(my_car.drive()) # drive() is not defined

# Fix: Define the drive method in the Car class or call an existing method
class Car:
    def start(self):
        return "Car started"

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

Build with Agent

AI responses may be inaccurate.

Generate Agent Instructions to onboard AI onto your codebase.

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.

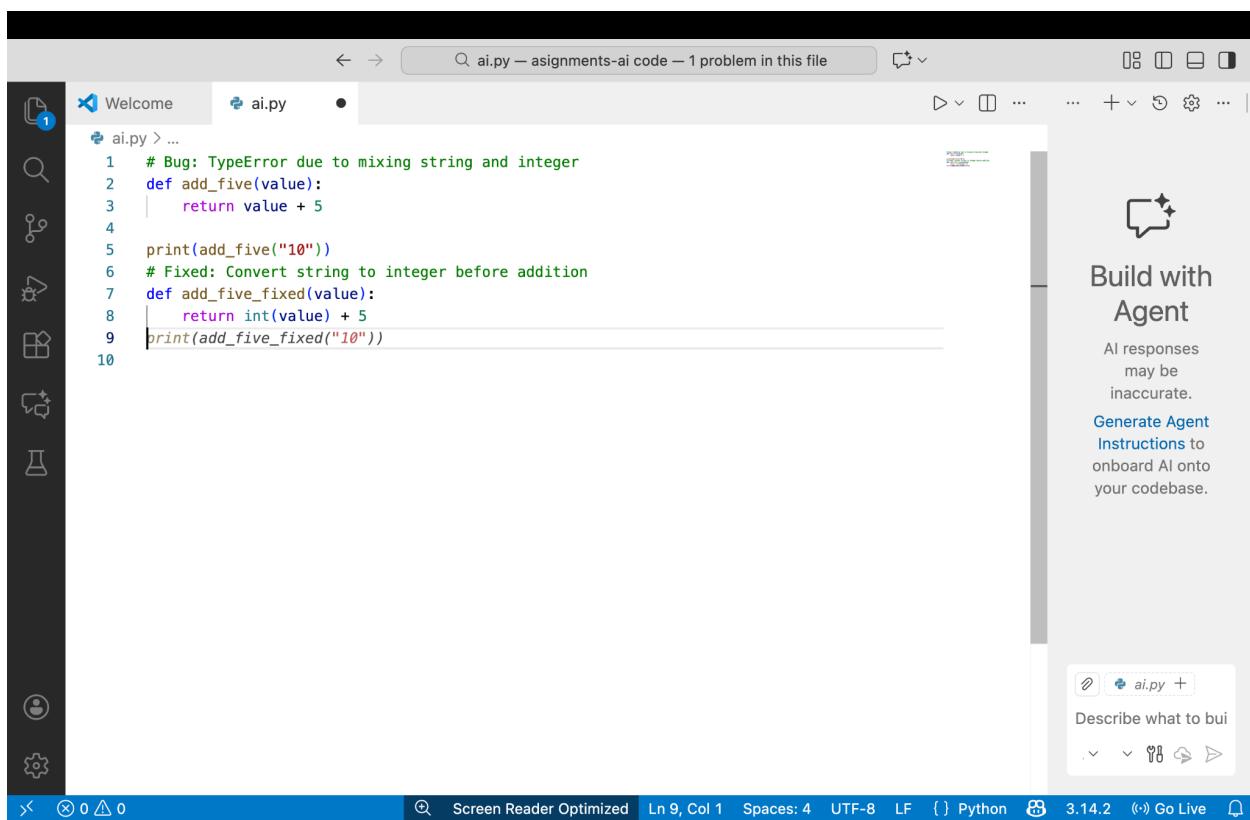
Expected Output #5:

- 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



The screenshot shows a code editor interface with a Python file named 'ai.py' open. The code contains two functions: 'add_five' and 'add_five_fixed'. The 'add_five' function has a comment indicating a bug where a string is mixed with an integer. The 'add_five_fixed' function is a corrected version that converts the input to an integer before performing the addition. The code editor has a sidebar on the right labeled 'Build with Agent' which provides instructions for onboard AI onto the codebase.

```
1  # Bug: TypeError due to mixing string and integer
2  def add_five(value):
3      return value + 5
4
5  print(add_five("10"))
6  # Fixed: Convert string to integer before addition
7  def add_five_fixed(value):
8      return int(value) + 5
9  print(add_five_fixed("10"))
10
```