

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.

```

ai.py > greet
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():
    print("Hello, AI Debugging Lab!")

```

Build with Agent

AI responses may be inaccurate.

[Generate Agent Instructions](#) to onboard AI onto your codebase.

ai.py +

Describe what to build

Screen Reader Optimized Ln 7, Col 13 Spaces: 4 UTF-8 LF {} Python 3.14.2 Go Live

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.

The screenshot shows a code editor with a file named `ai.py`. The code defines a function `check_number` that checks if a number `n` is equal to 10. The original code had a bug: it used the assignment operator `=` instead of the comparison operator `==`. A comment above the function states: `# Bug: Using assignment (=) instead of comparison (==)`. The corrected code is shown below the original code, using `==` for the comparison. The function returns "Ten" if `n == 10` and "Not Ten" otherwise. The editor interface includes a search bar at the top, a sidebar on the left with various icons, and a right sidebar with a "Build with Agent" section. The status bar at the bottom shows "Ln 8, Col 21", "Spaces: 4", "UTF-8", "LF", and "Python 3.14.2".

```

ai.py > check_number
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):
9     if n == 10:
10        return "Ten"
11    else:
12        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"))

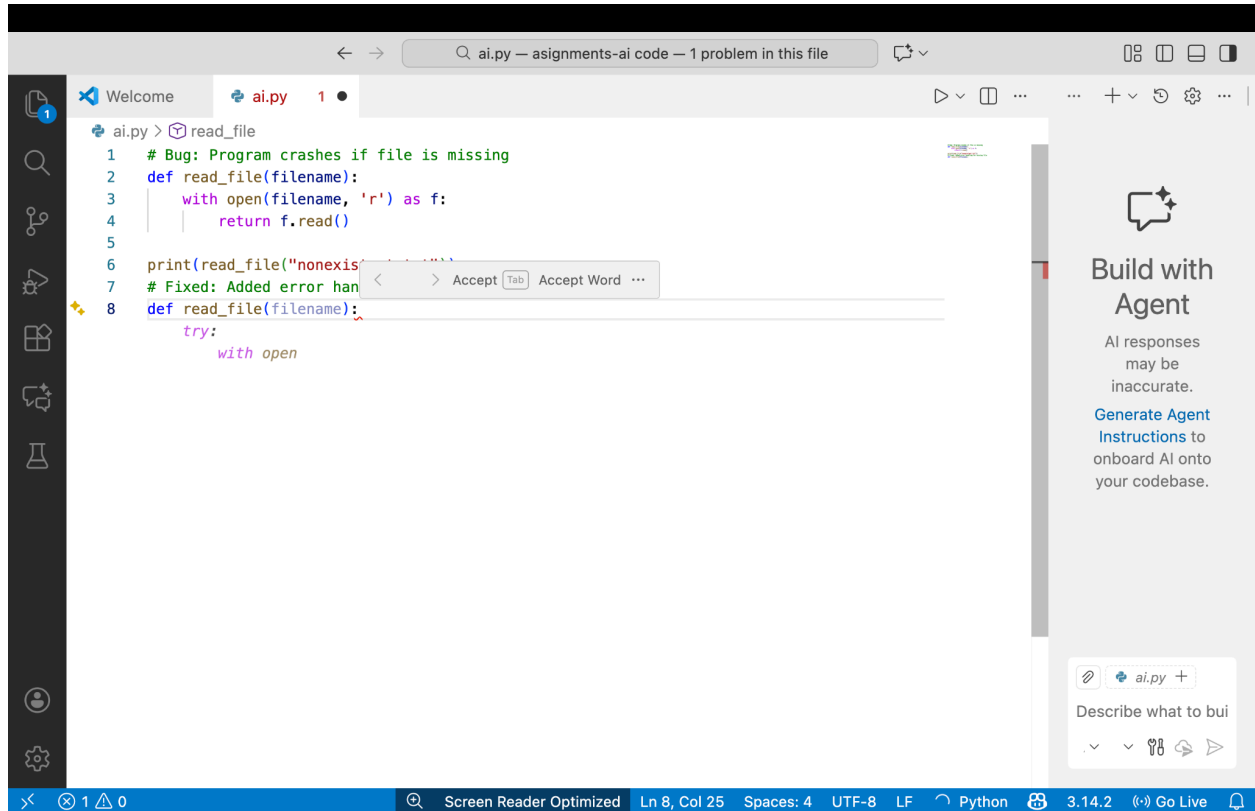
```

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.



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:

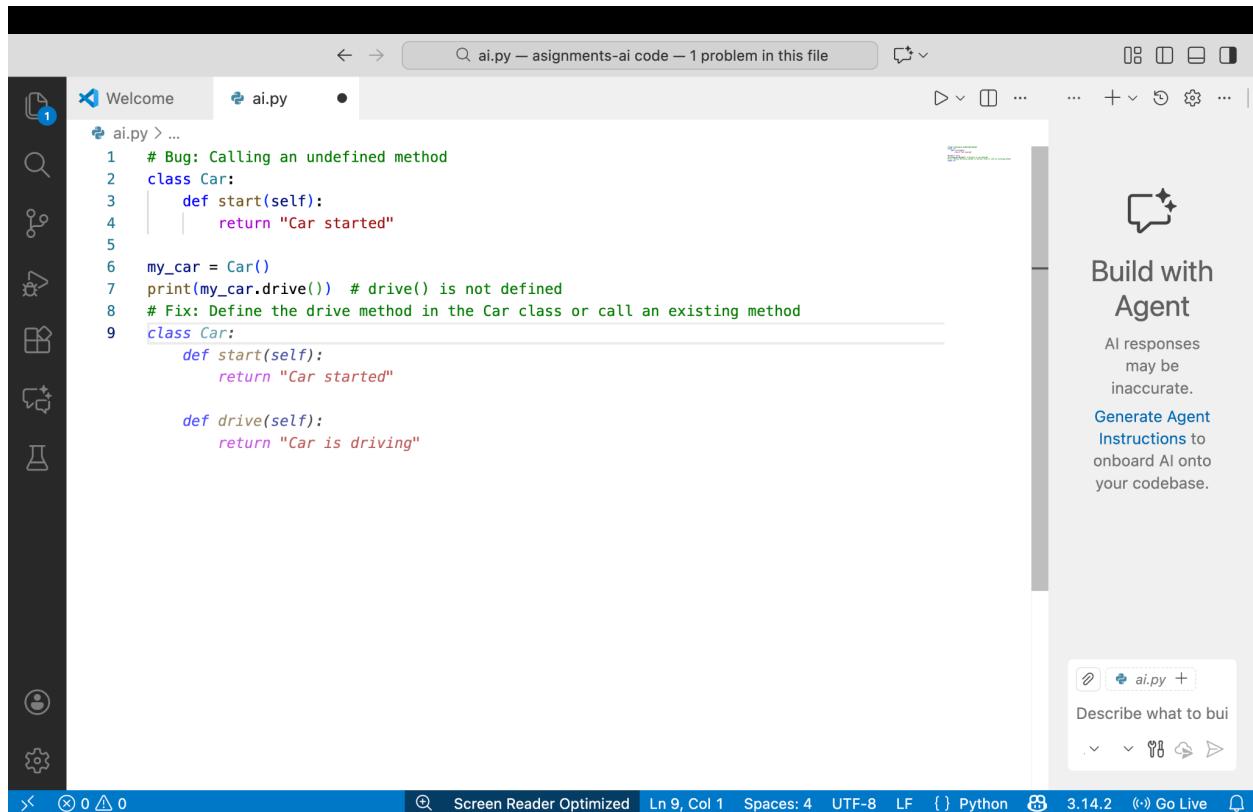
- Students must analyze whether to define the missing method

or correct the method call.

- Use 3 assert tests to confirm the corrected class works.

Expected Output #4:

- Corrected class with clear AI explanation.



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.

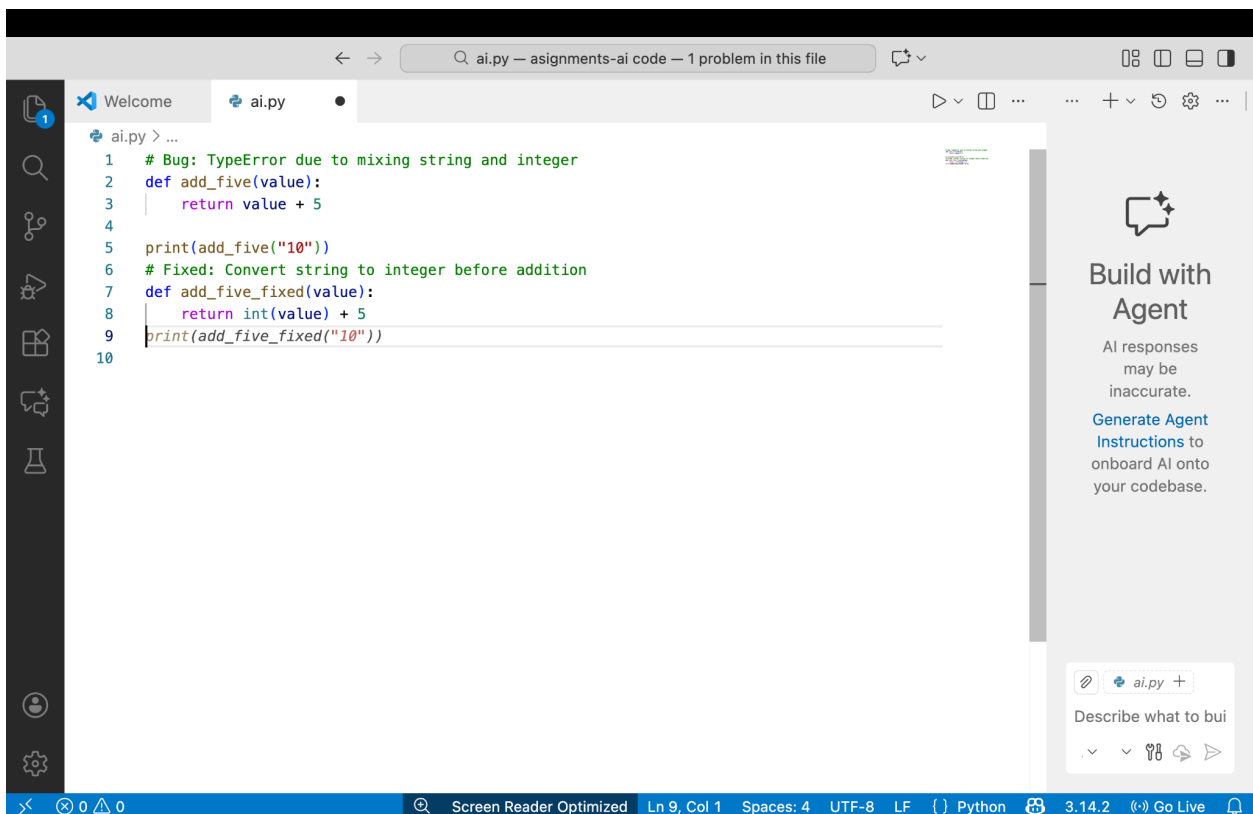
Expected Output #5:

- 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



```
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
```

Build with Agent

AI responses may be inaccurate.

[Generate Agent Instructions](#) to onboard AI onto your codebase.

ai.py +

Describe what to bui

Screen Reader Optimized Ln 9, Col 1 Spaces: 4 UTF-8 LF () Python 3.14.2 Go Live