

AI ASSISTED CODING

Assignment 7.1

N. Akshitha

230352116

Batch 40

1.ask 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.

Code:

```
def greet():
    print("Hello, AI Debugging Lab!")
greet()
```

Output:

```
Hello, AI Debugging Lab!
```

2. 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.

Code:

```
def check_number(n):  
    if n == 10:  
        return "Ten"  
    else:  
        return "Not Ten"  
n=int(input("enter a number: "))  
print(check_number(n))
```

Output:

```
enter a number: 10  
Ten
```

3. ask 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

Code:

```
def read_file(filename):  
    try:  
        with open(filename, 'r') as f:  
            return f.read()  
    except FileNotFoundError:  
        return "File not found"
```

Output:

```
File not found
```

4. 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.

Code:

```
class Car:  
    def start(self):  
        return "Car started"  
my_car = Car()  
print(my_car.start()) # drive() is not defined
```

Output:

```
Car started
```

5. 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.Code:

```
▽ def add_five(value):  
    |     return value + 5  
    | print(add_five(10))
```

Output:

```
15
```