**Assignment-7.1(AI ASSITED CODING)**

**2403A51251**

**B.Rakshith**

**batch:11**

Title: Error Debugging with AI – Systematic Approaches to Finding and Fixing Bugs

Task 1: Syntax Errors – Missing Parentheses in Print Statement

Buggy Code (given):

def greet():

    print "Hello, AI Debugging Lab!"

greet()

AI Explanation:  
Python 3 requires parentheses in print statements.

Corrected Code with Tests:

def greet():

    return "Hello, AI Debugging Lab!"

# Assert test cases

assert greet() == "Hello, AI Debugging Lab!"

print(greet())

My Explanation:  
The bug was due to Python 3 syntax change. After correction, the code works fine and passes the tests.

Output Example:

Hello, AI Debugging Lab!

Task 2: Logic Error – Incorrect Condition in If Statement

Buggy Code (given):

def check\_number(n):

    if n = 10:

        return "Ten"

    else:

        return "Not Ten"

AI Explanation:  
In Python, = is assignment. To compare values, == must be used.

Corrected Code with Tests:

def check\_number(n):

    if n == 10:

        return "Ten"

    else:

        return "Not Ten"

# Assert tests

assert check\_number(10) == "Ten"

assert check\_number(5) == "Not Ten"

assert check\_number(15) == "Not Ten"

My Explanation:  
The bug was a logic error because I used assignment instead of comparison. Now it correctly returns “Ten” only when n is 10.

Output Example:

Ten

Not Ten

Task 3: Runtime Error – File Not Found

Buggy Code (given):

def read\_file(filename):

    with open(filename, 'r') as f:

        return f.read()

print(read\_file("nonexistent.txt"))

AI Explanation:  
Program crashes if file doesn’t exist. Add try-except block.

Corrected Code with Tests:

def read\_file(filename):

    try:

        with open(filename, 'r') as f:

            return f.read()

    except FileNotFoundError:

        return "Error: File not found"

    except Exception as e:

        return f"Unexpected error: {e}"

# Test cases

assert read\_file("nonexistent.txt") == "Error: File not found"

My Explanation:  
The error occurs when a file doesn’t exist. With error handling, the program shows a friendly message instead of crashing.

Output Example:

Error: File not found

Task 4: AttributeError – Calling Non-Existent Method

Buggy Code (given):

class Car:

    def start(self):

        return "Car started"

my\_car = Car()

print(my\_car.drive())

AI Explanation:  
The method drive() does not exist. Replace with start() or define a new method.

Corrected Code with Tests:

class Car:

    def start(self):

        return "Car started"

my\_car = Car()

assert my\_car.start() == "Car started"

print(my\_car.start())

My Explanation:  
The error was calling a method that wasn’t defined. Correcting it to start() fixed the issue.

Output Example:

Car started

Task 5: TypeError – Mixing Strings and Integers

Buggy Code (given):

def add\_five(value):

    return value + 5

print(add\_five("10"))

AI Explanation:  
We cannot add a string and integer directly. Two possible fixes:

Convert string to integer (type casting).

Convert integer to string and concatenate.

Corrected Code – Option 1 (Type Casting):

def add\_five(value):

    return int(value) + 5

assert add\_five("10") == 15

Corrected Code – Option 2 (Concatenation):

def add\_five(value):

    return str(value) + "5"

assert add\_five("10") == "105"

My Explanation:  
The bug happens when adding different data types. With type casting, “10” + 5 = 15. With string concatenation, “10” + “5” = “105”.

Output Example:

15 (Type Casting)

105 (Concatenation)

Conclusion

This lab helped me understand how to detect and fix different types of errors using AI debugging tools. I practiced fixing syntax, logic, runtime, and type errors with AI explanations. I also learned the importance of systematic debugging and validating fixes with assert test cases.