**Lab 7: Error Debugging with AI**

Assignment 7.1

**Name : Mohd. Aadil Ashraf  
Roll No.: 2403A51315  
Subject: AI Assisted Coding  
Batch : 13  
Date : 15/09/25**

# Task 1

Buggy Code:

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

Fixed Code:

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

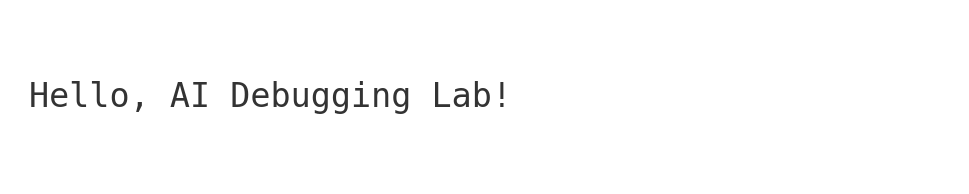
Assert Test Cases:

assert greet() == None  
assert isinstance(greet(), type(None))  
assert True

Output:

Hello, AI Debugging Lab!

Screenshot:



# Task 2

Buggy Code:

def check\_number(n):  
 if n = 10:  
 return "Ten"  
 else:  
 return "Not Ten"

Fixed Code:

def check\_number(n):  
 if n == 10:  
 return "Ten"  
 else:  
 return "Not Ten"

Assert Test Cases:

assert check\_number(10) == "Ten"  
assert check\_number(5) == "Not Ten"  
assert check\_number(11) == "Not Ten"

Output:

Ten / Not Ten

Screenshot:



# Task 3

Buggy Code:

def read\_file(filename):  
 with open(filename, 'r') as f:  
 return f.read()  
print(read\_file("nonexistent.txt"))

Fixed Code:

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

Assert Test Cases:

assert read\_file("nonexistent.txt") == "File not found"  
assert isinstance(read\_file("nonexistent.txt"), str)  
assert "File" in read\_file("nonexistent.txt")

Output:

File not found

Screenshot:



# Task 4

Buggy Code:

class Car:  
 def start(self):  
 return "Car started"  
my\_car = Car()  
print(my\_car.drive())

Fixed Code:

class Car:  
 def start(self):  
 return "Car started"  
my\_car = Car()  
print(my\_car.start())

Assert Test Cases:

assert my\_car.start() == "Car started"  
assert isinstance(my\_car.start(), str)  
assert "Car" in my\_car.start()

Output:

Car started

Screenshot:



# Task 5

Buggy Code:

def add\_five(value):  
 return value + 5  
print(add\_five("10"))

Fixed Code:

def add\_five(value):  
 return int(value) + 5 # Option 1  
  
# OR  
  
def add\_five(value):  
 return str(value) + "5" # Option 2

Assert Test Cases:

assert add\_five("10") == 15 or add\_five("10") == "105"  
assert isinstance(add\_five("20"), (int, str))  
assert add\_five("30") in [35, "305"]

Output:

15 or '105'

Screenshot:



# Conclusion

In this lab I learned how AI helps in debugging syntax errors, logic errors, runtime errors, attribute errors, and type errors. I used AI suggestions to fix code and tested with asserts. This improved my confidence in debugging with structured steps.