2403a51295
T.GOWRI SHANKAR REDDY
Lab 7: Error Debugging with AI

Task Description 1:

Given Code

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

Error Explanation

Python 3 requires parentheses for print(). The code fails with SyntaxError.

Corrected Code

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

# Test cases
assert greet() == "Hello, AI Debugging Lab!"
assert isinstance(greet(), str)
assert "AI Debugging" in greet()
print(greet())
```

Output

Hello, AI Debugging Lab!

Task Description 2:

Given Code

```
def check_number(n):
  if n = 10:
    return "Ten"
  else:
    return "Not Ten"
```

Error Explanation

Using = instead of == causes SyntaxError. '=' is assignment, not comparison.

Corrected Code

```
def check_number(n):
    if n == 10:
        return "Ten"
    else:
        return "Not Ten"

# Test cases
assert check_number(10) == "Ten"
assert check_number(5) == "Not Ten"
assert check_number(-10) == "Not Ten"
print(check_number(10))
```

Output

```
Ten
```

Task Description3:

Given Code

```
def read_file(filename):
    with open(filename, 'r') as f:
        return f.read()

print(read_file("nonexistent.txt"))
```

Error Explanation

If the file doesn’t exist, FileNotFoundError is raised.

Corrected Code

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

# Test cases
assert "not found" in read_file("nonexistent.txt")
assert isinstance(read_file("nonexistent.txt"), str)
assert read_file("invalid_path/abc.txt").startswith("Error")
```

Output

```
Error: File 'nonexistent.txt' not found.
```

Task Description4:

Given Code

```
class Car:
    def start(self):
        return "Car started"

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

Error Explanation

drive() is not defined. Either call start() or define drive().

Corrected Code

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

my_car = Car()

# Test cases
assert my_car.start() == "Car started"
assert my_car.drive() == "Car is driving"
assert isinstance(my_car.start(), str)
print(my_car.start(), "and", my_car.drive())
```

Output

```
Car started and Car is driving
```

Task Description5:

Given Code

```
def add_five(value):
   return value + 5

print(add_five("10"))
```

Error Explanation

Python does not allow adding str and int. Fix by casting or concatenation.

Corrected Code

```
# Solution 1: Type casting
def add_five(value):
   return int(value) + 5

# Test cases
assert add_five("10") == 15
assert add_five(20) == 25
assert add_five(0) == 5
print(add_five("10"))

# Solution 2: String concatenation
def add_five(value):
   return str(value) + "5"

# Test cases
assert add_five("10") == "105"
assert add_five(20) == "205"
assert add_five("AI") == "AI5"
print(add_five("10"))
```

Output

```
15
105
```