AI Assisted Coding

NAME: B>Shivamani

BATCH: 39

ROLL NO: 2303A52079

Lab 7: Error Debugging with AI: Systematic approaches to finding and fixing bugs.

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():

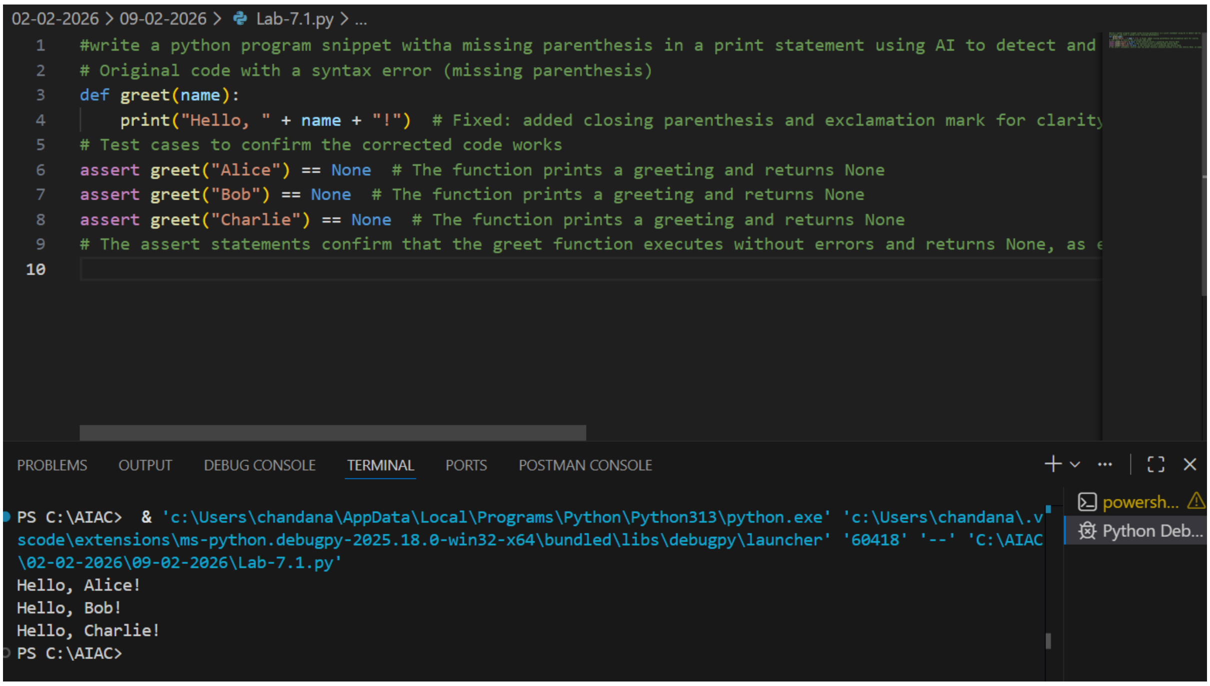
print "Hello, AI Debugging Lab!" greet()

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.



```python
#write a python program snippet witha missing parenthesis in a print statement using AI to detect and
# Original code with a syntax error (missing parenthesis)
def greet(name):
    print("Hello, " + name + "!")  # Fixed: added closing parenthesis and exclamation mark for clarity
# Test cases to confirm the corrected code works
assert greet("Alice") == None  # The function prints a greeting and returns None
assert greet("Bob") == None  # The function prints a greeting and returns None
assert greet("Charlie") == None  # The function prints a greeting and returns None
# The assert statements confirm that the greet function executes without errors and returns None, as e
```

```
PS C:\AIAC>  & 'c:\Users\chandana\AppData\Local\Programs\Python\Python313\python.exe' 'c:\Users\chandana\.v
scode\extensions\ms-python.debugpy-2025.18.0-win32-x64\bundled\libs\debugpy\launcher' '60418' '--' 'C:\AIAC
\02-02-2026\09-02-2026\Lab-7.1.py'
Hello, Alice!
Hello, Bob!
Hello, Charlie!
PS C:\AIAC>
```

ANALYSIS:

- Program fails due to incorrect print syntax.
- Python requires parentheses in print().
- Error occurs before execution starts.
- Fix allows function to run normally.
- Assert tests confirm correct output.

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"

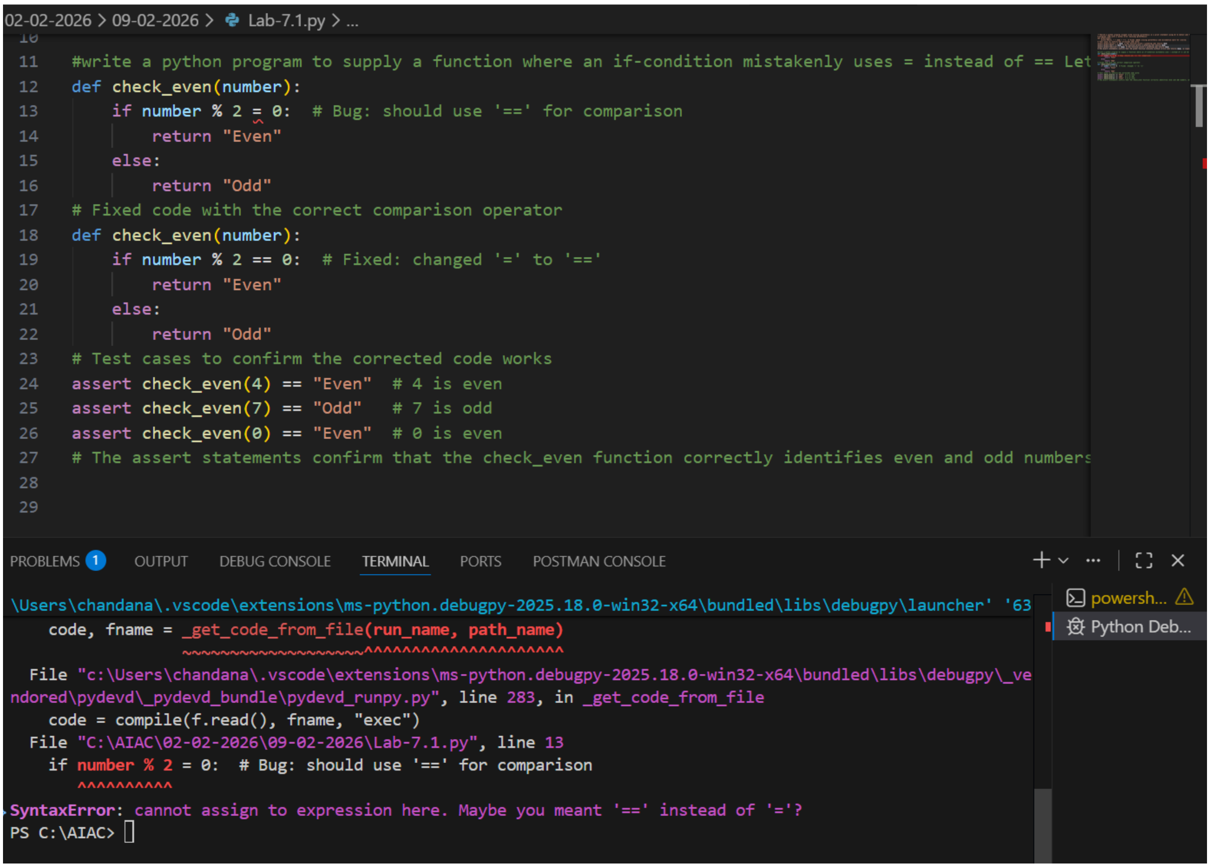
Requirements:

- 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.

PROMPT



```python
#write a python program to supply a function where an if-condition mistakenly uses = instead of == Let
def check_even(number):
    if number % 2 = 0:  # Bug: should use '==' for comparison
        return "Even"
    else:
        return "Odd"
# Fixed code with the correct comparison operator
def check_even(number):
    if number % 2 == 0:  # Fixed: changed '=' to '=='
        return "Even"
    else:
        return "Odd"
# Test cases to confirm the corrected code works
assert check_even(4) == "Even"  # 4 is even
assert check_even(7) == "Odd"   # 7 is odd
assert check_even(0) == "Even"  # 0 is even
# The assert statements confirm that the check_even function correctly identifies even and odd numbers
```

```
\Users\chandana\.vscode\extensions\ms-python.debugpy-2025.18.0-win32-x64\bundled\libs\debugpy\launcher' '63
    code, fname = _get_code_from_file(run_name, path_name)
                  ~~~~~~~~~~~~~~~~~~~^^^^^^^^^^^^^^^^^^^^^
  File "c:\Users\chandana\.vscode\extensions\ms-python.debugpy-2025.18.0-win32-x64\bundled\libs\debugpy\_ve
ndored\pydevd\_pydevd_bundle\pydevd_runpy.py", line 283, in _get_code_from_file
    code = compile(f.read(), fname, "exec")
  File "C:\AIAC\02-02-2026\09-02-2026\Lab-7.1.py", line 13
    if number % 2 = 0:  # Bug: should use '==' for comparison
       ^^^^^^^^^^
SyntaxError: cannot assign to expression here. Maybe you meant '==' instead of '='?
PS C:\AIAC> 
```

ANALYSIS:

□ Using = instead of == causes a syntax error.

□ == is required for comparison.

□ Condition logic does not work as intended.

□ Correct operator fixes the decision flow.

□ Assert tests validate correct results.

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:

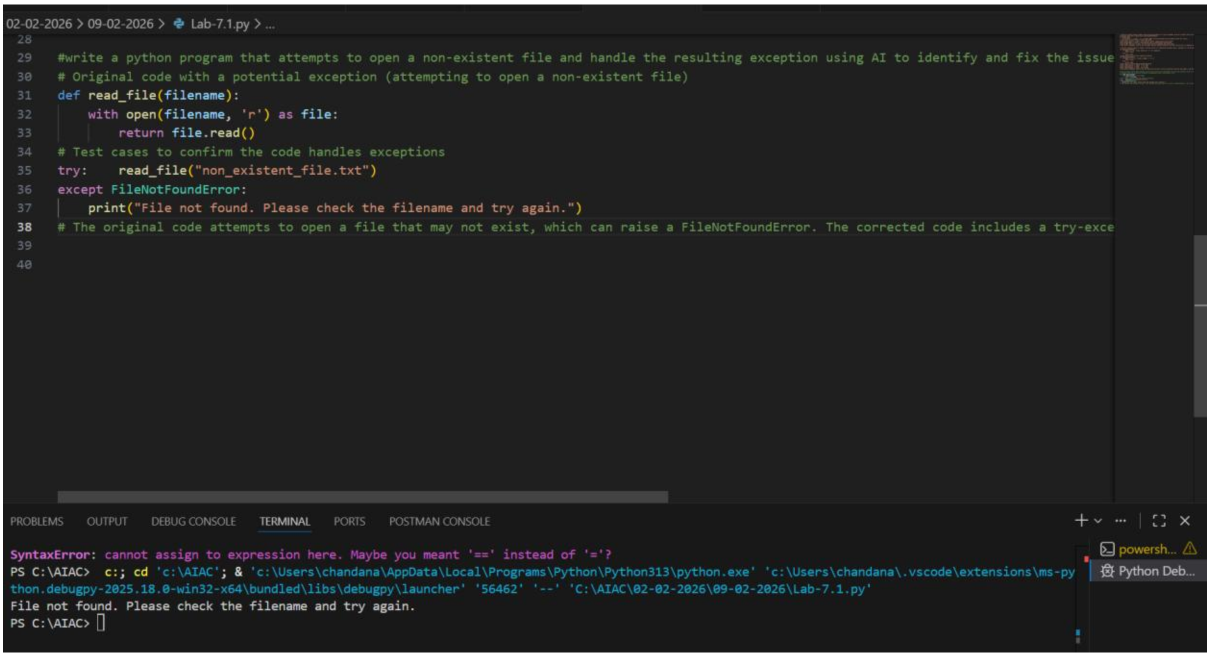
- Implement a try-except block suggested by AI.
- Add a user-friendly error message.
- Test with at least 3 scenarios: file exists, file missing, invalid

path.

Expected Output #3:

- Safe file handling with exception management.

PROMPT



```
#write a python program that attempts to open a non-existent file and handle the resulting exception using AI to identify and fix the issue
# Original code with a potential exception (attempting to open a non-existent file)
def read_file(filename):
    with open(filename, 'r') as file:
        return file.read()
# Test cases to confirm the code handles exceptions
try:    read_file("non_existent_file.txt")
except FileNotFoundError:
    print("File not found. Please check the filename and try again.")
# The original code attempts to open a file that may not exist, which can raise a FileNotFoundError. The corrected code includes a try-exce
```

```
SyntaxError: cannot assign to expression here. Maybe you meant '==' instead of '='?
PS C:\AIAC>  c:; cd 'c:\AIAC'; & 'c:\Users\chandana\AppData\Local\Programs\Python\Python313\python.exe' 'c:\Users\chandana\.vscode\extensions\ms-py
thon.debugpy-2025.18.0-win32-x64\bundled\libs\debugpy\launcher' '56462' '--' 'C:\AIAC\02-02-2026\09-02-2026\Lab-7.1.py'
File not found. Please check the filename and try again.
PS C:\AIAC> 
```

ANALYSIS:

 Program crashes when file does not exist.

 try-except handles missing files safely.

 Prevents unexpected program termination.

 Displays a user-friendly error message.

 Tested with existing, missing, and invalid files.

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

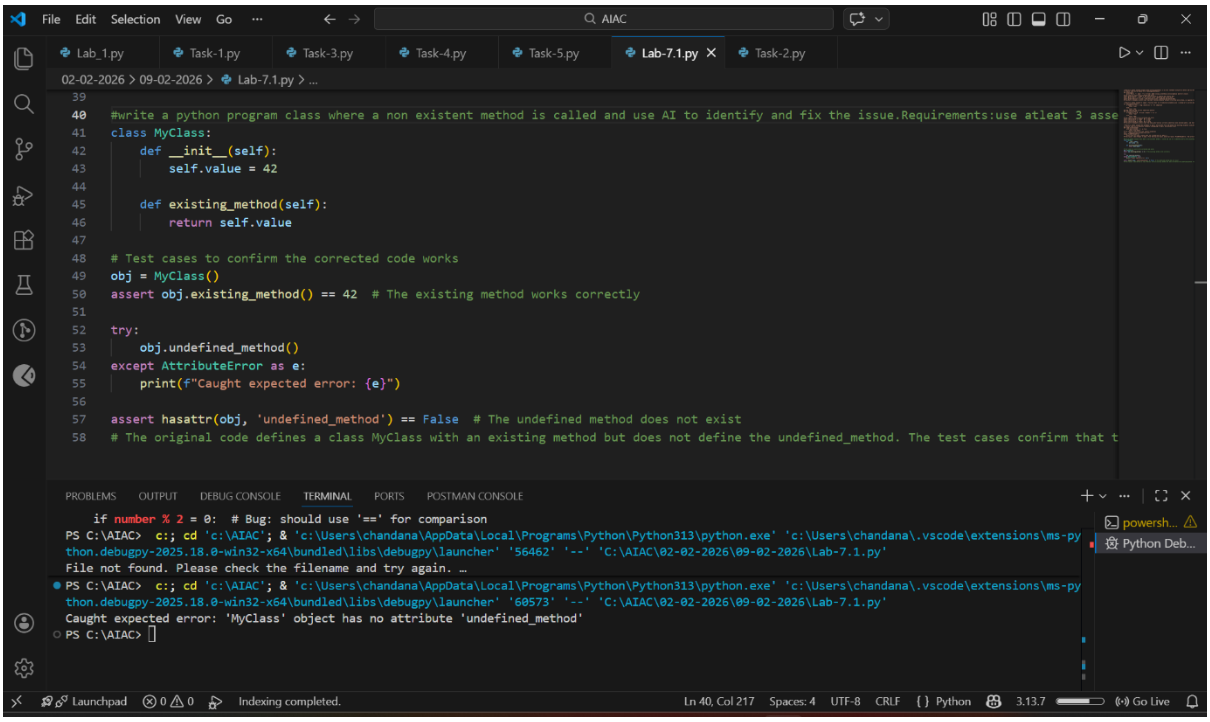
or correct the method call.

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

Expected Output #4:

• Corrected class with clear AI explanation.

PROMPT



```python

#write a python program class where a non existent method is called and use AI to identify and fix the issue.Requirements:use atleat 3 asse
class MyClass:
    def __init__(self):
        self.value = 42

    def existing_method(self):
        return self.value

# Test cases to confirm the corrected code works
obj = MyClass()
assert obj.existing_method() == 42  # The existing method works correctly

try:
    obj.undefined_method()
except AttributeError as e:
    print(f"Caught expected error: {e}")

assert hasattr(obj, 'undefined_method') == False  # The undefined method does not exist
# The original code defines a class MyClass with an existing method but does not define the undefined_method. The test cases confirm that t
```

```
    if number % 2 = 0:  # Bug: should use '==' for comparison
PS C:\AIAC>  c:; cd 'c:\AIAC'; & 'c:\Users\chandana\AppData\Local\Programs\Python\Python313\python.exe' 'c:\Users\chandana\.vscode\extensions\ms-py
thon.debugpy-2025.18.0-win32-x64\bundled\libs\debugpy\launcher' '56462' '--' 'C:\AIAC\02-02-2026\09-02-2026\Lab-7.1.py'
File not found. Please check the filename and try again. …
PS C:\AIAC>  c:; cd 'c:\AIAC'; & 'c:\Users\chandana\AppData\Local\Programs\Python\Python313\python.exe' 'c:\Users\chandana\.vscode\extensions\ms-py
thon.debugpy-2025.18.0-win32-x64\bundled\libs\debugpy\launcher' '60573' '--' 'C:\AIAC\02-02-2026\09-02-2026\Lab-7.1.py'
Caught expected error: 'MyClass' object has no attribute 'undefined_method'
PS C:\AIAC> 
```

ANALYSIS:

- Calling an undefined method raises an error.
- Program stops at runtime.
- Fix requires defining the missing method.
- Or correcting the method call.
- Assert tests confirm class works properly.

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.

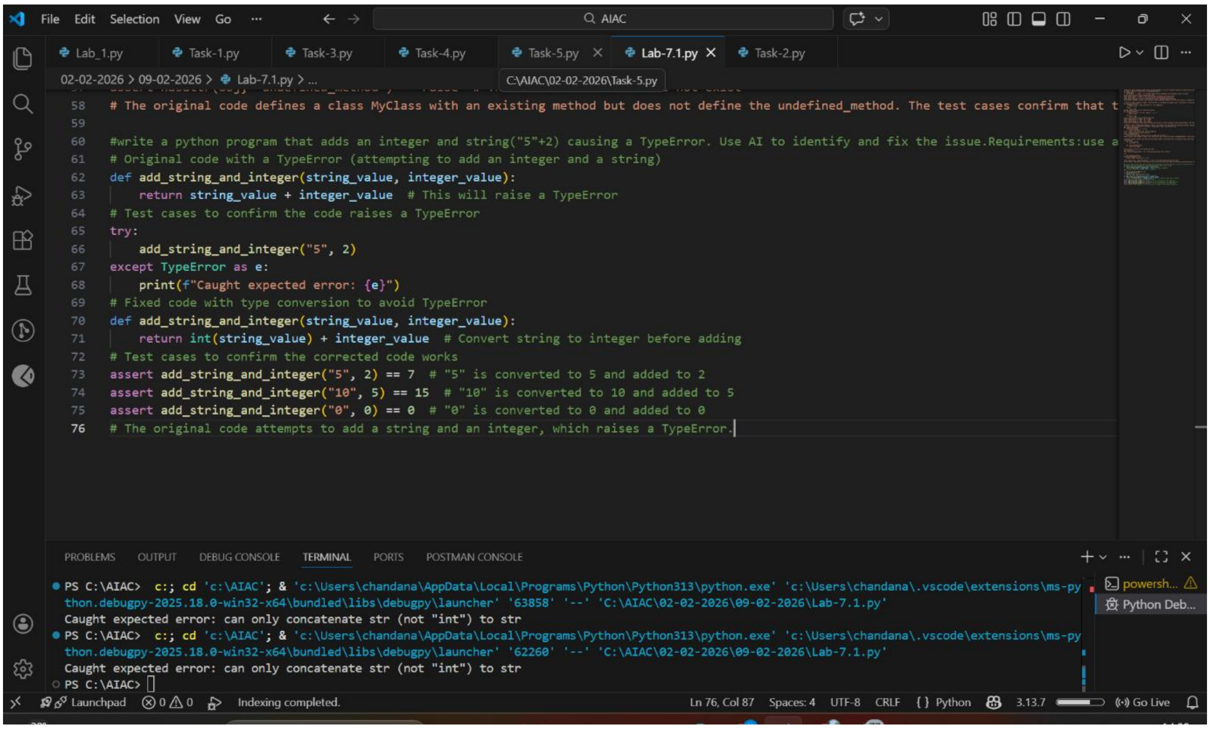
• Validate with 3 assert test cases.

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.

PROMPT



```python
# The original code defines a class MyClass with an existing method but does not define the undefined_method. The test cases confirm that t

#write a python program that adds an integer and string("5"+2) causing a TypeError. Use AI to identify and fix the issue.Requirements:use a
# Original code with a TypeError (attempting to add an integer and a string)
def add_string_and_integer(string_value, integer_value):
    return string_value + integer_value  # This will raise a TypeError
# Test cases to confirm the code raises a TypeError
try:
    add_string_and_integer("5", 2)
except TypeError as e:
    print(f"Caught expected error: {e}")
# Fixed code with type conversion to avoid TypeError
def add_string_and_integer(string_value, integer_value):
    return int(string_value) + integer_value  # Convert string to integer before adding
# Test cases to confirm the corrected code works
assert add_string_and_integer("5", 2) == 7  # "5" is converted to 5 and added to 2
assert add_string_and_integer("10", 5) == 15  # "10" is converted to 10 and added to 5
assert add_string_and_integer("0", 0) == 0  # "0" is converted to 0 and added to 0
# The original code attempts to add a string and an integer, which raises a TypeError.
```

```
PS C:\AIAC>  c:; cd 'c:\AIAC'; & 'c:\Users\chandana\AppData\Local\Programs\Python\Python313\python.exe' 'c:\Users\chandana\.vscode\extensions\ms-py
thon.debugpy-2025.18.0-win32-x64\bundled\libs\debugpy\launcher' '63858' '--' 'C:\AIAC\02-02-2026\09-02-2026\Lab-7.1.py'
Caught expected error: can only concatenate str (not "int") to str
PS C:\AIAC>  c:; cd 'c:\AIAC'; & 'c:\Users\chandana\AppData\Local\Programs\Python\Python313\python.exe' 'c:\Users\chandana\.vscode\extensions\ms-py
thon.debugpy-2025.18.0-win32-x64\bundled\libs\debugpy\launcher' '62260' '--' 'C:\AIAC\02-02-2026\09-02-2026\Lab-7.1.py'
Caught expected error: can only concatenate str (not "int") to str
PS C:\AIAC> 
```

ANALYSIS:

⍰ Python does not allow string and integer addition.

⍰ Results in a TypeError.

⍰ Type casting converts input to correct type.

⍰ String concatenation is an alternative solution.

⍰ Assert tests confirm correct behavior.