ASSIGNMENT- 7.1

Name : T.Rakshitha

H.T.NO: 2303A51172

Batch : 18

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.

Promt :

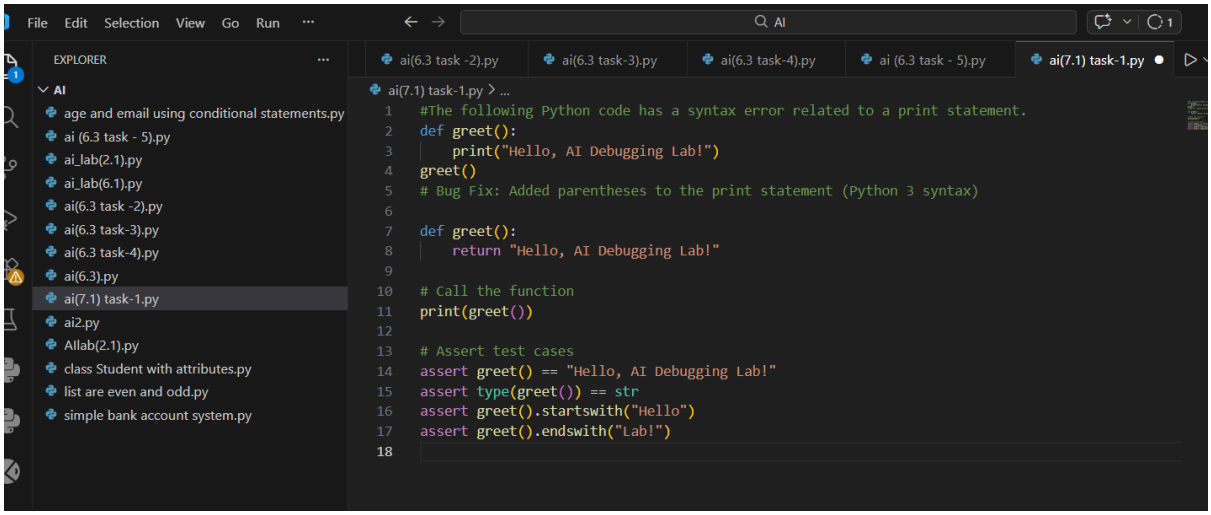
#The following Python code has a syntax error related to a print statement.

Bug Fix: Added parentheses to the print statement (Python 3 syntax)

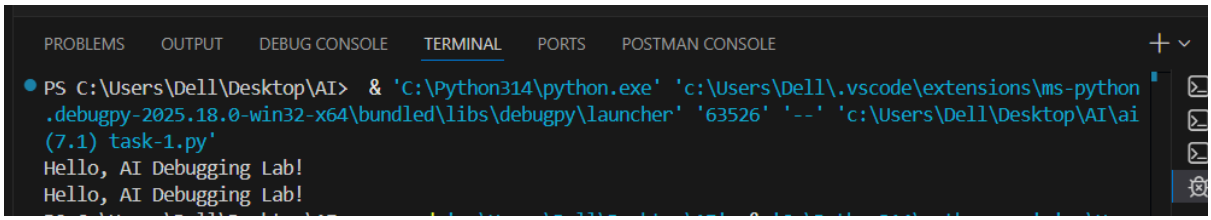
Call the function

Assert test cases

INPUT:



OUTPUT:



EXPLANATION:

The error occurred because Python 3 requires parentheses in the print() function.

AI detected the syntax error and corrected it.

Assert test cases confirm the corrected code works properly.

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.

PROMT:

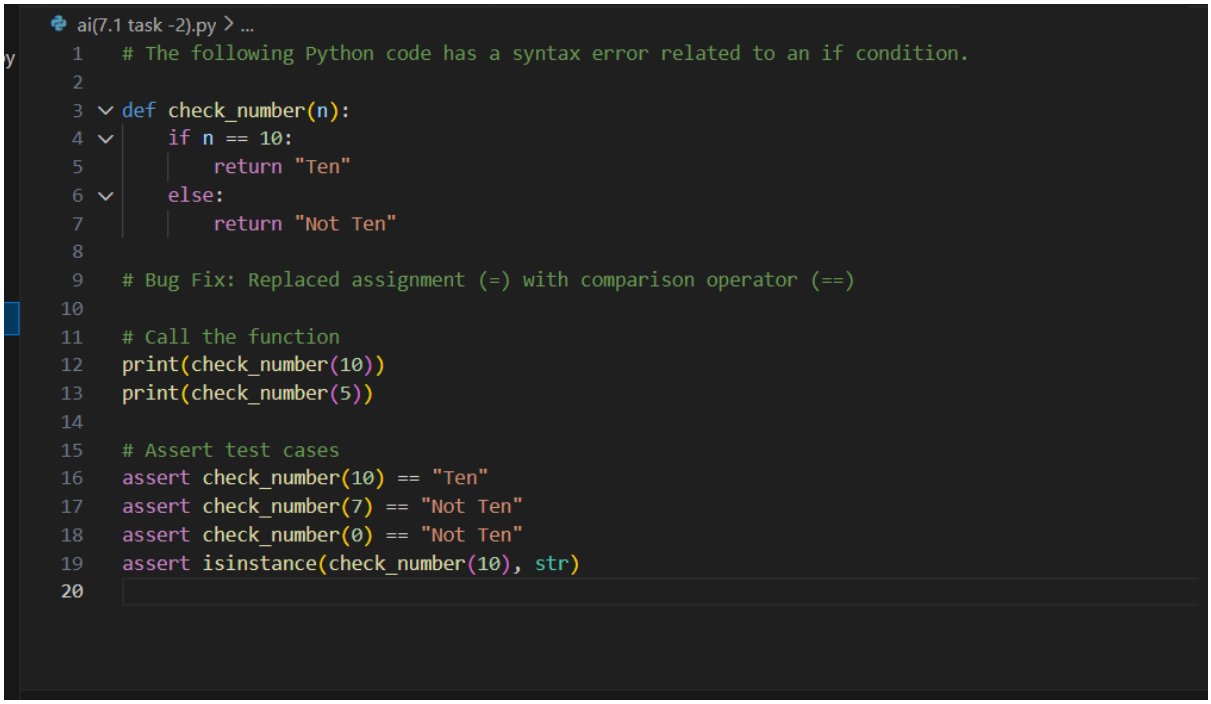
The following Python code has a syntax error related to an if condition.

Bug Fix: Replaced assignment (=) with comparison operator (==)

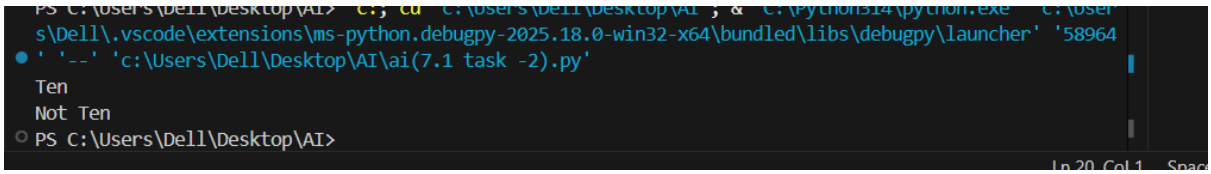
Call the function

Assert test cases

INPUT:



OUTPUT:



EXPLANATION:

The assignment operator = cannot be used in conditions.

AI corrected it to == for comparison and validated the logic using assert tests.

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.

PROMT:

```
# The following Python code crashes if the file does not exist.
# Bug Fix: Added try-except block for safe file handling
# Test Scenario 1 – File exists
# Test Scenario 2 – File missing
# Test Scenario 3 – Invalid path
# Assert test cases
```

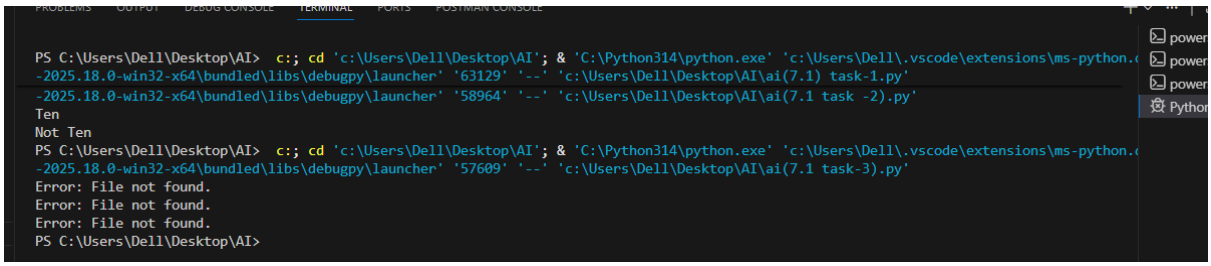
INPUT :

ai(6.3 task -2).py | ai(6.3 task-3).py | ai(6.3 task-4).py | ai (6.3 task - 5).py | ai(7.1) task-1.py

ai(7.1 task-3).py > ...

```python
# The following Python code crashes if the file does not exist.

def read_file(filename):
    try:
        with open(filename, 'r') as f:
            return f.read()
    except FileNotFoundError:
        return "Error: File not found."
    except OSError:
        return "Error: Invalid file path."

# Bug Fix: Added try-except block for safe file handling

# Test Scenario 1 - File exists
# (Create a file named sample.txt manually before running)
print(read_file("sample.txt"))

# Test Scenario 2 - File missing
print(read_file("nonexistent.txt"))

# Test Scenario 3 - Invalid path
print(read_file("///invalid_path///file.txt"))

# Assert test cases
assert isinstance(read_file("nonexistent.txt"), str)
assert read_file("nonexistent.txt") == "Error: File not found."
assert isinstance(read_file("///invalid_path///file.txt"), str)

```

OUTPUT:



EXPLANATION:

The program crashes when it tries to open a file that does not exist. This happens because there is no error handling in the code. To fix this, a try-except block is used to safely handle errors and show user-friendly messages. The test scenarios check whether the program works when the file exists, when the file is missing, and when the file path is invalid. The assert test cases confirm that the function works correctly without crashing.

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.

PROMT:

The following Python code has an error due to calling a non-existent method.

Bug Fix: Added the missing method to the class

Create object and call methods

Assert test cases

INPUT:

```python
# The following Python code has an error due to calling a non-existent method.

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

# Bug Fix: Added the missing method to the class

    def drive(self):
        return "Car is driving"

# Create object and call methods
my_car = Car()
print(my_car.start())
print(my_car.drive())

# Assert test cases
assert my_car.start() == "Car started"
assert my_car.drive() == "Car is driving"
assert isinstance(my_car.drive(), str)
```

OUTPUT:

```
-2025.18.0-win32-x64\bundled\libs\debugpy\launcher' '50225' '--' 'c:\Users\Dell\Desktop\AI\ai(7.1 task - 4).py'
Car started
Car is driving
PS C:\Users\Dell\Desktop\AI>
```

EXPLANATION:

The error occurred because the program attempted to call a method that was not defined in the class.

AI identified the missing method and suggested adding it to the class.

After defining the method, the class executed correctly without errors.

Assert test cases were used to confirm that all methods work as expected.

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

PROMT:

The following Python code causes a TypeError due to mixing string and integer.

Bug Fix: Converted value to integer using type casting

Alternative Fix: Used string concatenation

Call the functions

Assert test cases

INPUT:

```python
# The following Python code causes a TypeError due to mixing string and integer.

def add_five(value):
    return value + 5

# Bug Fix: Converted value to integer using type casting

def add_five_fixed(value):
    return int(value) + 5

# Alternative Fix: Used string concatenation

def add_five_string(value):
    return value + "5"

# Call the functions
print(add_five_fixed("10"))
print(add_five_string("10"))

# Assert test cases
assert add_five_fixed("10") == 15
assert add_five_fixed(5) == 10
assert add_five_string("10") == "105"
```

OUTPUT:

```
Car is driving
PS C:\Users\Dell\Desktop\AI>  c:; cd 'c:\Users\Dell\Desktop\AI'; & 'C:\Python314\python.exe' 'c:\Users\Dell\.vscode\extensions\ms-python.
-2025.18.0-win32-x64\bundled\libs\debugpy\launcher' '59270' '--' 'c:\Users\Dell\Desktop\AI\ai(7.1 task-5).py'
15
105
PS C:\Users\Dell\Desktop\AI>
```

EXPLANATION:

The error occurred because Python does not allow addition between a string and an integer.

AI identified the TypeError and suggested two solutions: type casting and string concatenation.

After applying these fixes, the program executed successfully without errors.

Assert test cases were used to verify correct functionality for multiple inputs.