AI ASSISTED CODING

LAB ASSIGNMENT 7.1

ERROR DEBUGGING WITH AI

2303A51165

BATCH – 18

Task 1: Syntax Error – Missing Parentheses

QUESTION

Provide a Python snippet with missing parentheses in print statement. Detect and fix using AI.

PROMPT

Identify syntax error caused by missing parentheses and correct the function. Verify using test cases.

CODE

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

greet()

assert greet() is None
```

def greet():

print("Hello, AI Debugging Lab!")

greet()

assert greet() is None

OUTPUT

```
PS D:\3-2 SEM> python -u "d:\3-2 SEM\AI ASSISTED\ASSVSCODE\
Hello, AI Debugging Lab!
Hello, AI Debugging Lab!
PS D:\3-2 SEM>
```

QUESTION PROMPT CODE OUTPUT EXPLANATION

EXPLANATION

Python 3 requires parentheses in print statements. Adding parentheses resolves syntax error.

TASK 2: INCORRECT CONDITION IN IF STATEMENT

QUESTION

Fix code where = is used instead of ==.

PROMPT

Explain why assignment operator causes error and correct the function.

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(0) == "Not Ten"

print("All tests passed")
```

def check_number(n):

if n == 10:

return "Ten"

else:

return "Not Ten"

 # Test cases

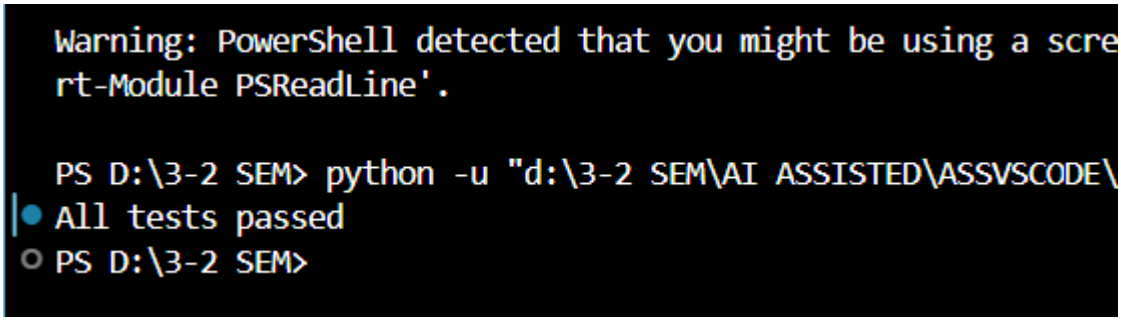
assert check_number(10) == "Ten"

assert check_number(5) == "Not Ten"

```
assert check_number(0) == "Not Ten"

print("All tests passed")
```

OUTPUT



EXPLANATION

Single equals is assignment operator. Using == performs comparison correctly.

TASK 3: RUNTIME ERROR – FILE NOT FOUND

QUESTION

Fix program that crashes when file does not exist.

PROMPT

Implement try-except block for safe file handling.

CODE

```
def read_file(filename):
    try:
        with open(filename,'r') as f:
            return f.read()
    except FileNotFoundError:
        return "File not found"
print(read_file("test.txt"))
print(read_file("missing.txt"))
```

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

print(read_file("test.txt"))
print(read_file("missing.txt"))
```

OUTPUT

```
Warning: PowerShell detected that you might be using a scre
urposes. If you want to re-enable it, run 'Import-Module PS

PS D:\3-2 SEM> python -u "d:\3-2 SEM\AI ASSISTED\ASSVSCODE\
File not found
File not found
PS D:\3-2 SEM>
```

EXPLANATION

Try-except prevents crash and provides user-friendly error message.

Task 4: Calling Non-Existent Method

QUESTION

Fix class where undefined method is called.

PROMPT

Analyze whether to define method or correct call.

Code

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

my_car = Car()

assert my_car.start() == "Car started"

assert my_car.drive() == "Car is driving"

print(“Assertions passed”)

```
class Car:
    def start(self):
        return "Car started"
    def drive(self):
        return "Car is driving"
my_car = Car()
assert my_car.start() == "Car started"
assert my_car.drive() == "Car is driving"
print("Assertion passed")
```

OUTPUT

```
PS D:\3-2 SEM> python -u
Assertion passed
```

Assertions passed

EXPLANATION

Undefined method caused AttributeError. Adding method resolves issue.

Task 5: TypeError – Mixing String and Integer

QUESTION

Fix code adding string and integer.

PROMPT

Provide solutions using type casting.

Code

def add_five(value):

    return int(value) + 5

assert add_five("10") == 15

assert add_five(5) == 10

assert add_five("0") == 5

print(“Assertions passed”)

```
class Car:
    def start(self):
        return "Car started"
    def drive(self):
        return "Car is driving"
my_car = Car()
assert my_car.start() == "Car started"
assert my_car.drive() == "Car is driving"
print("Assertion passed")
```

OUTPUT

```
PS D:\3-2 SEM> python -u
Assertion passed
```

Assertions passed

EXPLANATION

Type casting ensures correct numeric addition.