

AI ASS 7.1

Name:B.Sanjana

H.T.No:2303A52306

Batch:43

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.

The screenshot shows a Jupyter Notebook interface with a file named 'AI ass 7.1.ipynb'. The code cell contains the following Python code:

```
#1 Bug: Missing parentheses in print statement
def greet():
    print ("Hello, AI Debugging Lab!")
greet()
#3 assert test cases
assert greet() == None
assert str(greet()) == "None"
assert type(greet()) == type(None)
```

The output cell shows the execution results:

```
... Hello, AI Debugging Lab!
Hello, AI Debugging Lab!
Hello, AI Debugging Lab!
Hello, AI Debugging Lab!
```

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

The screenshot shows a Jupyter Notebook interface with a single code cell containing Python code. The code defines a function `check_number` that returns "Ten" if `n == 10`, and "Not Ten" otherwise. It also contains three assertions: `assert check_number(10) == "Ten"`, `assert check_number(5) == "Not Ten"`, and `assert check_number(0) == "Not Ten"`. When run, the code prints the results of each assertion: "Ten", "Not Ten", and "Not Ten". The notebook has a sidebar with various icons for file operations, and the top right corner shows memory usage (RAM and Disk).

```
#2 Bug: Using assignment (=) instead of comparison (==)
def check_number(n):
    if n == 10:
        return "Ten"
    else:
        return "Not Ten"

# 3 asserts
assert check_number(10) == "Ten"
assert check_number(5) == "Not Ten"
assert check_number(0) == "Not Ten"
print(check_number(10))
print(check_number(5))
print(check_number(0))

...
Ten
Not Ten
Not Ten
```

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.

The screenshot shows a Jupyter Notebook interface with the following details:

- Title:** AI ass 7.1.ipynb
- Toolbar:** File, Edit, View, Insert, Runtime, Tools, Help.
- Cell Type:** Code (highlighted).
- Code Content:**

```
#3 Bug: Program crashes if file is missing
def read_file(filename):
    try:
        with open(filename, 'r') as f:
            return f.read()
    except FileNotFoundError:
        return "Error: File not found. Please check the filename or path."
    except Exception as e:
        return f"Error: {str(e)}"

# 3 Scenarios
print(read_file("sample.txt"))           # File exists
print(read_file("nonexistent.txt"))       # File missing
print(read_file("C:/wrong/path/file.txt")) # Invalid path
```
- Output:**

```
... Error: File not found. Please check the filename or path.
Error: File not found. Please check the filename or path.
Error: File not found. Please check the filename or path.
```
- Runtime Status:** RAM: 1.2 GB / 16 GB, Disk: 1.2 GB / 16 GB.

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.

The screenshot shows a Jupyter Notebook interface with a single cell containing Python code. The code defines a class `Car` with methods `start` and `drive`, and a variable `my_car` of type `Car`. It then prints the results of `start()` and `drive()`, and asserts their equality to the expected strings. The output shows the expected results for each print statement and the successful execution of all three assertions.

```
#4Bug: Calling an undefined method
class Car:
    def start(self):
        return "Car is started"

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

my_car = Car()
print(my_car.start())
print(my_car.drive())
print("Is my_car a Car?", type(my_car) == Car)
# 3 Assert Statements
assert my_car.start() == "Car is started"
assert my_car.drive() == "Car is driving"
assert type(my_car) == Car

...
Car is started
Car is driving
Is my_car a Car? True
```

Task Description #5

(`TypeError – Mixing Strings and Integers inAddition`)

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.

AI ass 7.1.ipynb ⚡ ☁

File Edit View Insert Runtime Tools Help

Commands + Code + Text ▶ Run all

RAM Disk

[37] #5Bug: TypeError due to mixing string and integer
#solution:
def add_five(value):
 return int(value) + 5
assert add_five("10") == 15
assert add_five(5) == 10
assert add_five("0") == 5
print(add_five("10"))
print(add_five(5))
print(add_five("0"))

[32] ... 15
10
5

[32] #Solution 2: String Concatenation
def add_five_str(value):
 return str(value) + "5"
assert add_five_str("10") == "105"
assert add_five_str(5) == "55"
assert add_five_str("0") == "05"
print(add_five_str("10"))
print(add_five_str(5))
print(add_five_str("0"))

[32] 105
55
05