

# **Assignment-7.1**

## **AI Assistant Coding**

**Name : P.Manikanta**

**Btach: 08**

**Rollno : 2303A51271**

### **Task Description #1**

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.

### Task Description #1 (Syntax Errors – Missing Parentheses in Print Statement)

#### AI Explanation

In Python 3, print is a function, so parentheses are required. The old style print "text" only worked in Python 2.

Double-click (or enter) to edit

#### Buggy Code(wrong,error one)

```
[8] 0s
def greet():
    print "Hello, AI Debugging Lab!"

greet()

▼   File "/tmp/ipython-input-2393093315.py", line 2
      print "Hello, AI Debugging Lab!"
      ^
SyntaxError: Missing parentheses in call to 'print'. Did you mean print(...)?
```

Next steps: [Explain error](#)

#### Error Observed

SyntaxError: Missing parentheses in call to 'print'

#### Corrected Code

```
[1] 0s
▶ def greet():
    return "Hello, AI Debugging Lab!"

# Calling function
print(greet())

# Tests
assert greet() == "Hello, AI Debugging Lab!"
assert isinstance(greet(), str)
assert greet().startswith("Hello")

▼   ... Hello, AI Debugging Lab!
```

#### Observation:

- The original program failed to run due to a **SyntaxError**.
- Python 3 requires parentheses for function calls, including `print()`.
- AI identified that the code followed **Python 2 syntax**.
- After correction, the program executed successfully.
- All assert test cases passed, confirming the function returned the expected string.

**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.

#### Task Description #2 (Incorrect condition in an If Statement)

##### AI Explanation

= is assignment. == is used for comparison inside conditions.

##### Buggy Code(wrong error one)

```
[7] ① 0s  def check_number(n):  
    if n = 10:  
        return "Ten"  
    else:  
        return "Not Ten"  
  
▼ ...   File "/tmp/ipython-input-857983334.py", line 2  
      if n = 10:  
      ^  
SyntaxError: invalid syntax. Maybe you meant '==' or ':=' instead of '='?
```

Next steps: [Explain error](#)

##### Error Observed

SyntaxError: cannot assign to expression

### Corrected Code

```
[4] ✓ Os
def check_number(n):
    if n == 10:
        return "Ten"
    else:
        return "Not Ten"

# Tests
assert check_number(10) == "Ten"
assert check_number(5) == "Not Ten"
assert check_number(0) == "Not Ten"
```

### Observation:

- The program produced a **syntax error** because = was used instead of ==.
- AI explained that = performs assignment, not comparison.
- Replacing = with == fixed the logical condition.
- The corrected function behaved correctly for different inputs.
- All assert test cases passed without errors.

### 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.

### Task Description #3 (Runtime Error – File Not Found)

#### AI Explanation

If the file does not exist, Python crashes. We must use try–except to handle this safely.

#### Buggy Code

```
[6] ① Os
def read_file(filename):
    with open(filename, 'r') as f:
        return f.read()

print(read_file("nonexistent.txt"))

-----
FileNotFoundError                         Traceback (most recent call last)
/tmp/ipython-input-2343812504.py in <cell line: 0>()
      3         return f.read()
      4
----> 5 print(read_file("nonexistent.txt"))

/tmp/ipython-input-2343812504.py in read_file(filename)
      1 def read_file(filename):
----> 2     with open(filename, 'r') as f:
      3         return f.read()
      4
      5 print(read_file("nonexistent.txt"))
```

#### Error Observed

FileNotFoundException: [Errno 2] No such file or directory

#### Corrected Code

```
[5]
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."

# ---- Tests ----

# 1) File missing
assert read_file("missing123.txt") == "Error: File not found."

# 2) Invalid path
assert "Error" in read_file("//badpath")

# 3) Existing file test
with open("sample.txt", "w") as f:
    f.write("Hello")

assert read_file("sample.txt") == "Hello"
```

#### Observation:

- The original code crashed when attempting to open a missing file.
- AI detected the unhandled FileNotFoundError.
- A try–except block was introduced to prevent program termination.
- User-friendly error messages were displayed instead of 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.

#### Task Description #4 (Calling a Non-Existent Method)

##### AI Explanation

drive() does not exist in the class. We can:

1. Add the method OR 2. Call the correct existing method.

##### Buggy code(error)

```
[9] ① os
    class Car:
        def start(self):
            return "Car started"

        my_car = Car()
        print(my_car.drive())

...
...
AttributeError                                     Traceback (most recent call last)
/tmp/ipython-input-268690456.py in <cell line: 0>()
      4
      5 my_car = Car()
----> 6 print(my_car.drive())

AttributeError: 'Car' object has no attribute 'drive'
```

AttributeError: 'Car' object has no attribute 'drive'

Next steps: [Explain error](#)

##### Error Observed

AttributeError: 'Car' object has no attribute 'drive'

##### Corrected Code

```
0] ①
    class Car:
        def start(self):
            return "Car started"

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

    my_car = Car()

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

##### Observation:

- The program raised an **AttributeError** because the drive() method was not defined.
- AI identified two possible fixes: define the method or change the call.
- Adding the drive() method solved the issue.
- The corrected class worked as expected.

- All assert tests passed, verifying class functionality.

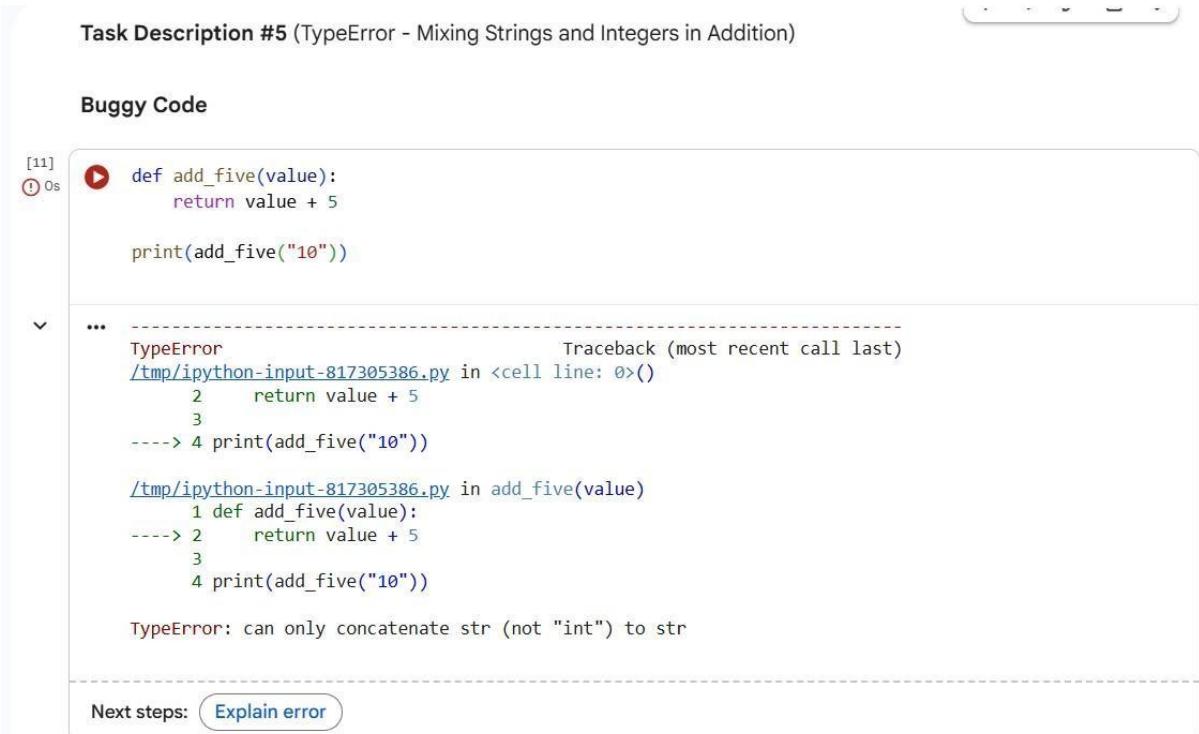
### Task Description #5 (TypeError – Mixing Strings and Integers in Addition)

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:



The screenshot shows a Jupyter Notebook cell with the following content:

**Task Description #5 (TypeError - Mixing Strings and Integers in Addition)**

**Buggy Code**

```
[11] ① Os
  def add_five(value):
      return value + 5

  print(add_five("10"))

...
----- Traceback (most recent call last)
/tmp/ipython-input-817305386.py in <cell line: 0>()
      2     return value + 5
      3
----> 4 print(add_five("10"))

/tmp/ipython-input-817305386.py in add_five(value)
      1 def add_five(value):
----> 2     return value + 5
      3
      4 print(add_five("10"))

TypeError: can only concatenate str (not "int") to str
```

Next steps: [Explain error](#)

### Error Observed

TypeError: can only concatenate str (not "int") to str

### AI Explanation

Python does not allow adding numbers and strings directly.

We can fix this in two ways:

#### Solution 1: Type Casting to Integer

```
[12] ✓ 0s
def add_five(value):
    return int(value) + 5

# Tests
assert add_five(10) == 15
assert add_five("5") == 10
assert add_five(0) == 5
```

#### Solution 2: String Concatenation

```
[13] ✓ 0s
def add_five_str(value):
    return str(value) + "5"
```

```
[12] ✓ 0s
def add_five(value):
    return int(value) + 5

# Tests
assert add_five(10) == 15
assert add_five("5") == 10
assert add_five(0) == 5
```

#### Solution 2: String Concatenation

```
[13] ✓ 0s
def add_five_str(value):
    return str(value) + "5"

# Tests
assert add_five_str("10") == "105"
assert add_five_str(7) == "75"
assert add_five_str(0) == "05"
```

### Observation:

The program crashed with a TypeError when adding a string and integer.

AI explained Python's strict type rules for addition.

Two valid solutions were proposed:

Convert input to an integer.

Convert everything to strings for concatenation.

