AI Assisted Coding

ASSIGNMENT 7.5

NAME: G.V.N HASINI

HT.NO: 2303A51099

Task 1

(Mutable Default Argument – Function Bug)

Task: Analyze given code where a mutable default argument causes unexpected behavior. Use AI to fix it.

Bug: Mutable default argument

def add_item(item, items=[]):

items.append(item)

return items

print(add_item(1))

print(add_item(2))

```
def add_item(item, items=None):
    if items is None:
        items = []
    items.append(item)
    return items

print(add_item(1))  # [1]
print(add_item(2))  # [2]'''
```

OUTPUT:

```
PS C:\Users\hasin\OneDrive\Desktop\java programming\ai assignments> & C:\Users\hasin\AppData\Local\Programs\Python\Python313\python.exe "c:/Users/hasin/OneDri
ve/Desktop/java programming/ai assignments/ass 7 ai"
[1]
[2]                                                 > & C:\Users\hasin\AppData\Local\Programs\Python\Python313\python.exe "c:/Users/hasin/OneDri
ve/Desktop/java programming/ai assignments/ass 7 ai"ai assignments> &
```

Task 2

(Floating-Point Precision Error)

Task: Analyze given code where floating-point comparison fails. Use AI to correct with tolerance.

# Bug: Floating point precision issue
def check_sum():
return (0.1 + 0.2) == 0.3
print(check_sum())

```
def check_sum():
    return math.isclose(0.1 + 0.2, 0.3, rel_tol=1e-9)

print(check_sum())  # True
```

OUTPUT:

```
PS C:\Users\hasin\OneDrive\Desktop\java programming\ai assignments> & C:\Users\hasin\AppData\Local\Programs\Python\Python313\python.exe "c:/Users/hasin/OneDri
ve/Desktop/java programming/ai assignments/ass 7 ai"
True
```

Task 3

(Recursion Error – Missing Base Case)

Task: Analyze given code where recursion runs infinitely due to missing base case. Use AI to fix.

# Bug: No base case
def countdown(n):
print(n)
return countdown(n-1)
countdown(5)

```
def countdown(n):
    if n <= 0:
        print("Done!")
        return
    print(n)
    countdown(n-1)

countdown(5)
```

OUTPUT:

```
PS C:\Users\hasin\OneDrive\Desktop\java programming\ai assignments> & C:\Users\hasin\AppData\Local\Programs\Python\Python313\python.exe "c:/Users/hasin/OneDrive/Desktop/java p
rogramming/ai assignments/ass 7 ai"
5
4
3
2
1
Done!
PS C:\Users\hasin\OneDrive\Desktop\java programming\ai assignments>
```

Task 4

(Dictionary Key Error)

Task: Analyze given code where a missing dictionary key causes error. Use AI to fix it.

# Bug: Accessing non-existing key
def get_value():
data = {"a": 1, "b": 2}
return data["c"]
print(get_value())

```python
def get_value():
    data = {"a": 1, "b": 2}
    try:
        return data["c"]
    except KeyError:
        return "missing"
```

OUTPUT:

```
PS C:\Users\hasin\OneDrive\Desktop\java programming\ai assignments> & C:\Users\hasin\AppData\Local\Programs\Python\Python313\python.exe "c:/Users/hasin/OneDrive/Desktop/java p
rogramming/ai assignments/ass 7 ai"
missing
PS C:\Users\hasin\OneDrive\Desktop\java programming\ai assignments>
```

Task 5

(Infinite Loop – Wrong Condition)

Task: Analyze given code where loop never ends. Use AI to detect and fix it.

# Bug: Infinite loop
def loop_example():
i = 0
while i < 5:

print(i)

```
def loop_example():
    i = 0
    while i < 5:
        print(i)
        i += 1

loop_example()  # prints 0..4
```

OUTPUT:

```
PS C:\Users\hasin\OneDrive\Desktop\java programming\ai assignments> & C:\Users\hasin\AppData\Local\Programs\Python\Python313\python.exe "c:/Users/hasin/OneDrive/Deskt
rogramming/ai assignments/ass 7 ai"
0
1
2
3
4
PS C:\Users\hasin\OneDrive\Desktop\java programming\ai assignments>
```

Task 6

(Unpacking Error – Wrong Variables)
Task: Analyze given code where tuple unpacking fails. Use AI to fix it.
Bug: Wrong unpacking
a, b = (1, 2, 3)

```
# match exactly
a, b, c = (1, 2, 3)
print(a, b)  # 1 2

# ignore extra
a, b, _ = (1, 2, 3)
print(a, b)  # 1 2

# or capture extras
a, b, *rest = (1, 2, 3)
print(a, b, rest)  # 1 2 [3]
```

OUTPUT:

```
PS C:\Users\hasin\OneDrive\Desktop\java programming\ai assignments> & C:\Users\hasin\AppData\Local\Programs\Python\Python313\python.exe "c:/Users/hasin/OneDrive/Desktop/java p
rogramming/ai assignments/ass 7 ai"
1 2
1 2
1 2 [3]
PS C:\Users\hasin\OneDrive\Desktop\java programming\ai assignments>
```

Task 7

(Mixed Indentation – Tabs vs Spaces)

Task: Analyze given code where mixed indentation breaks execution. Use AI to fix it.

# Bug: Mixed indentation
def func():
x = 5
y = 10
return x+y

```
def func():
    x = 5
    y = 10
    return x + y

print(func())  # 15
```

OUTPUT:

```
PS C:\Users\hasin\OneDrive\Desktop\java programming\ai assignments> & C:\Users\hasin\AppData\Local\Programs\Python\Python313\python.exe "c:/Users/hasin/OneDrive/Desktop/java p
rogramming/ai assignments/ass 7 ai"
15
```

Task 8

(Import Error – Wrong Module Usage)

Task: Analyze given code with incorrect import. Use AI to fix.

# Bug: Wrong import
import maths
print(maths.sqrt(16))

```
import math
print(math.sqrt(16))
```

OUTPUT:

```
PS C:\Users\hasin\OneDrive\Desktop\java programming\ai assignments> & C:\Users\hasin\AppData\Local\Programs\Python\Python313\python.exe "c:/Users/hasin/OneDri
ve/Desktop/java programming/ai assignments/ass 7 ai"
4.0
PS C:\Users\hasin\OneDrive\Desktop\java programming\ai assignments>
```

Task 9

(Unreachable Code – Return Inside Loop)

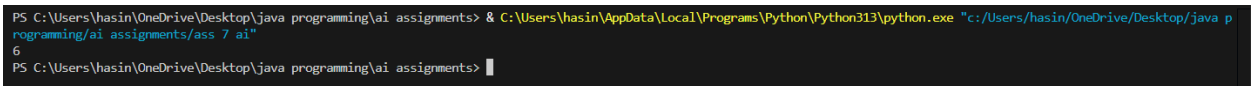
Task: Analyze given code where a return inside a loop prevents full iteration. Use AI to fix it.

# Bug: Early return inside loop
def total(numbers):
for n in numbers:
return n
print(total([1,2,3]))

```
# fixed with explicit loop
def total(numbers):
    total = 0
    for n in numbers:
        total += n
    return total

print(total([1, 2, 3]))  # 6
```

OUTPUT:



Task 10

(Name Error – Undefined Variable)

Task: Analyze given code where a variable is used before being defined. Let AI detect and fix the error.

# Bug: Using undefined variable
def calculate_area():
return length * width
print(calculate_area())

Requirements:

- Run the code to observe the error.
- Ask AI to identify the missing variable definition.
- Fix the bug by defining length and width as parameters.

• Add 3 assert test cases for correctness.

Expected Output :

• Corrected code with parameters.

• AI explanation of the bug.

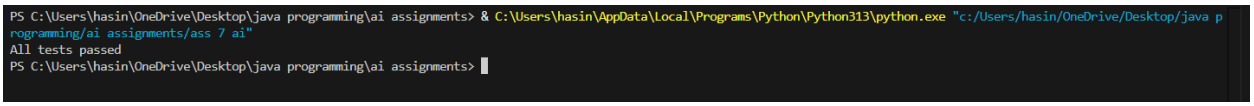
Successful execution of assertions.

```python
def calculate_area(length, width):
    return length * width

# tests
assert calculate_area(2, 3) == 6
assert calculate_area(0, 5) == 0
assert calculate_area(3.5, 2) == 7.0

print("All tests passed")
```

OUTPUT:



Bug Explanation:

• The function tried to use `length` and `width` without defining them → causes **NameError**.

• Fix: add `length` and `width` as **function parameters**, so values are passed during the function call.

• Assertions verify correctness automatically.

Task 11

(Type Error – Mixing Data Types Incorrectly)

Task: Analyze given code where integers and strings are added

incorrectly. Let AI detect and fix the error.

```
# Bug: Adding integer and string
def add_values():
return 5 + "10"
print(add_values())
```

Requirements:

• Run the code to observe the error.

• AI should explain why int + str is invalid.
• Fix the code by type conversion (e.g., int("10") or str(5)).
• Verify with 3 assert cases.
Expected Output #6:
• Corrected code with type handling.
• AI explanation of the fix.

```python
def add_values():
    return 5 + int("10")

# tests
assert add_values() == 15
assert add_values() == 5 + 10
assert isinstance(add_values(), int)

print("All tests passed")
```

OUTPUT:

```
PS C:\Users\hasin\OneDrive\Desktop\java programming\ai assignments> & C:\Users\hasin\AppData\Local\Programs\Python\Python313\python.exe "c:/Users/hasin/OneDrive/Desktop/java p
rogramming/ai assignments/ass 7 ai"
All tests passed
PS C:\Users\hasin\OneDrive\Desktop\java programming\ai assignments>
```

AI explanation of the fix:

Python disallows addition between int and str (TypeError). Converting "10" to int with int("10") makes both operands integers so addition works.

Task 12

(Type Error – String + List Concatenation)
Task: Analyze code where a string is incorrectly added to a list.
# Bug: Adding string and list
def combine():
return "Numbers: " + [1, 2, 3]
print(combine())
Requirements:
• Run the code to observe the error.

- Explain why str + list is invalid.
- Fix using conversion (str([1,2,3]) or " ".join()).
- Verify with 3 assert cases.

Expected Output:

- Corrected code
- Explanation
- Successful test validation

```python
def combine():
    return "Numbers: " + " ".join(map(str, [1, 2, 3]))

# tests
assert combine() == "Numbers: 1 2 3"
assert isinstance(combine(), str)
assert combine().startswith("Numbers:")

print("All tests passed")
```

OUTPUT:

```
PS C:\Users\hasin\OneDrive\Desktop\java programming\ai assignments> & C:\Users\hasin\AppData\Local\Programs\Python\Python313\python.exe "c:/Users/hasin/OneDrive/Desktop/java p
rogramming/ai assignments/ass 7 ai"
All tests passed
PS C:\Users\hasin\OneDrive\Desktop\java programming\ai assignments>
```

Ai explanation:

Python cannot add different types directly; converting the list elements to strings and joining them produces a single string that can be concatenated with the prefix.

Task 13

(Type Error – Multiplying String by Float)

Task: Detect and fix code where a string is multiplied by a float.

Bug: Multiplying string by float

def repeat_text():

return "Hello" * 2.5

print(repeat_text())

Requirements:

- Observe the error.
- Explain why float multiplication is invalid for strings.
- Fix by converting float to int.
- Add 3 assert test cases.

```python
def repeat_text(times):
    return "Hello" * int(times)

# tests
assert repeat_text(2.5) == "HelloHello"
assert repeat_text(3.0) == "HelloHelloHello"
assert repeat_text(0.9) == ""

print("All tests passed")
```

OUTPUT:

```
PS C:\Users\hasin\OneDrive\Desktop\java programming\ai assignments> & C:\Users\hasin\AppData\Local\Programs\Python\Python313\python.exe "c:/Users/hasin/OneDrive/Desktop/java p
rogramming/ai assignments/ass 7 ai"
All tests passed
PS C:\Users\hasin\OneDrive\Desktop\java programming\ai assignments>
```

The Error:

Python raises TypeError: can't multiply sequence by non-int of type 'float'. Strings can only be repeated by integers, so a float (2.5) is invalid.

Explanation:

Multiplying a string by a non-integer is not allowed because repetition count must be an integer; using int() converts the float to an integer so repetition works.

Task 14

(Type Error – Adding None to Integer)

Task: Analyze code where None is added to an integer.

# Bug: Adding None and integer
def compute():
value = None
return value + 10
print(compute())

Requirements:

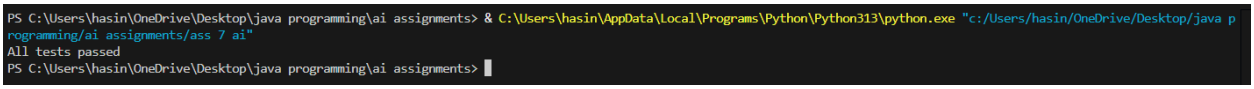
- Run and identify the error.
- Explain why NoneType cannot be added.
- Fix by assigning a default value.
- Validate using asserts.

```python
def compute(value=None):
    if value is None:
        value = 0
    return value + 10

# tests
assert compute() == 10
assert compute(5) == 15
assert compute(-10) == 0

print("All tests passed")
```

OUTPUT:



Finding:

Running the original snippet shows TypeError: unsupported operand type(s) for +: 'NoneType' and 'int' because None cannot be added to a number. I fixed it by assigning a default numeric value (0) when value is None and added assertions to validate behavior.

Explanation:

NoneType has no numeric addition behavior, so None + 10 raises a TypeError. Providing a sensible default (e.g., 0) ensures the function always performs a valid numeric addition.

Task 15

(Type Error – Input Treated as String Instead of Number)

Task: Fix code where user input is not converted properly.

```
# Bug: Input remains string
def sum_two_numbers():
a = input("Enter first number: ")
b = input("Enter second number: ")
return a + b
print(sum_two_numbers())
```

Requirements:

- Explain why input is always string.
- Fix using int() conversion.
- Verify with assert test cases.

```python
def sum_two_numbers(a=None, b=None):
    if a is None:
        a = input("Enter first number: ")
    if b is None:
        b = input("Enter second number: ")
    return int(a) + int(b)

# tests
assert sum_two_numbers('2', '3') == 5
assert sum_two_numbers(0, '5') == 5
assert sum_two_numbers('-1', 1) == 0

print("All tests passed")
```

OUTPUT:

```
PS C:\Users\hasin\OneDrive\Desktop\java programming\ai assignments> & C:\Users\hasin\AppData\Local\Programs\Python\Python313\python.exe "c:/Users/hasin/OneDrive/Desktop/java p
rogramming/ai assignments/ass 7 ai"
All tests passed
PS C:\Users\hasin\OneDrive\Desktop\java programming\ai assignments>
```

Problem:

input() always returns a string, so a + b concatenates (e.g., "2" + "3" == "23"). Convert to int() to get numeric addition.