

AI Assisted Coding

Assignment 7.5

Name: **D.Akash**

Ht.no: **2303A51422**

Batch: **21**

Lab 7: Error Debugging with AI: Systematic approaches to finding and fixing bugs

Lab Objectives:

- To identify and correct syntax, logic, and runtime errors in Python programs using AI tools.
- To understand common programming bugs and AI-assisted debugging suggestions.
- To evaluate how AI explains, detects, and fixes different types of coding errors.
- To build confidence in using AI to perform structured debugging practices.

Lab Outcomes (LOs):

After completing this lab, students will be able to:

- Use AI tools to detect and correct syntax, logic, and runtime errors.
- Interpret AI-suggested bug fixes and explanations.
- Apply systematic debugging strategies supported by AI-generated insights.

Task 1 (Mutable Default Argument – Function Bug)

Task: Analyze given code where a mutable default argument cause

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))
```

Expected Output: Corrected function avoids shared list bug.

Code:

```
lab 06.py X
lab 06.py > ...
1  '''Task: Analyze given code where a mutable default argument causes
2  unexpected behavior. Use AI to fix it.
3  # Bug: Mutable default argument
4  def add_item(item, items=[]):
5      items.append(item)
6      return items
7  print(add_item(1))
8  print(add_item(2))
9  Expected Output: Corrected function avoids shared list bug.'''
10
11 # Bug: Mutable default argument
12 def add_item(item, items=[]):
13     items.append(item)
14     return items
15 # Fix: Use None as the default value and create a new list inside the function
16 def add_item(item, items=None):
17     if items is None:
18         items = []
19     items.append(item)
20     return items
21 # Testing the corrected function
22 print(add_item(1))
23 print(add_item(2))
```

Output:

```
PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS SQL HISTORY TASK MONITOR
[Done] exited with code=0 in 0.172 seconds

[Running] python -u "c:\Users\akash\OneDrive\Desktop\betech_3_2\AI_Assisted_Coding\lab 06.py"
[1]
[2]

[Done] exited with code=0 in 0.236 seconds
```

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())
```

Expected Output: Corrected function

Code:

```
temp.py > ...
1  '''Task: Analyze given code where floating-point comparison fails.
2  Use AI to correct with tolerance.
3  # Bug: Floating point precision issue
4  def check_sum():
5  return (0.1 + 0.2) == 0.3
6  print(check_sum())
7  Expected Output: Corrected function'''
8
9  # Bug: Floating point precision issue
0  def check_sum():
1  |   return (0.1 + 0.2) == 0.3
2  # Fix: Use a tolerance for comparison
3  def check_sum():
4  |   return abs((0.1 + 0.2) - 0.3) < 1e-9
5  # Testing the corrected function
6  print(check_sum())
7
```

Filter (e.g. text, !excludeText, t... Code ▾ ☰

Output:

```
PROBLEMS  OUTPUT  DEBUG CONSOLE  TERMINAL  PORTS  SQL HISTORY  TASK MONITOR

[Done] exited with code=0 in 0.236 seconds

[Running] python -u "c:\Users\akash\OneDrive\Desktop\betech_3_2\AI_Assisted_Coding\temp.py"
True

[Done] exited with code=0 in 0.166 seconds
```

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)
```

Expected Output : Correct recursion with stopping condition.

Code:

```

temp.py > ...
1  '''Task 3 (Recursion Error ☐ Missing Base Case)
2  Task: Analyze given code where recursion runs infinitely due to
3  missing base case. Use AI to fix.
4  # Bug: No base case
5  def countdown(n):
6  print(n)
7  return countdown(n-1)
8  countdown(5)
9  Expected Output : Correct recursion with stopping condition.'''
10
11 # Bug: No base case
12 def countdown(n):
13     print(n)
14     return countdown(n-1)
15 # Fix: Add a base case to stop recursion
16 def countdown(n):
17     if n <= 0:
18         print("Blast off!")
19         return
20     print(n)
21     return countdown(n-1)
22 # Testing the corrected function
23 countdown(5)
24

```

Output:

```

PROBLEMS  OUTPUT  DEBUG CONSOLE  TERMINAL  PORTS  SQL HISTORY  TASK MONITOR
PS C:\Users\akash\OneDrive\Desktop\betech_3_2\AI_Assisted_Coding> & C:\Users\akash\AppData\Local\Programs\Python\Python311\python.exe C:\Users\akash\OneDrive\Desktop\betech_3_2\AI_Assisted_Coding/temp.py
5
4
3
2
1
Blast off!
PS C:\Users\akash\OneDrive\Desktop\betech_3_2\AI_Assisted_Coding>

```

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())
```

Expected Output: Corrected with .get() or error handling.

Code:

```
temp.py > ...  
1  '''Task 4 (Dictionary Key Error)  
2  Task: Analyze given code where a missing dictionary key causes  
3  error. Use AI to fix it.  
4  # Bug: Accessing non-existing key  
5  def get_value():  
6  data = {"a": 1, "b": 2}  
7  return data["c"]  
8  print(get_value())  
9  Expected Output: Corrected with .get() or error handling.'''  
10  
11 # Bug: Accessing non-existing key  
12 def get_value():  
13     data = {"a": 1, "b": 2}  
14     return data["c"]  
15 # Fix: Use .get() method to avoid KeyError  
16 def get_value():  
17     data = {"a": 1, "b": 2}  
18     return data.get("c", "Key not found")  
19 # Testing the corrected function  
20 print(get_value())  
21
```

Output:

```
PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS SQL HISTORY TASK MONITOR
● PS C:\Users\akash\OneDrive\Desktop\betech_3_2\AI_Assisted_Coding> & C:\Users\akash\AppData\Local\Programs\Python\Python313\python
tech_3_2\AI_Assisted_Coding/temp.py
Key not found
○ PS C:\Users\akash\OneDrive\Desktop\betech_3_2\AI_Assisted_Coding> 
```

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)
```

Expected Output: Corrected loop increments i.

Code:

```

temp.py > loop_example
1  '''Task 5 (Infinite Loop ❌ Wrong Condition)
2  Task: Analyze given code where loop never ends. Use AI to detect
3  and fix it.
4  # Bug: Infinite loop
5  def loop_example():
6  i = 0
7  while i < 5:
8  print(i)
9  Expected Output: Corrected loop increments i.'''
10
11 # Bug: Infinite loop
12 def loop_example():
13     i = 0
14     while i < 5:
15         print(i)
16 # Fix: Increment i inside the loop
17 def loop_example():
18     i = 0
19     while i < 5:
20         print(i)
21         i += 1
22 # Testing the corrected function
23 loop_example()
24
25

```

Output:

```

PROBLEMS  OUTPUT  DEBUG CONSOLE  TERMINAL  PORTS  SQL HISTORY  TASK MONITOR
[Running] python -u "c:\Users\akash\OneDrive\Desktop\betech_3_2\AI_Assisted_Coding\temp.py"
0
1
2
3
4
[Done] exited with code=0 in 0.195 seconds

```

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)

Expected Output: Correct unpacking or using _ for extra values.

Code:

```
temp.py > ...
1  '''Task 6 (Unpacking Error [ ] Wrong Variables)
2  Task: Analyze given code where tuple unpacking fails. Use AI to
3  fix it.
4  # Bug: Wrong unpacking
5  a, b = (1, 2, 3)
6  Expected Output: Correct unpacking or using _ for extra values.'''
7
8
9  # fix error by correcting the number of variables
10 a, b, c = (1, 2, 3) # Correcting the number of variables
11 print(a, b, c)
12
13
14
15
```

Output:

```
PROBLEMS  OUTPUT  DEBUG CONSOLE  TERMINAL  PORTS  SQL HISTORY  TASK MONITOR
PS C:\Users\akash\OneDrive\Desktop\betech_3_2\AI_Assisted_Coding> & C:\Users\akash\AppData\Local\Programs\Python\Python312\AI_Assisted_Coding/temp.py
1 2 3
PS C:\Users\akash\OneDrive\Desktop\betech_3_2\AI_Assisted_Coding> |
```

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
```

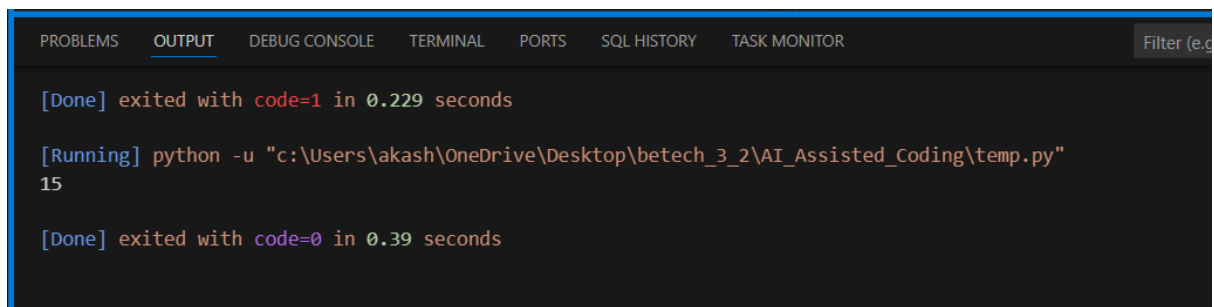
```
    return x+y
```

Expected Output : Consistent indentation applied.

Code:

```
temp.py > ...
1  '''Task 7 (Mixed Indentation  Tabs vs Spaces)
2  Task: Analyze given code where mixed indentation breaks
3  execution. Use AI to fix it.
4  # Bug: Mixed indentation
5  def func():
6      x = 5
7      y = 10
8      return x+y
9  Expected Output : Consistent indentation applied.'''
10
11  # Bug: Mixed indentation
12  def func():
13      x = 5
14      y = 10
15      return x + y
16  # Fix: Ensure consistent indentation (using spaces)
17  def func():
18      x = 5
19      y = 10
20      return x + y
21  # Testing the corrected function
22  print(func())
23  |
```

Output:

A screenshot of an IDE's Output window. The window has a dark background with a light blue border. At the top, there is a tab bar with several tabs: PROBLEMS, OUTPUT (which is selected and highlighted in blue), DEBUG CONSOLE, TERMINAL, PORTS, SQL HISTORY, and TASK MONITOR. To the right of the tabs is a search filter input field with the placeholder text "Filter (e.g...". The main area of the window displays three lines of execution logs in a monospaced font. The first line is "[Done] exited with code=1 in 0.229 seconds". The second line is "[Running] python -u \"c:\\Users\\akash\\OneDrive\\Desktop\\betech_3_2\\AI_Assisted_Coding\\temp.py\" 15". The third line is "[Done] exited with code=0 in 0.39 seconds".

```
PROBLEMS  OUTPUT  DEBUG CONSOLE  TERMINAL  PORTS  SQL HISTORY  TASK MONITOR  Filter (e.g...

[Done] exited with code=1 in 0.229 seconds

[Running] python -u "c:\\Users\\akash\\OneDrive\\Desktop\\betech_3_2\\AI_Assisted_Coding\\temp.py"
15

[Done] exited with code=0 in 0.39 seconds
```

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))
```

Expected Output: Corrected to import math

Code:

```
temp.py
1  '''Task 8 (Import Error ❌ Wrong Module Usage)
2  Task: Analyze given code with incorrect import. Use AI to fix.
3  # Bug: Wrong import
4  import maths
5  print(maths.sqrt(16))
6  Expected Output: Corrected to import math'''
7
8  # Bug: Wrong import
9  import math
10 print(math.sqrt(16))
11 # Fix: Correct the import statement to import math
12 import math
13 print(math.sqrt(16))
14
```

Output:

```
PROBLEMS  OUTPUT  DEBUG CONSOLE  TERMINAL  PORTS  SQL HISTORY  TASK MONITOR
[Done] exited with code=0 in 0.39 seconds

[Running] python -u "c:\Users\akash\OneDrive\Desktop\betech_3_2\AI_Assisted_Coding\temp.py"
4.0
4.0

[Done] exited with code=0 in 0.326 seconds
```