

# AI ASSISTED CODING

SHASHANK YELAGAM

2303A510i3

BATCH – 03

06 – 02 – 2026

---

## ASSIGNMENT – 7.5

Lab – 07: Error Debugging with AI : Systematic Approaches to finding and fixing bugs.

Task – 01: Mutable Default Argument – Function Bug.

The screenshot shows a code editor interface with the following sections:

- Commands:** Search, Code, Text, Run all.
- RAM/Disk:** RAM 1.5GB, Disk 100GB.
- TASK 01**
- ERROR CODE:**

```
[18] # Bug: Mutable default argument
def add_item(item, items=[]):
    items.append(item)
    return items
print(add_item(1))
print(add_item(2))
```

Output: [1]  
[1, 2]
- FIXED CODE:**

```
[9] # Bug: Mutable default argument
def add_item(item, items=None):
    if items is None:
        items = []
    items.append(item)
    return items
print(add_item(1))
print(add_item(2))
```

Output: ... [1]  
[2]
- Variables, Terminal:** 11:42 AM, Python 3.

Explanation : The above error occurs because the above list items are created only once and it is Reused.

Task – 02 : Floating – Point Precision Error.

**TASK 02**

**ERROR CODE**

```
[12] ❶ # Bug: Floating point precision issue
❷ def check_sum():
❸     return (0.1 + 0.2) == 0.3
❹ print(check_sum())
❺
❻ False
```

**FIXED CODE**

```
[13] ❶ def check_sum():
❷     return abs((0.1 + 0.2) - 0.3) < 1e-9
❸
❹ print(check_sum())
❺
❻ ... True
```

Variables Terminal 11:50 AM Python 3

**Explanation :** The above error occurs because Float – Point Numbers cannot be Compared Directly. So, it is Fixed Using Tolerance.

### Task – 03 : Recursion Error – Missing base Case.

File Edit View Insert Runtime Tools Help

Commands + Code + Text Run all RAM Disk

**TASK 03**

**ERROR CODE**

```
[17] ❶ def countdown(n):
❷     print(n)
❸     return countdown(n-1)
❹
❺ countdown(5)
❻
⠁ ... 5
⠁ 4
⠁ 3
⠁ 2
⠁ 1
⠁ 0
⠁ -1
⠁ -2
⠁ -3
⠁ -4
⠁ -5
⠁ -6
```

Variables Terminal 11:59 AM Python 3

The screenshot shows a Python code editor interface. At the top, there's a menu bar with File, Edit, View, Insert, Runtime, Tools, Help. Below the menu is a toolbar with Commands, Code, Text, Run all, and a RAM/Disk indicator. The main area has a sidebar with tabs for Code, Text, and Run all. The main pane displays the following code:

```
def countdown(n):
    if n <= 0: # Base case
        return
    print(n)
    countdown(n-1)
countdown(5)
```

When run, it prints:

```
5
4
3
2
1
```

An error message is shown: "RecursionError: maximum recursion depth exceeded". Below the error message is a button labeled "Explain error". Under the error message, there's a section titled "FIXED CODE" containing the following code:

```
[16]
def countdown(n):
    if n <= 0: # Base case
        return
    print(n)
    countdown(n-1)
countdown(5)
```

The status bar at the bottom shows Variables, Terminal, 11:59 AM, Python 3.

**Explanation :** The above error occurs because in the error code there is no stopping condition it has infinite loop. So, fixed it using loop condition.

## Task – 04 : Dictionary key Errors.

The screenshot shows a Python code editor interface. At the top, there's a menu bar with File, Edit, View, Insert, Runtime, Tools, Help. Below the menu is a toolbar with Commands, Code, Text, Run all, and a RAM/Disk indicator. The main area has a sidebar with tabs for Code, Text, and Run all. The main pane displays the following code:

```
[22]
def get_value():
    data = {"a": 1, "b": 2}
    return data["c"]

print(get_value())
```

An error message is shown: "KeyError: 'c'". Below the error message is a detailed traceback:

```
-----  
KeyError                         Traceback (most recent call last)  
/tmp/ipython-input-1845996374.py in <cell line: 0>()  
      3     return data["c"]  
      4  
----> 5 print(get_value())  
  
/tmp/ipython-input-1845996374.py in get_value()
      1 def get_value():
      2     data = {"a": 1, "b": 2}
----> 3     return data["c"]
      4
```

The status bar at the bottom shows Variables, Terminal, 12:04 PM, Python 3.

The screenshot shows a Python code editor interface. At the top, there are tabs for Commands, Code, Text, Run all, and a RAM/Disk indicator. The main area displays the following code:

```
    3     return data["c"]
    4
    5 print(get_value())
KeyError: 'c'
```

Below the code, a message says "Next steps: Explain error".

**FIXED CODE**

```
[23] 0s
  def get_value():
      data = {"a": 1, "b": 2}
      return data.get("c", "Key not found")

print(get_value())
... Key not found
```

At the bottom, there are buttons for Variables and Terminal, and a timestamp of 12:04 PM.

**Explanation:** The above Error occurs in the above code is key Error because it has accessing the key which is not existed. So, Fixed it using returning None value or Key Not Found Method.

### Task – 05: Infinite Loop – Wrong Condition.

The screenshot shows a Python code editor interface. At the top, there are tabs for Commands, Code, Text, Run all, and a RAM/Disk indicator. The main area displays the following code:

```
[24] 49%
  def loop_example():
      i = 0
      while i < 5:
          print(i)
      loop_example()
... Show hidden output
```

A red circle with a question mark icon is next to the first line of code, indicating an error.

**FIXED CODE**

```
[25]
  def loop_example():
      i = 0
      while i < 5:
          print(i)
          i += 1 # Increment added
      loop_example()

0
1
2
3
4
```

At the bottom, there are buttons for Variables and Terminal, and a timestamp of 12:10PM.

**Explanation:** The above Error occurs in the above code because in the above loop the “I” is never incremented in the loop variable. So, the code is fixed by incrementing the “I” in the loop.

### Task – 06: Unpacking Error – Wrong variables.

```
a, b = (1, 2, 3)

ValueError: too many values to unpack (expected 2)
```

```
a, b, _ = (1, 2, 3)
print(a, b)
```

**Explanation:** The above Error occurs in the above code because it has too many values to unpack. So, fixed the above code using packing method.

### Task – 07: Mixed Indentation – Tabs vs Spaces.

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

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

**Explanation:** The above error occurs in the above code because, python does not allows the mixing tabs and spaces. So, it is fixed by consisting only 4 spaces systematically.

### Task – 08: Import Error – Wrong Module Usage.

The screenshot shows a Jupyter Notebook interface with two code cells and their outputs.

**Cell 1 (Error):**

```
[30] Gemini
import maths
print(maths.sqrt(16))
+import math
+print(math.sqrt(16))

...
ModuleNotFoundError: Traceback (most recent call last)
/tmp/ipython-input-1512532258.py in <cell line: 0>()
----> 1 import maths
      2 print(maths.sqrt(16))

ModuleNotFoundError: No module named 'maths'

NOTE: If your import is failing due to a missing package, you can
manually install dependencies using either !pip or !apt.

To view examples of installing some common dependencies, click the
"Open Examples" button below.
```

**Cell 2 (Fixed):**

```
[31] Gemini
import math
print(math.sqrt(16))

4.0
```

**UI Elements:**

- Toolbar: Commands, + Code, + Text, Run all, RAM Disk
- Sidebar: Task list, Error code, Gemini status, Variables, Terminal
- Bottom: 12:22PM, Python 3

**Explanation:** The above error occurs in the above code because the module name is Incorrect. So, fixed it by renaming the module name correctly.