

# AI ASSISTED CODING

SUMANTH POLAM

2303A51121

BATCH – 03

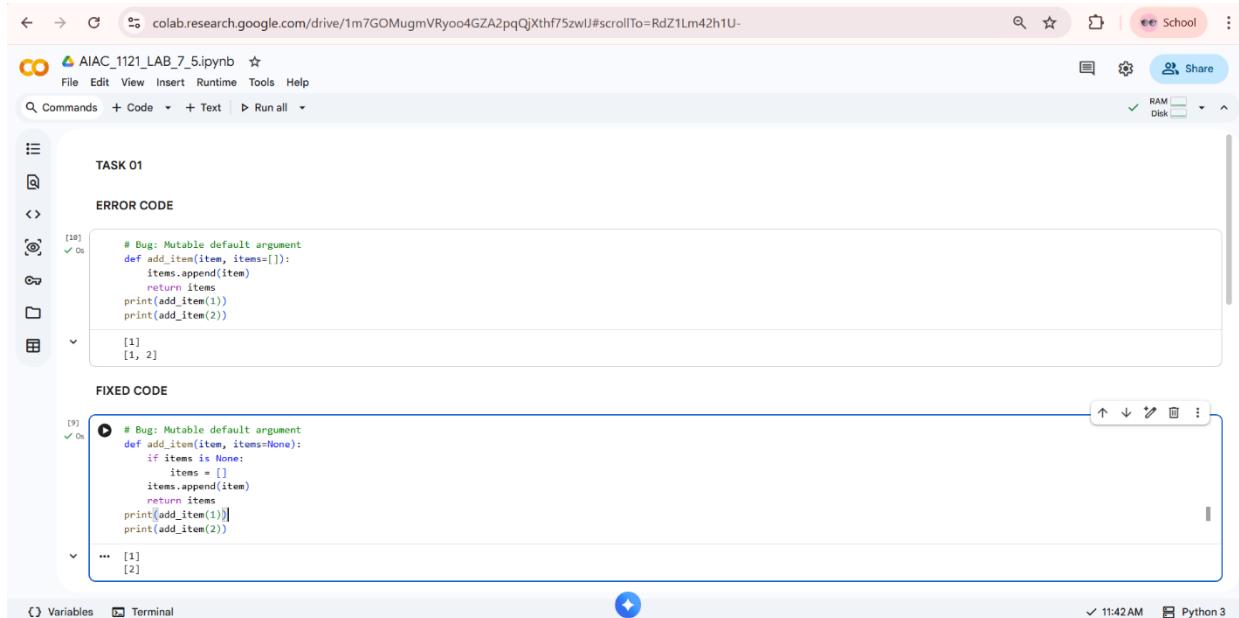
06 – 02 – 2026

---

## 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 Google Colab notebook titled "AIAC\_1121\_LAB\_7\_5.ipynb". The code cell [18] contains the following Python code:

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

The output of this code is:

```
[1]
[1, 2]
```

Below the code cell is a "FIXED CODE" section containing the corrected code:

```
# 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))
```

The output of the fixed code is:

```
[1]
[1, 2]
```

The Colab interface includes standard navigation and sharing tools at the top, and a sidebar on the left. The bottom status bar shows "11:42 AM" and "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.

The screenshot shows a Google Colab notebook titled "AIAC\_1121\_LAB\_7\_5.ipynb". The code cell [12] contains:

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

The output is "False".

The code cell [13] contains:

```
def check_sum():
    return abs((0.1 + 0.2) - 0.3) < 1e-9
print(check_sum())
```

The output is "... True".

At the bottom right, it says "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.

The screenshot shows a Google Colab notebook titled "AIAC\_1121\_LAB\_7\_5.ipynb". The code cell [17] contains:

```
def countdown(n):
    print(n)
    return countdown(n-1)

countdown(5)
```

The output shows a stack overflow error with the following trace:

```
... 5
4
3
2
1
0
-1
-2
-3
-4
-5
-6
```

At the bottom right, it says "11:59 AM Python 3".

The screenshot shows a Google Colab notebook titled "AIAC\_1121\_LAB\_7\_5.ipynb". In the code editor, there is a cell containing the following Python code:

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

The output of this code is:

```
5
4
3
2
1
```

An error message is displayed below the code:

```
RecursionError: maximum recursion depth exceeded
```

Next steps: [Explain error](#)

Below the code editor, under the heading "FIXED CODE", is a corrected version of the code:

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

The output of the fixed code is:

```
5
4
3
2
1
```

At the bottom of the interface, there are buttons for "Variables" and "Terminal", and a status bar showing "11:59 AM" and "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 Google Colab notebook titled "AIAC\_1121\_LAB\_7\_5.ipynb". In the code editor, there is a cell containing the following Python code:

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

print(get_value())
```

An error message is displayed below the code:

```
KeyError Traceback (most recent call last)
/tmp/ipython-input-1845996374.py in <cell line: 3>()
      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
```

At the bottom of the interface, there are buttons for "Variables" and "Terminal", and a status bar showing "12:04 PM" and "Python 3".

The screenshot shows a Google Colab notebook titled "AIAC\_1121\_LAB\_7\_5.ipynb". In the code editor, there is an error message:

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

Below the error, a "FIXED CODE" section shows the corrected code:

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

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

At the bottom of the screen, there are tabs for "Variables" and "Terminal". The status bar indicates the time is 12:04 PM and the Python version is Python 3.

**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 Google Colab notebook titled "AIAC\_1121\_LAB\_7\_5.ipynb". In the code editor, there is an error message indicating an infinite loop:

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

Below the error, a "FIXED CODE" section shows the corrected 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 of the screen, there are tabs for "Variables" and "Terminal". The status bar indicates the time is 12:10 PM and the Python version is Python 3.

**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.

The screenshot shows a Google Colab notebook titled "AIAC\_1121\_LAB\_7\_5.ipynb". In cell [26], the code `a, b = (1, 2, 3)` is run, resulting in a `ValueError` because there are too many values to unpack (expected 2). The error message is displayed in the output. Below the error, a link to "Explain error" is shown. In the "FIXED CODE" section, the code is modified to `a, b, \_ = (1, 2, 3)` to handle the extra value. The status bar at the bottom right shows "12:15 PM" and "Python 3".

**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.

The screenshot shows a Google Colab notebook titled "AIAC\_1121\_LAB\_7\_5.ipynb". In cell [28], the code contains mixed indentation: `def func():` uses a tab, while `x = 5` and `y = 10` use spaces. This results in an `IndentationError: unexpected indent`. The error message is shown in the output. Below the error, a link to "Explain error" is present. In the "FIXED CODE" section, the code is corrected to use统一的缩进 (spaces) throughout. The status bar at the bottom right shows "12:19 PM" and "Python 3".

**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 two screenshots of Google Colab notebooks, one above the other, demonstrating the process of fixing an import error.

**Screenshot 1 (Top):** This screenshot shows an error in the code cell:

```
[30]:  
❶ import maths  
❷ print(maths.sqrt(16))  
❸ +import math  
❹ +print(math.sqrt(16))
```

The error message is:

```
ModuleNotFoundError: No module named 'maths'
```

Below the error message, there is a note:

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.

**Screenshot 2 (Bottom):** This screenshot shows the code after it has been fixed:

```
[31]:  
❶ import math  
❷ print(math.sqrt(16))
```

The output of the code is:

```
4.0
```

At the bottom of the code cell, there is a "Next steps:" section with a "Fix" button.

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

**THANK YOU!!**