# AI ASSISTED CODING

**SAI THRISHOOL** **2303A51127**

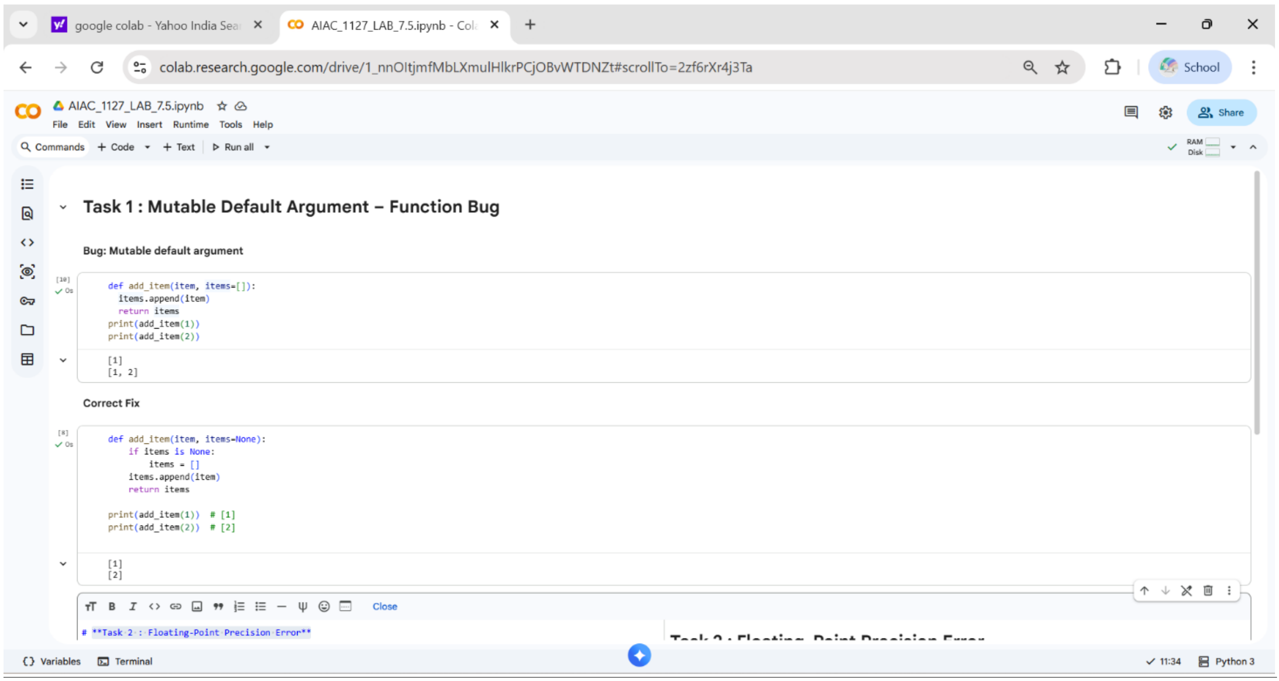
**BATCH – 03** **06 – 02 – 2026**

## ASSIGNMENT – 7.5

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

## TASK - 01 : Mutable Default Argument – Function Bug

### ERROR AND FIXED CODE:



Task 1 : Mutable Default Argument – Function Bug

Bug: Mutable default argument

```
def add_item(item, items=[]):
  items.append(item)
  return items
print(add_item(1))
print(add_item(2))
```

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

Correct Fix

```
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]
```

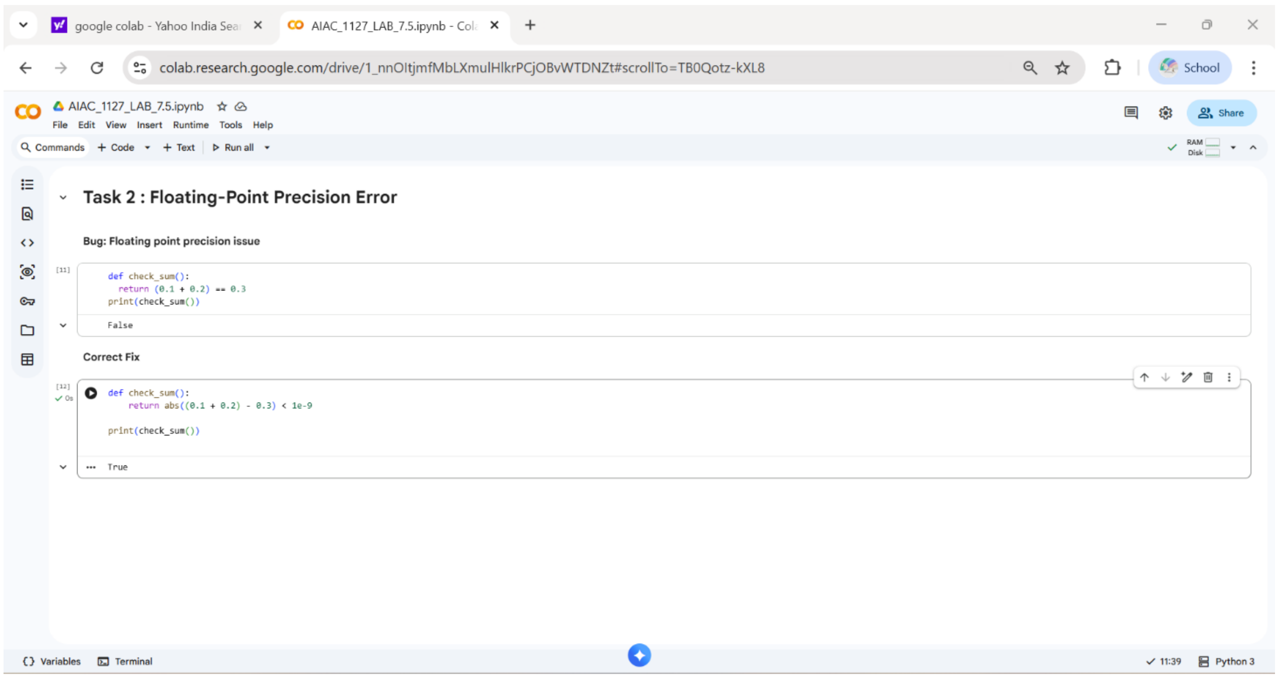
```
[1]
[2]
```

```
# **Task 2 : Floating-Point Precision Error**
```

**Explanation :** Using None instead of a mutable default argument creates a new list on every function call and avoids shared data issues.

## Task 2: Floating-Point Precision Error

### ERROR AND FIXED CODE:



Task 2 : Floating-Point Precision Error

**Bug: Floating point precision issue**

```
def check_sum():
    return (0.1 + 0.2) == 0.3
print(check_sum())
```

```
False
```

**Correct Fix**

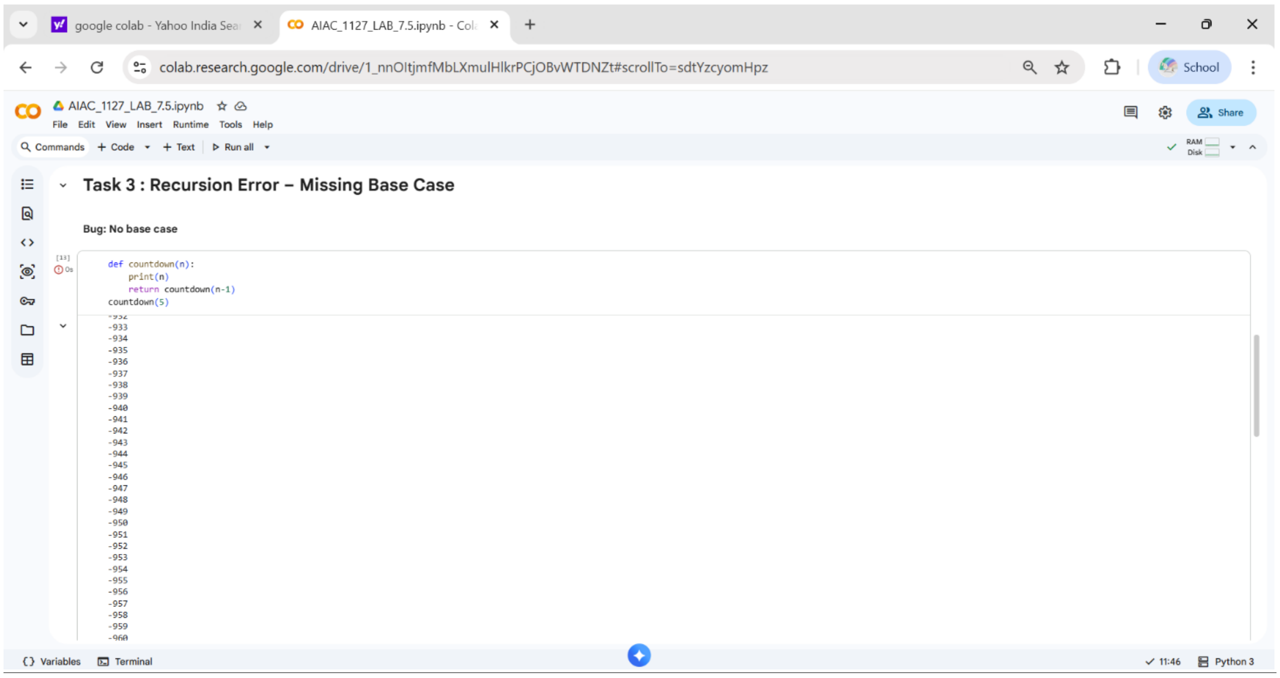
```
def check_sum():
    return abs((0.1 + 0.2) - 0.3) < 1e-9

print(check_sum())
```

```
True
```

**Explanation:** Floating-point values are compared using a tolerance (or math.isclose) instead of direct equality to handle precision errors.

## Task 3: Recursion Error – Missing Base Case

## ERROR AND FIXED CODE :

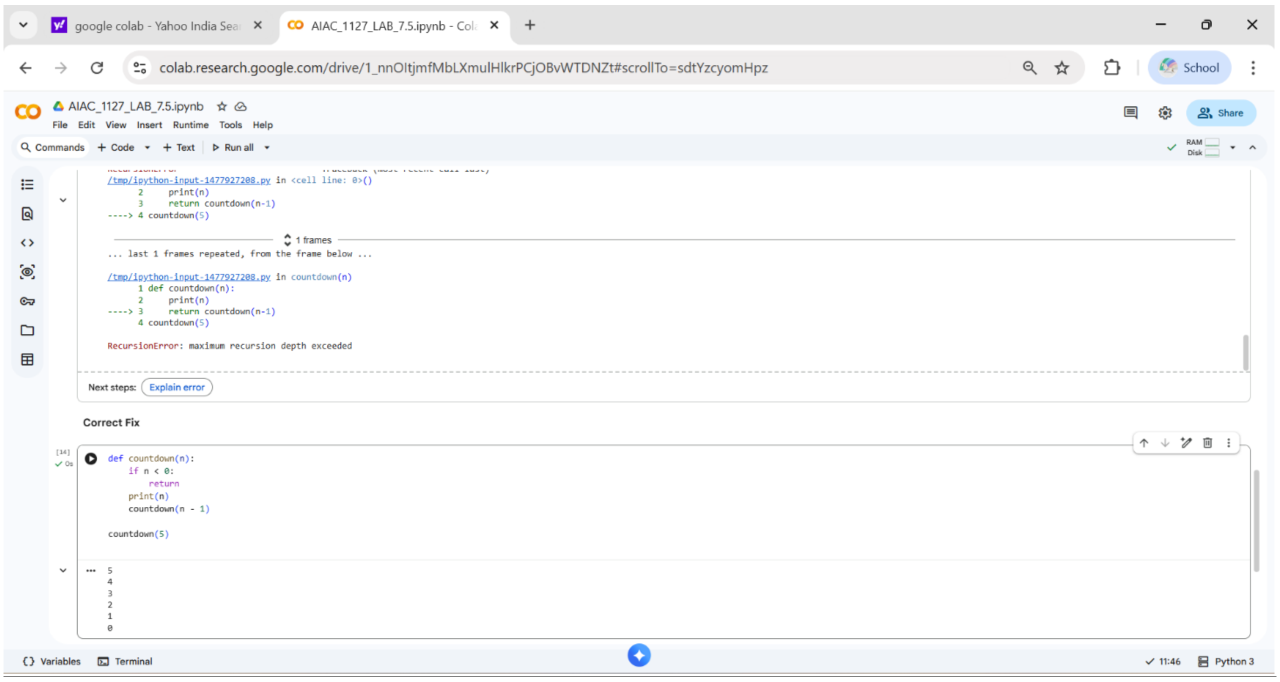


Task 3 : Recursion Error – Missing Base Case

**Bug: No base case**

```
def countdown(n):
    print(n)
    return countdown(n-1)
countdown(5)
```

```
-932
-933
-934
-935
-936
-937
-938
-939
-940
-941
-942
-943
-944
-945
-946
-947
-948
-949
-950
-951
-952
-953
-954
-955
-956
-957
-958
-959
-960
```



```
/tmp/ipython-input-1477927208.py in <cell line: 0>()
      2     print(n)
      3     return countdown(n-1)
----> 4 countdown(5)

... last 1 frames repeated, from the frame below ...

/tmp/ipython-input-1477927208.py in countdown(n)
      1 def countdown(n):
      2     print(n)
----> 3     return countdown(n-1)
      4 countdown(5)

RecursionError: maximum recursion depth exceeded
```

Correct Fix

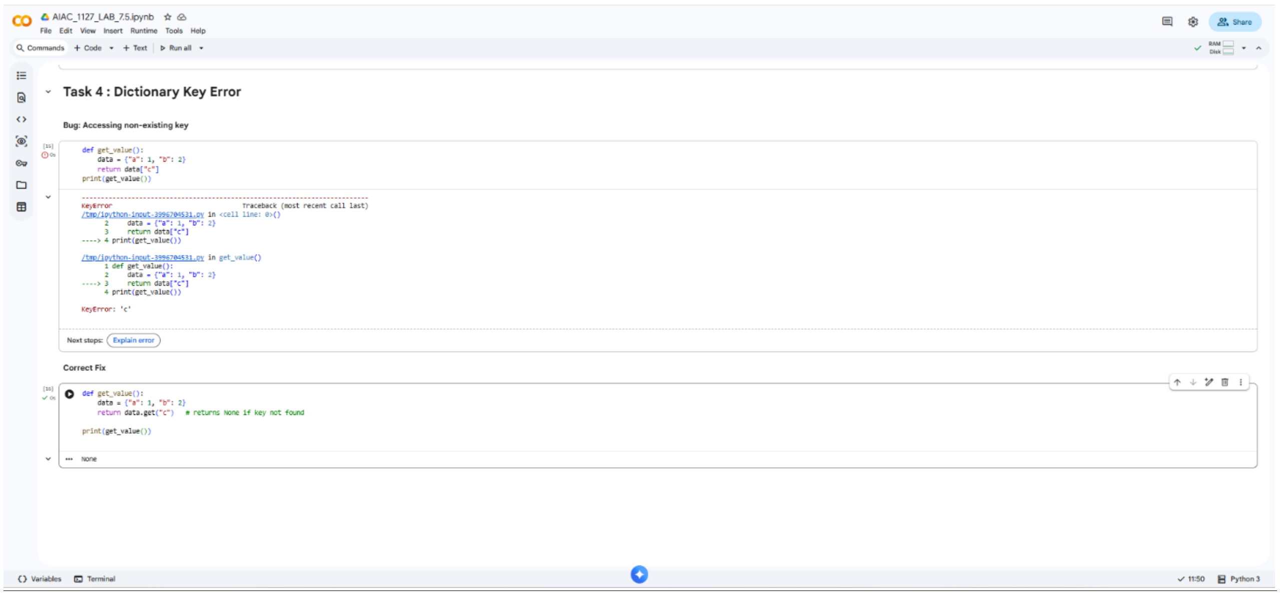
```
def countdown(n):
    if n < 0:
        return
    print(n)
    countdown(n - 1)

countdown(5)
```

```
5
4
3
2
1
0
```

**Explanation:** A base case is added to stop recursive calls and prevent infinite recursion.

## Task 4: Dictionary Key Error

## ERROR AND FIXED CODE:



Task 4 : Dictionary Key Error

Bug: Accessing non-existing key

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

```
---------------------------------------------------------------------------
KeyError                                  Traceback (most recent call last)
/tmp/ipython-input-3996704531.py in <cell line: 0>()
      2     data = {"a": 1, "b": 2}
      3     return data["c"]
----> 4 print(get_value())

/tmp/ipython-input-3996704531.py in get_value()
      1 def get_value():
      2     data = {"a": 1, "b": 2}
----> 3     return data["c"]
      4 print(get_value())

KeyError: 'c'
```

Correct Fix

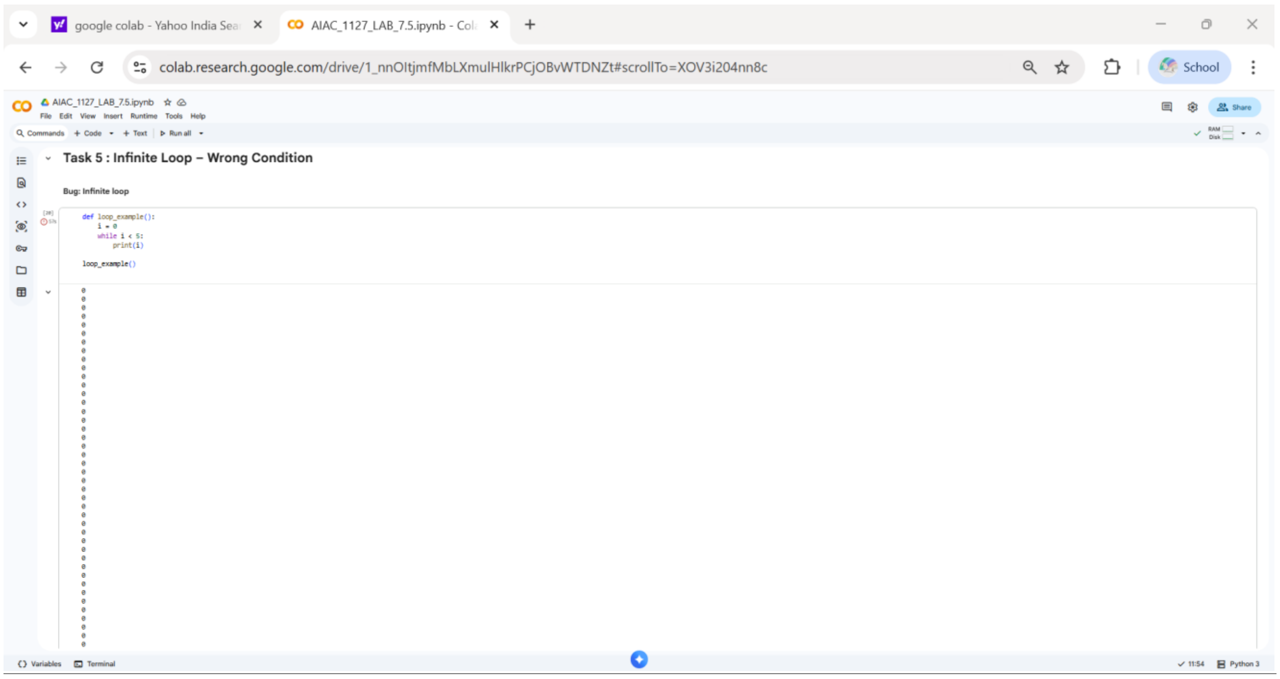
```
def get_value():
    data = {"a": 1, "b": 2}
    return data.get("c")   # returns None if key not found

print(get_value())
```

```
None
```

**Explanation:** Using dict.get() or exception handling prevents KeyError when accessing missing dictionary keys.

## Task 5: Infinite Loop – Wrong Condition

## ERROR AND FIXED CODE:

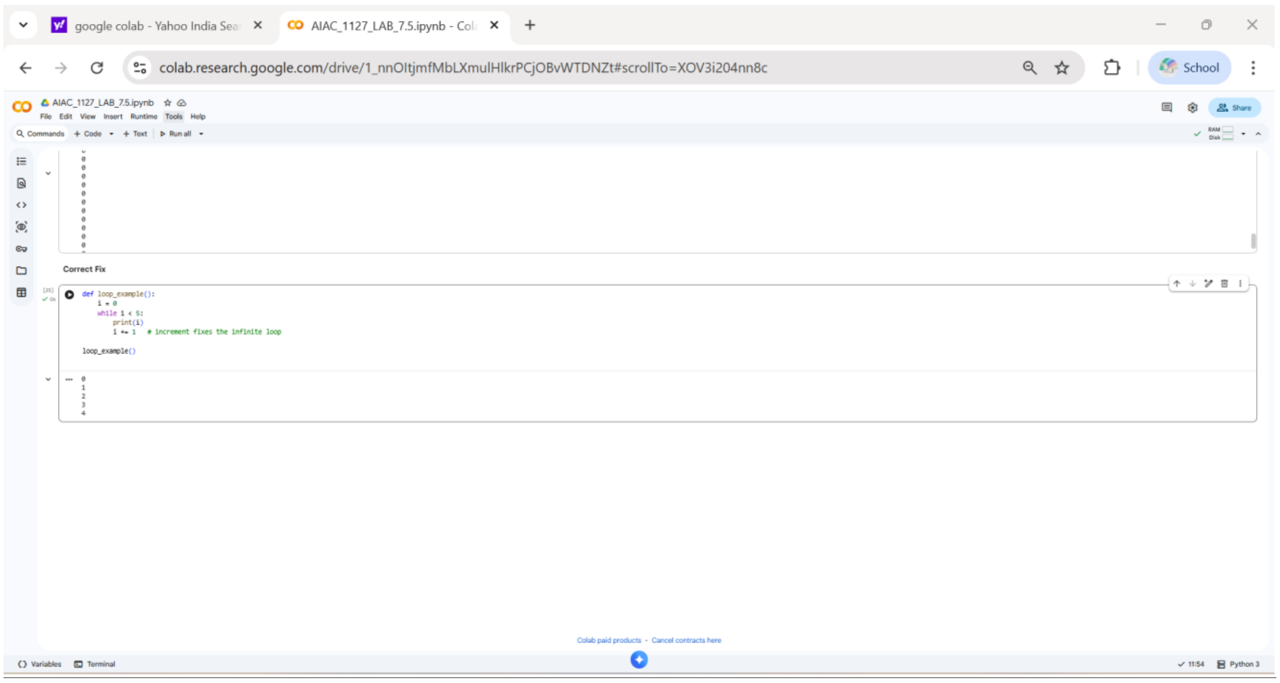


Task 5 : Infinite Loop – Wrong Condition

Bug: Infinite loop

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

loop_example()
```

```
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
```



```
0
0
0
0
0
0
0
0
0
0
0
0
```

Correct Fix

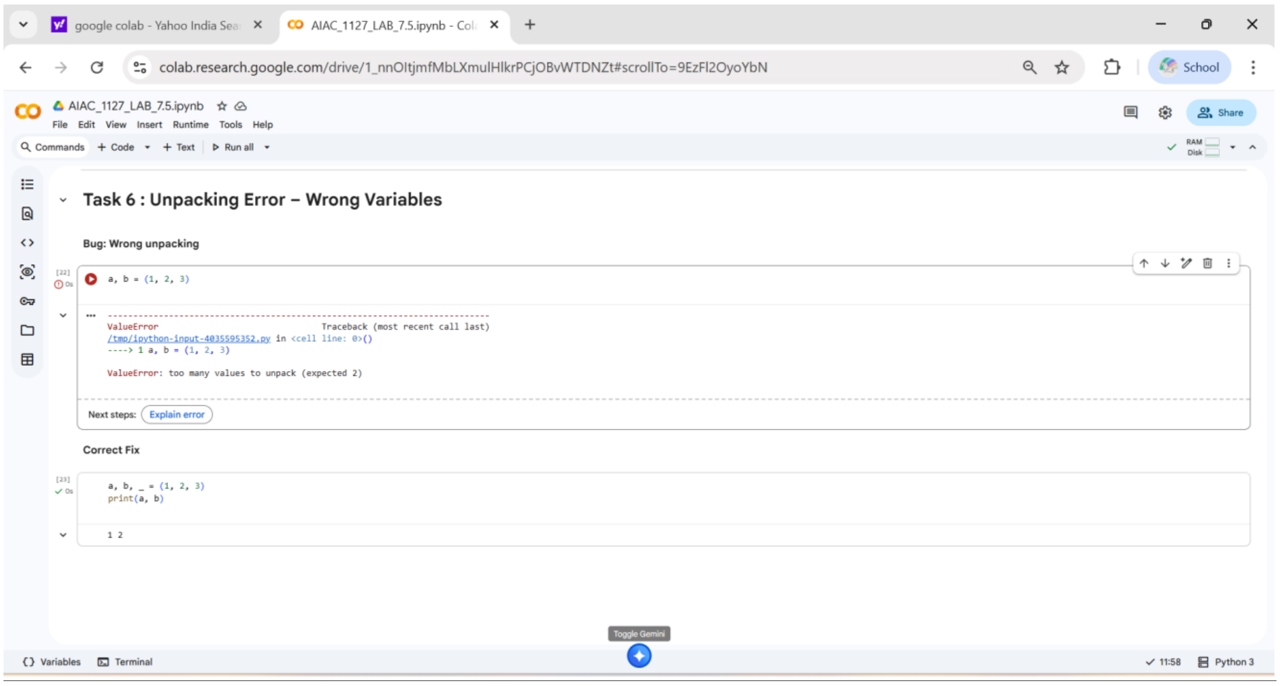
```python
def loop_example():
    i = 0
    while i < 5:
        print(i)
        i += 1   # increment fixes the infinite loop

loop_example()
```

```
0
1
2
3
4
```

**Explanation:** Incrementing the loop variable ensures the loop condition eventually becomes false.

## TASK 6: Unpacking Error – Wrong Variables

## ERROR AND FIXED CODE:



### Task 6 : Unpacking Error – Wrong Variables

**Bug: Wrong unpacking**

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

```
---------------------------------------------------------------------------
ValueError                                Traceback (most recent call last)
/tmp/ipython-input-4035595352.py in <cell line: 0>()
----> 1 a, b = (1, 2, 3)

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

**Correct Fix**

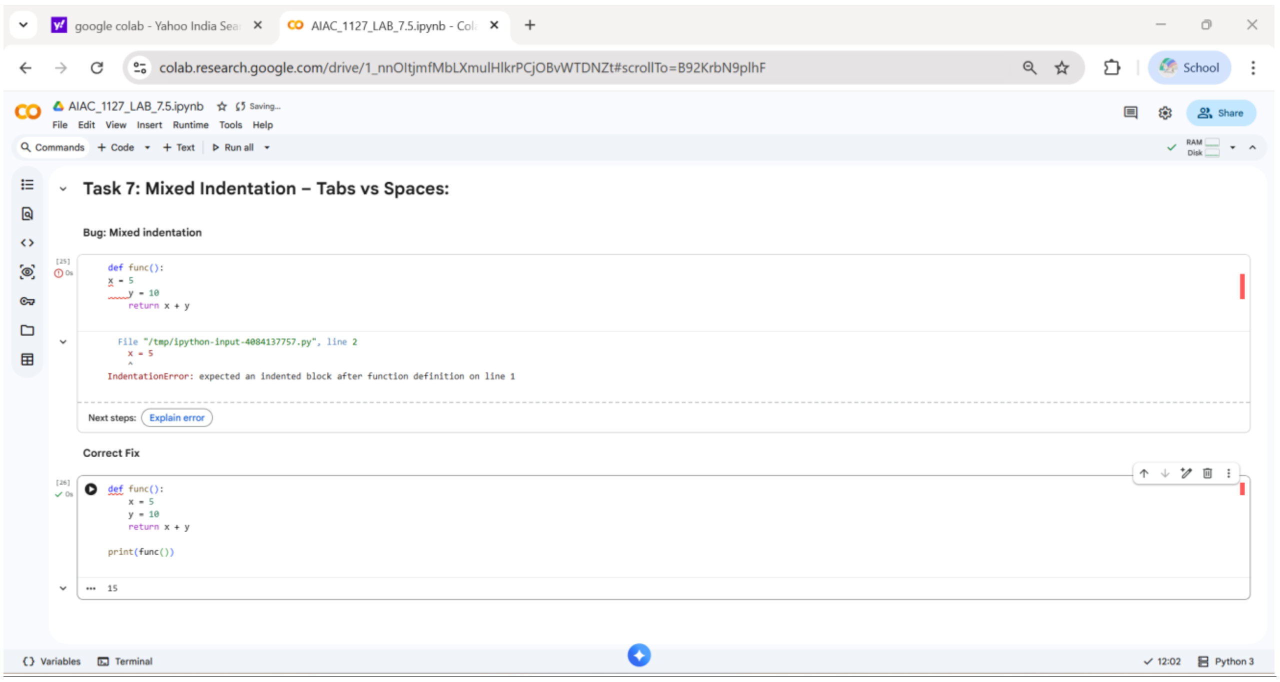
```
a, b, _ = (1, 2, 3)
print(a, b)
```

```
1 2
```

**Explanation:** Correct unpacking is achieved by matching variable count or ignoring extra values using _ or *.

## Task 7: Mixed Indentation – Tabs vs Spaces

## ERROR AND FIXED CODE:



Task 7: Mixed Indentation – Tabs vs Spaces:

Bug: Mixed indentation

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

```
  File "/tmp/ipython-input-4084137757.py", line 2
    x = 5
    ^
IndentationError: expected an indented block after function definition on line 1
```

Correct Fix

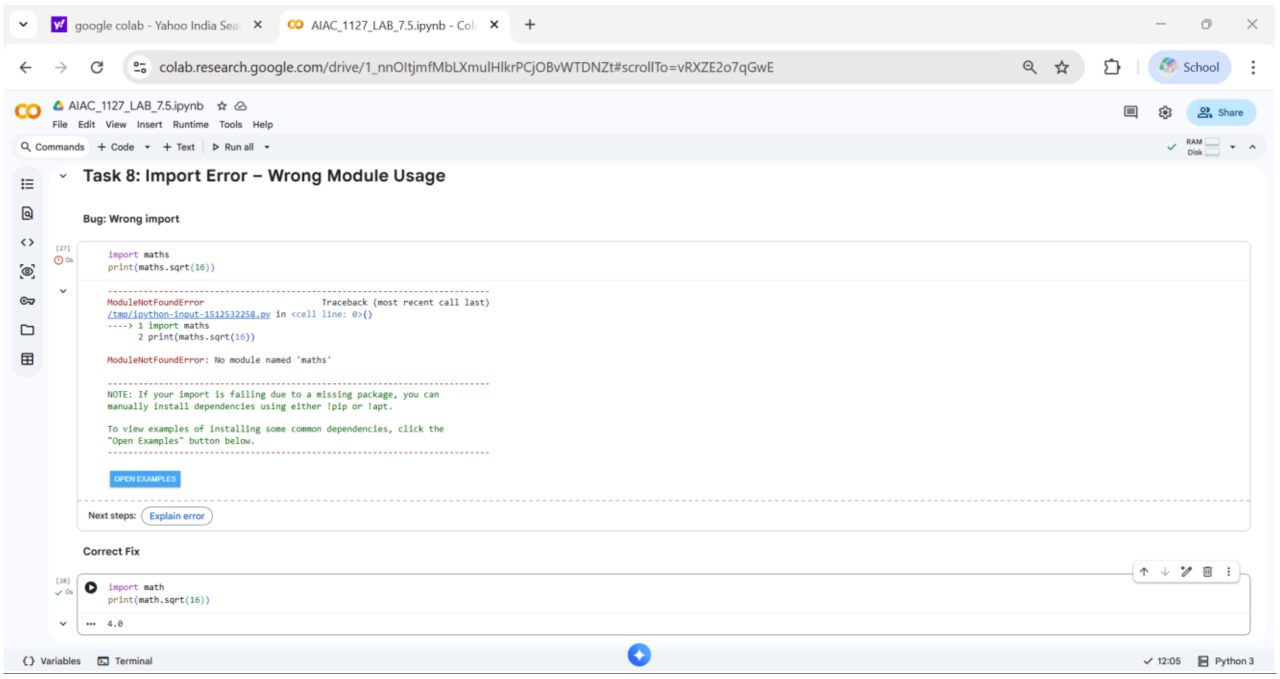
```
def func():
    x = 5
    y = 10
    return x + y

print(func())
```

```
15
```

**EXPLANATION:** Consistent indentation using spaces fixes IndentationError and allows proper code execution.

## Task 8: Import Error – Wrong Module Usage

**EEROR AND FIXED CODE:** Correcting the module name to math resolves the import error.



Task 8: Import Error – Wrong Module Usage

Bug: Wrong import

```
import maths
print(maths.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.
---------------------------------------------------------------------------
```

Correct Fix

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

```
4.0
```

**Explanation**: Correcting the module name to math resolves the import error.