Assignment -7.4

Hall Ticket: 2303A51630

Name: K. Nithin Kumar

Batch -23

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))

Expected Output: Corrected function avoids shared list bug.

```
4 > 02 > 2026 > 7-1.py > ...
1   #error Dont use mutable default argument
2   def add_item(item, items=None):
3       if items is None:
4           items = []
5       items.append(item)
6       return items
7   print(add_item(1,[10,20]))
8   print(add_item(2,[11.11, 22.22]))
9
```

Output:

```
PROBLEMS 11  OUTPUT  DEBUG CONSOLE  TERMINAL  PORTS  GITLENS

PS C:\Users\NITHIN\OneDrive\Desktop\AI - ASS>  c:; cd 'c:\Users
' 'c:\Users\NITHIN\.vscode\extensions\ms-python.debugpy-2025.18
2026\7-1.py'
[10, 20, 1]
[11.11, 22.22, 2]
[100, 200, 3]
PS C:\Users\NITHIN\OneDrive\Desktop\AI - ASS>
```

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

```
4 > 02 > 2026 > 7-2.py > ...
  1   #error Floating-point precision
  2   def check_sum():
  3       return abs((0.1 + 0.2) - 0.3) < 1e-10
  4   print(check_sum())
  5
```

Output:

```
PROBLEMS 14   OUTPUT   DEBUG CONSOLE   TERMINAL   PORTS   GITLEN

PS C:\Users\NITHIN\OneDrive\Desktop\AI - ASS> & C:/Users/NITH
7-2.py"
True
PS C:\Users\NITHIN\OneDrive\Desktop\AI - ASS>
```

Task 3 (Recursion Error – Missing Base Case)

Task: Analyze given code where recursion runs infinitely due tomissing 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.

```
4 > 02 > 2026 > 7-3.py > ...
1   #Error Recursion limit exceeded
2   def countdown(n):
3       if n == 0:
4           return
5       print(n)
6       countdown(n-1)
7   countdown(5)
8
```

Output:

```
PROBLEMS 17  OUTPUT  DEBUG CONSOLE  TERMINAL  PORTS  GITLENS  POSTGR

PS C:\Users\NITHIN\OneDrive\Desktop\AI - ASS> & C:/Users/NITHIN/AppData
7-3.py"
5
4
3
2
1
PS C:\Users\NITHIN\OneDrive\Desktop\AI - ASS>
```

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():

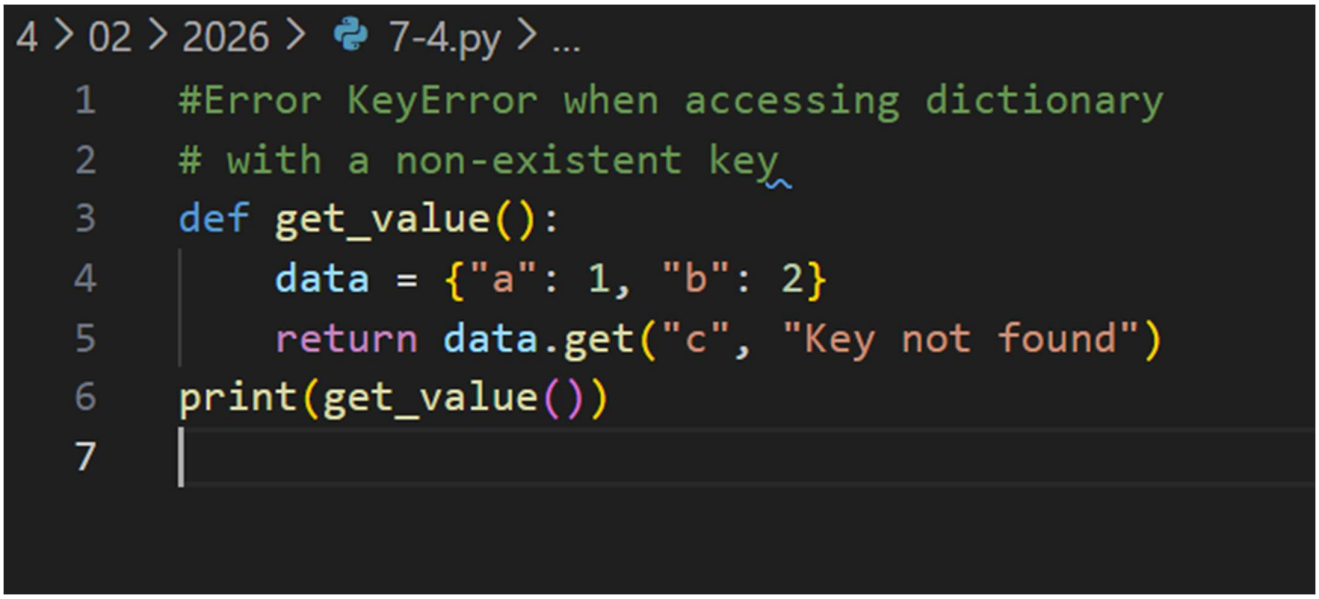
data = {"a": 1, "b": 2}

return data["c"]

print(get_value())

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

CODE:



OUTPUT:

Task 5 (Infinite Loop – Wrong Condition)

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

Bug: Infinite loop

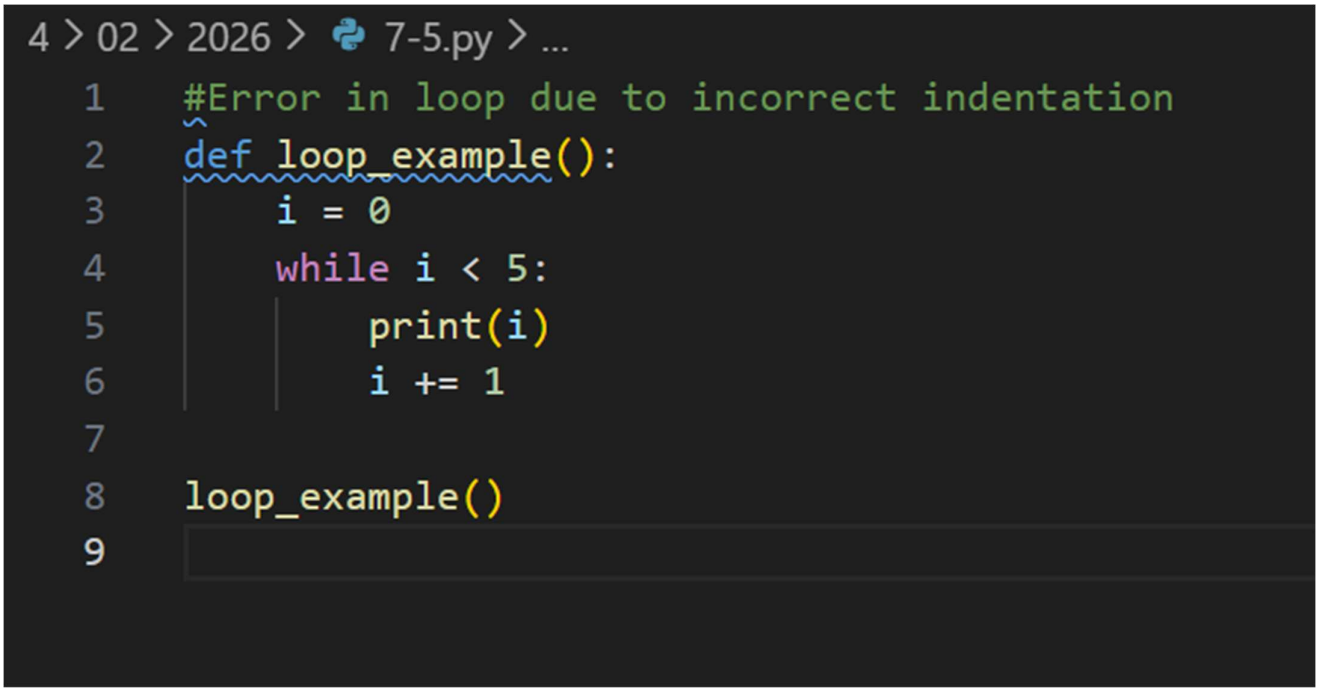
def loop_example():

i = 0

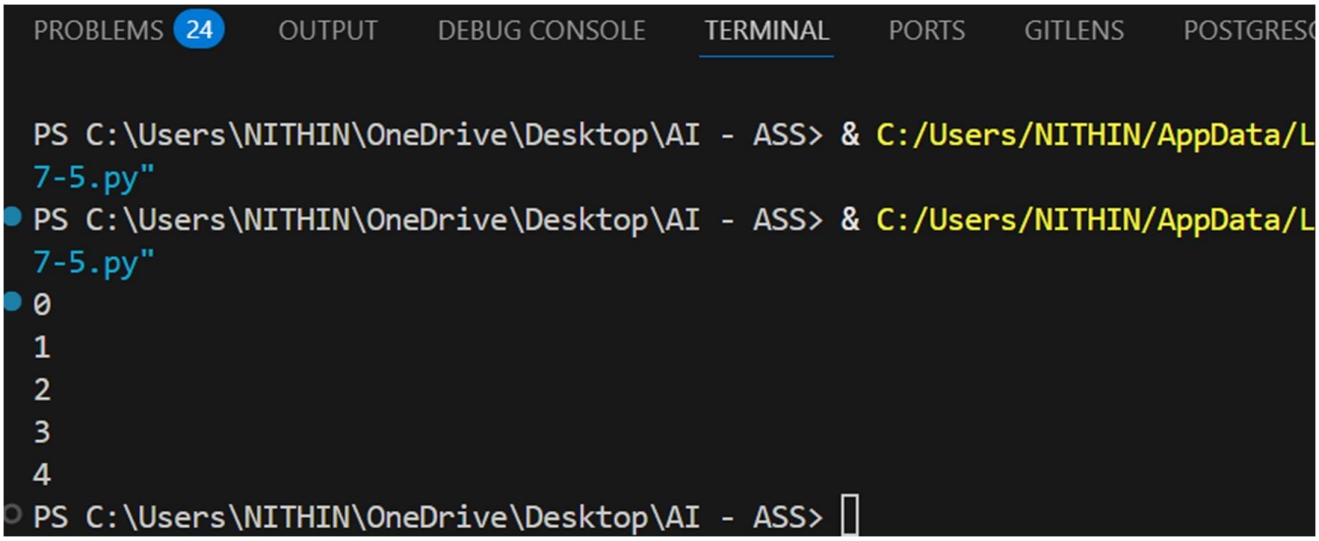
while i < 5:

print(i)

Expected Output: Corrected loop increments i.



Output:



Task 6 (Unpacking Error – Wrong Variables)

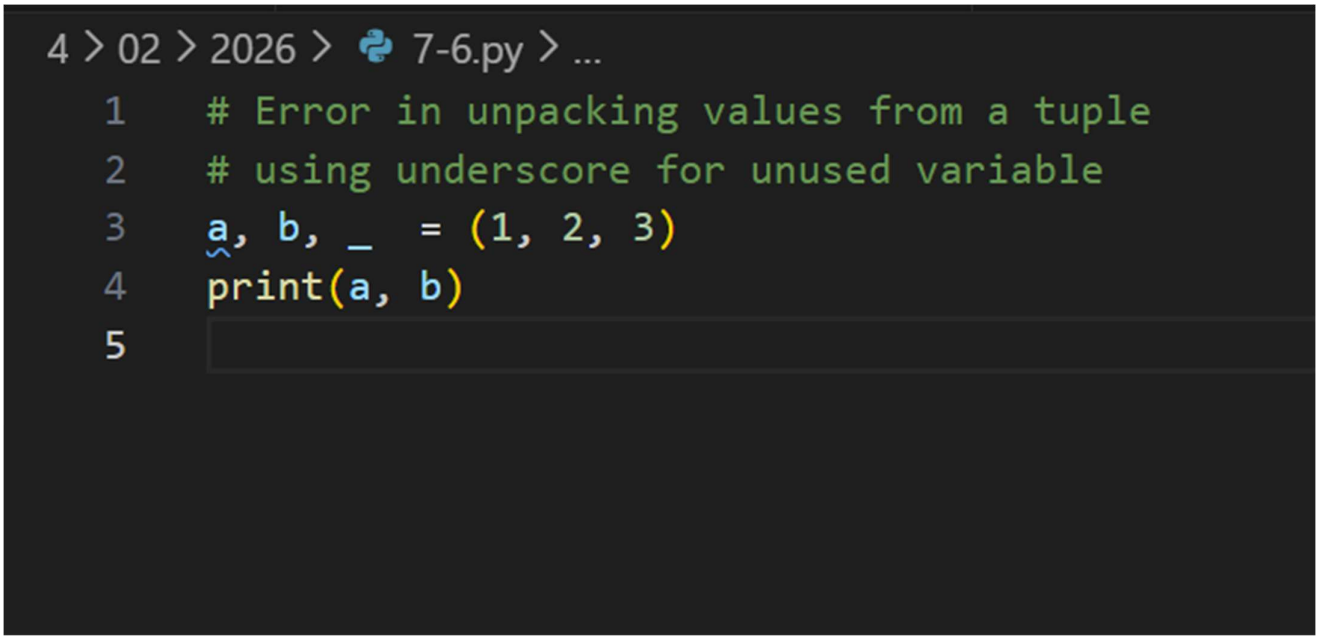
Task: Analyze given code where tuple unpacking fails. Use AI to

fix it.

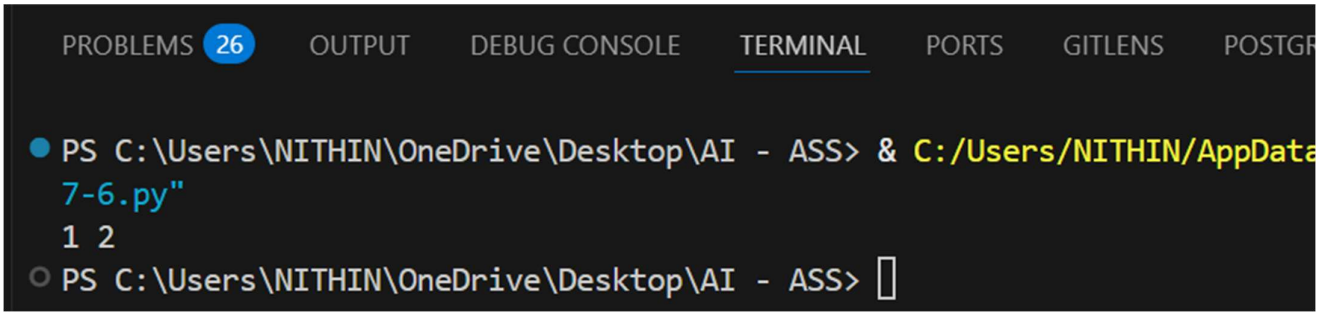
Bug: Wrong unpacking

a, b = (1, 2, 3)

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



OUTPUT:



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.

```
4 > 02 > 2026 > 7-7.py > ...
1   # Bug: Mixed indentation
2   def func():
3       x = 5
4       y = 10
5       return x+y
6   print(func())
7
```

Output:

```
PROBLEMS 33   OUTPUT   DEBUG CONSOLE   TERMINAL   PORTS

PS C:\Users\NITHIN\OneDrive\Desktop\AI - ASS> & C:/Users
7-7.py"
15
PS C:\Users\NITHIN\OneDrive\Desktop\AI - ASS>
```

Task 8 (Import Error – Wrong Module Usage)

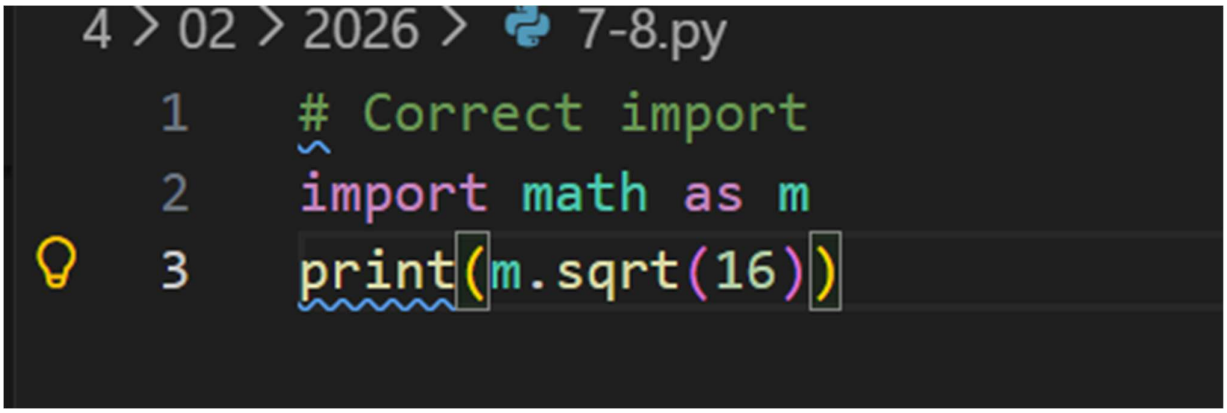
Task: Analyze given code with incorrect import. Use AI to fix.

Bug: Wrong import

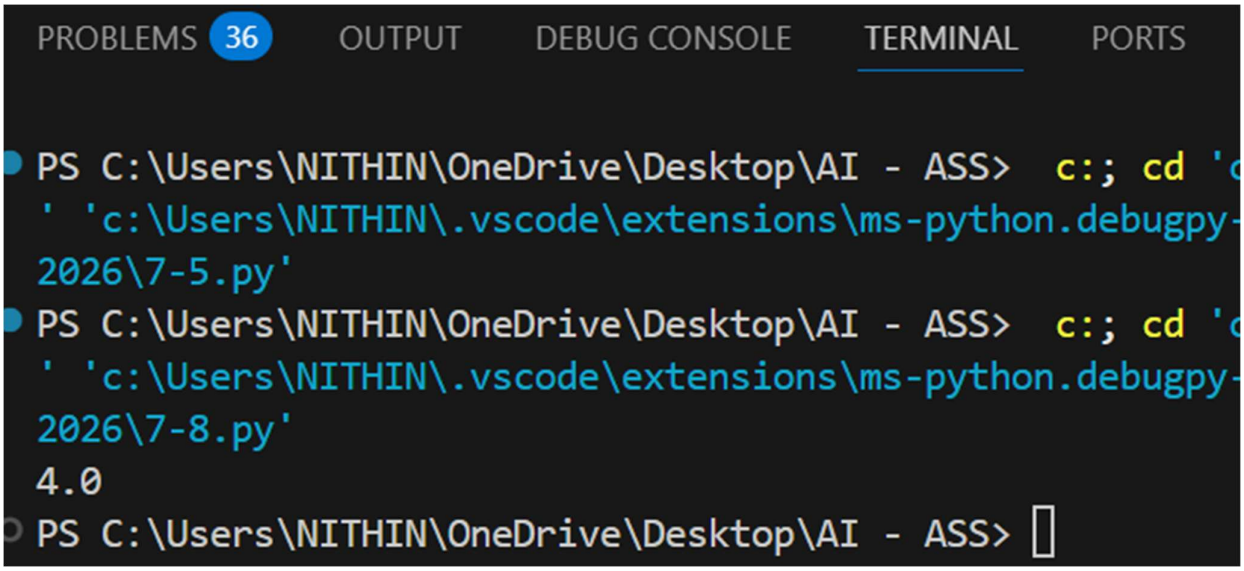
import maths

print(maths.sqrt(16))

Expected Output: Corrected to import math



OutPut:



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]))

Expected Output: Corrected code accumulates sum and returns

after loop.

```
4 > 02 > 2026 > 7-9.py > ...
1   # Task 9 (Logical Error - Incorrect Loop Logic)
2   def total(numbers):
3       total_sum = 0
4       for n in numbers:
5           total_sum += n
6       return total_sum
7   print(total([1,2,3]))
8
```

Output:

```
PROBLEMS 40  OUTPUT  DEBUG CONSOLE  TERMINAL  PORTS  GITLENS  P

PS C:\Users\NITHIN\OneDrive\Desktop\AI - ASS> & C:/Users/NITHIN/App
7-9.py"
6
PS C:\Users\NITHIN\OneDrive\Desktop\AI - ASS>
```

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.

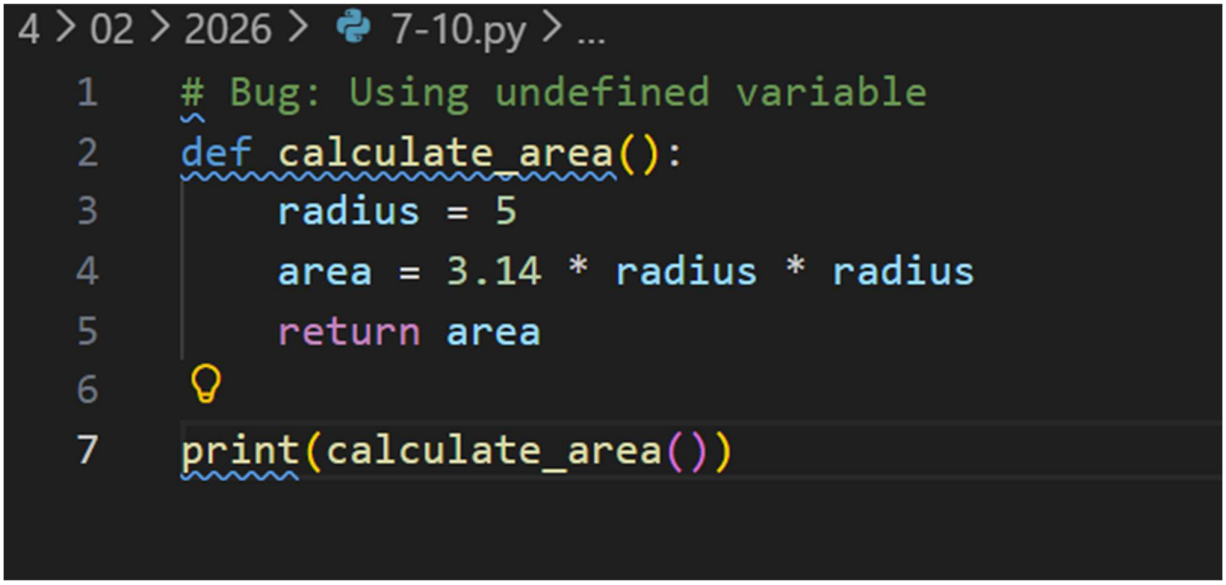
• Add 3 assert test cases for correctness.

Expected Output :

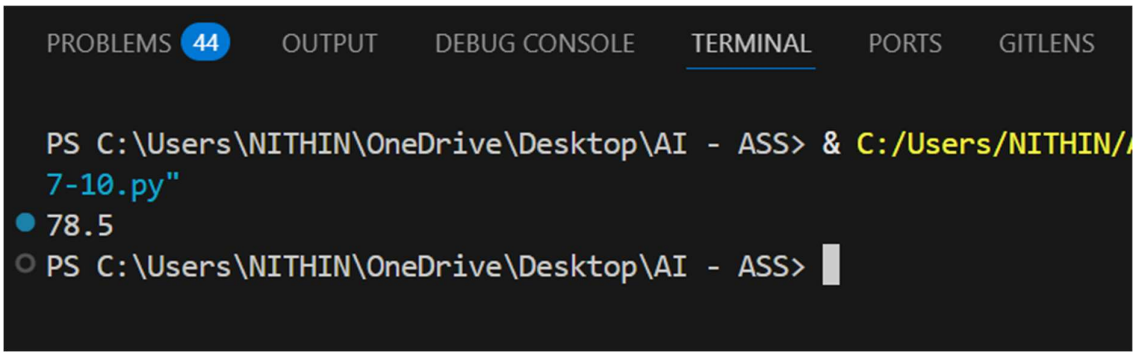
• Corrected code with parameters.

• AI explanation of the bug.

Successful execution of assertions.



OutPut:



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.

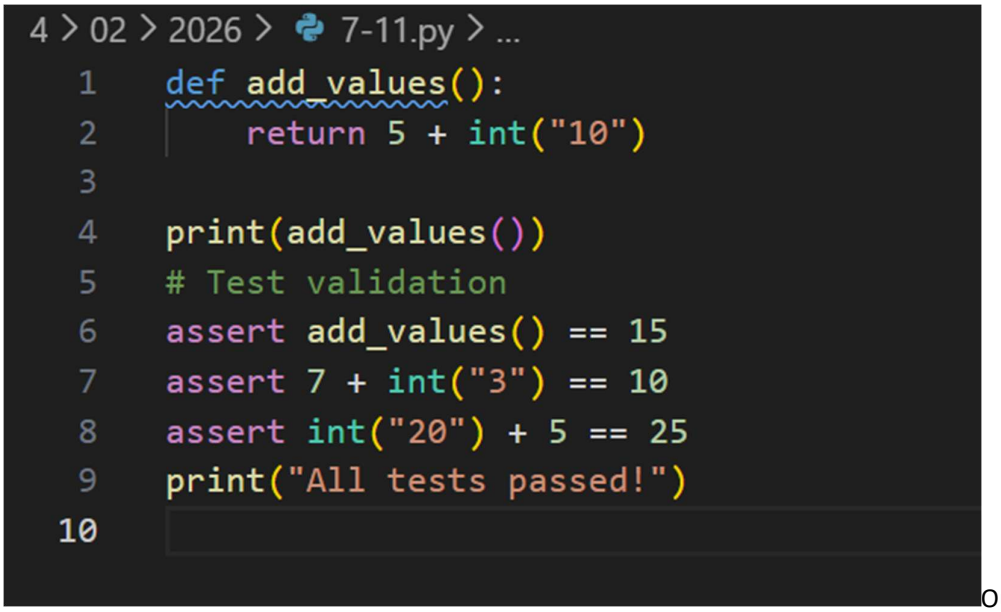
• Fix the code by type conversion (e.g., int("10") or str(5)).

• Verify with 3 assert cases.

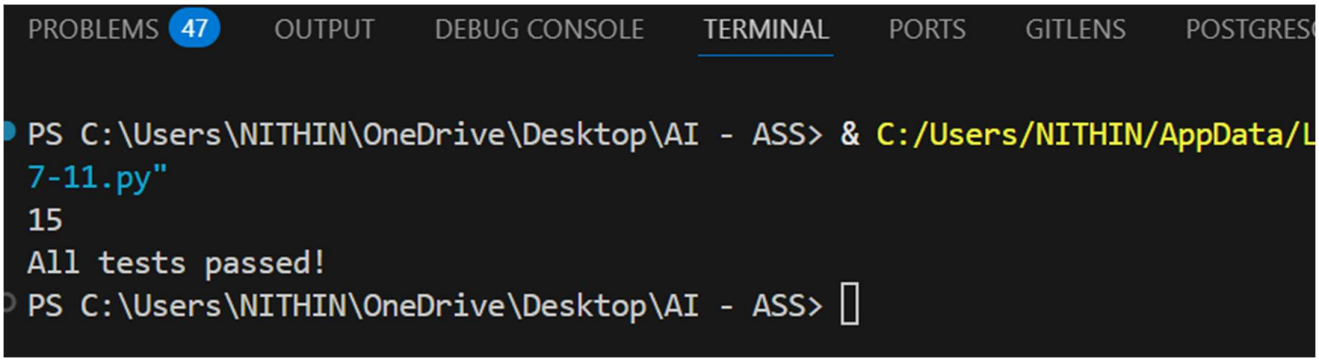
Expected Output #6:

• Corrected code with type handling.

• AI explanation of the fix. Successful test validation.



OutPut:



,

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

```
4 > 02 > 2026 > 7-12.py > ...
1   def combine():
2       return "Numbers: " + str([1, 2, 3])
3   print(combine())
4   assert combine() == "Numbers: [1, 2, 3]"
5   assert "List: " + str([4, 5]) == "List: [4, 5]"
6   assert "Values: " + str([]) == "Values: []"
7
```

OutPut:

```
PROBLEMS 52   OUTPUT   DEBUG CONSOLE   TERMINAL   PORTS

PS C:\Users\NITHIN\OneDrive\Desktop\AI - ASS> & C:/Users/
7-12.py"
Numbers: [1, 2, 3]
PS C:\Users\NITHIN\OneDrive\Desktop\AI - ASS>
```

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.

```
4 > 02 > 2026 > 7-13.py > ...
1   def repeat_text():
2       return "Hello" * int(2.5)
3
4   print(repeat_text())
5
6   assert repeat_text() == "HelloHello"
7   assert "Hi" * int(3.0) == "HiHiHi"
8   assert "A" * int(1.9) == "A"
9
10
```

Output:

```
PROBLEMS 55   OUTPUT   DEBUG CONSOLE   TERMINAL   PORTS

PS C:\Users\NITHIN\OneDrive\Desktop\AI - ASS> & C:/Users/
7-13.py"
HelloHello
PS C:\Users\NITHIN\OneDrive\Desktop\AI - ASS>
```

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.

CODE:

```
4 > 02 > 2026 > 7-14.py > ...
  1  def compute():
  2      value = None
  3      if value is None:
  4          value = 0
  5      return value + 10
  6
  7  print(compute())
  8
  9  assert compute() == 10
 10  assert (0 + 10) == 10
 11  assert (5 + 10) == 15
 12
```

Output:

```
PROBLEMS 58  OUTPUT  DEBUG CONSOLE  TERMINAL  PORTS  GI

PS C:\Users\NITHIN\OneDrive\Desktop\AI - ASS> & C:/Users/N
7-14.py"
10
PS C:\Users\NITHIN\OneDrive\Desktop\AI - ASS>
```

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.

CODE:

```
4 > 02 > 2026 > 7-15.py > ...
1   def sum_two_numbers(a, b):
2       return int(a) + int(b)
3   print(sum_two_numbers("2", "3"))
4   assert sum_two_numbers("2", "3") == 5
5   assert sum_two_numbers("10", "5") == 15
6   assert sum_two_numbers("7", "8") == 15
7
```

OUTPUT:

```
PROBLEMS 61   OUTPUT   DEBUG CONSOLE   TERMINAL   PORTS

PS C:\Users\NITHIN\OneDrive\Desktop\AI - ASS> & C:/Users/
7-15.py"
5
PS C:\Users\NITHIN\OneDrive\Desktop\AI - ASS>
```