AI ASSISTED CODING

Assignment-7.4

HTNO:2303A510B4

Batch:14

Task 1: Debugging a Recursive Calculation Module

Scenario

You are maintaining a utility module in a software project that performs

mathematical computations. One function is meant to calculate the

factorial of a number, but users are reporting crashes or incorrect outputs.

Task Description

You are given a Python function intended to calculate the factorial of a

number using recursion, but it contains logical or syntactical errors (such

as a missing base condition or incorrect recursive call).

Faulty code:

C: > Users > malya > Ai > Untitled-1.py > factorial

```
1   def factorial(n)
2       return n * factorial(n-1)
```

Output:

```
                ^
SyntaxError: expected ':'
PS C:\Users\nithe\OneDrive\Documents\pthon learning>
```

Ln 9, Col 1 Spaces: 4

Explanation:

SYNTAX ERROR TROUBLESHOOTING IN CODE

@workspace /explain Expected ":", Expected ":", Expected ":"

2.1.py:1-5

Workspace

> Used 1 reference

I can see the syntax errors in your code. Python requires colons (:) at the end of function definitions, conditional statements, and other control structures. Here's the corrected version:

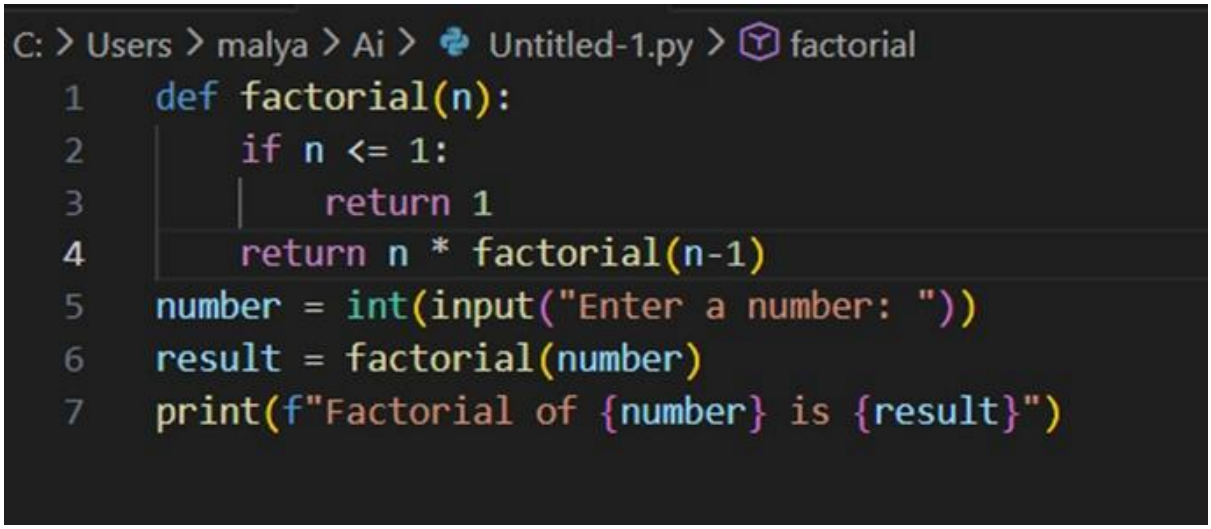
```
def factorial(n):
    if n == 0:
        return 1
    else:
        return n * factorial(n - 1)
```

What was wrong:

- `def factorial(n)` → missing `:` after the function definition
- `if n == 0` → missing `:` after the condition
- `else` → missing `:` after the else statement

How it works:
This is a recursive function that calculates the factorial of a number. It uses the base case `n == 0` (which equals 1) and recursively multiplies `n` by the factorial of `n - 1` until it reaches zero.

Code after fixing:



Output:

Task 2: Fixing Data Type Errors in a Sorting Utility

Scenario

You are developing a data processing script that sorts user input values.

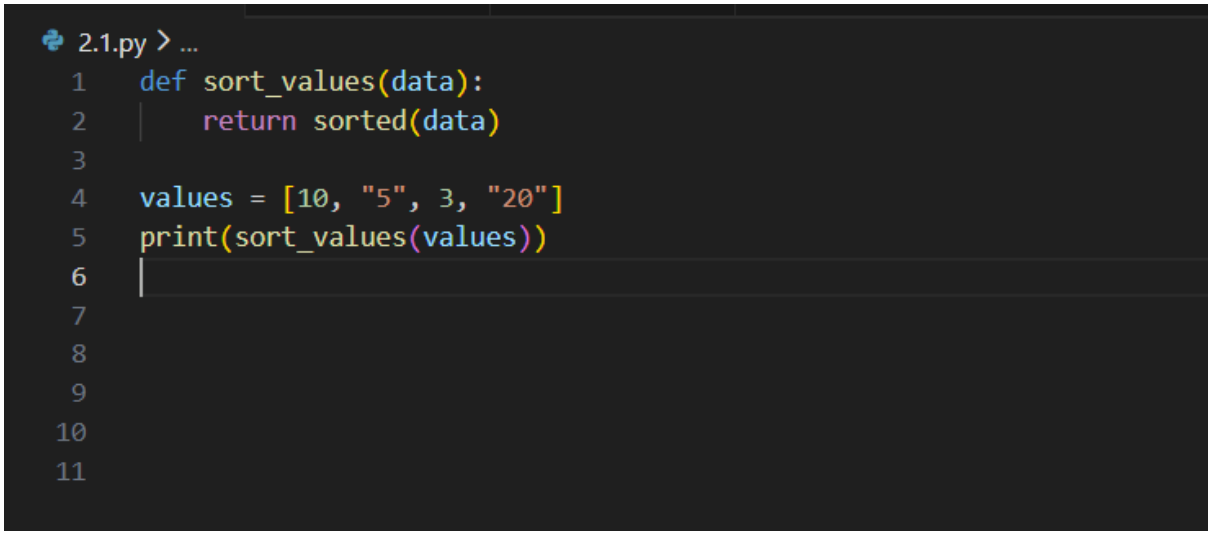
The program crashes when users enter mixed data types.

Task Description

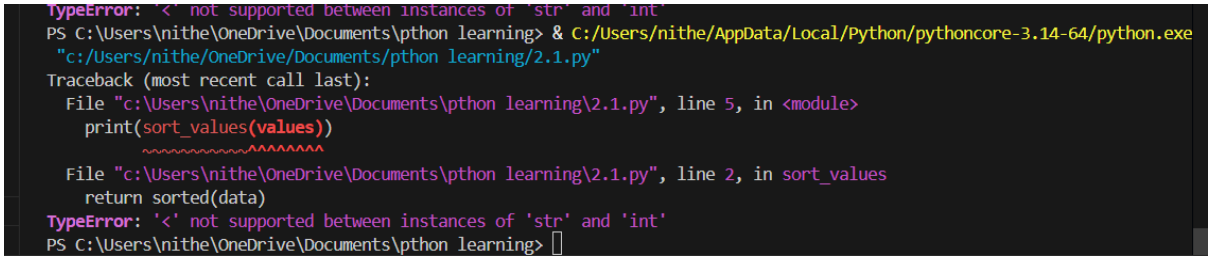
You are provided with a list-sorting function that fails due to a

TypeError caused by mixed data types (e.g., integers and strings).

Faulty code:



Output:



Code after fixing error:

```python
def sort_values(data):
    converted_data = list(map(int, data))
    return sorted(converted_data)

values = [10, "5", 3, "20"]
print(sort_values(values))
```

Output:

```
PS C:\Users\nithe\OneDrive\Documents\pthon learning> & C:/Users/nithe/AppData/Local/Python/pythoncore-3.14-64/python.exe
 "c:/Users/nithe/OneDrive/Documents/pthon learning/2.1.py"
[3, 5, 10, 20]
PS C:\Users\nithe\OneDrive\Documents\pthon learning>
```

Explaination of code improvement:

The code has a bug. Here's what's happening:

data = [10, "5", 3, "20", 1]

print(sorted(data))

TheProblem:

You're trying to sort a list that contains both integers and strings. In Python 3, you cannot compare integers with strings directly, so sorted() fails with a TypeError.

Why it fails:

- The list has: 10 (int), "5" (string), 3 (int), "20" (string), 1 (int)
- When sorting, Python tries to compare 10 with "5", but integers and strings aren't comparable

Task 3: Improving File Handling Reliability

Scenario

A backend script reads data from files regularly. Over time, the system

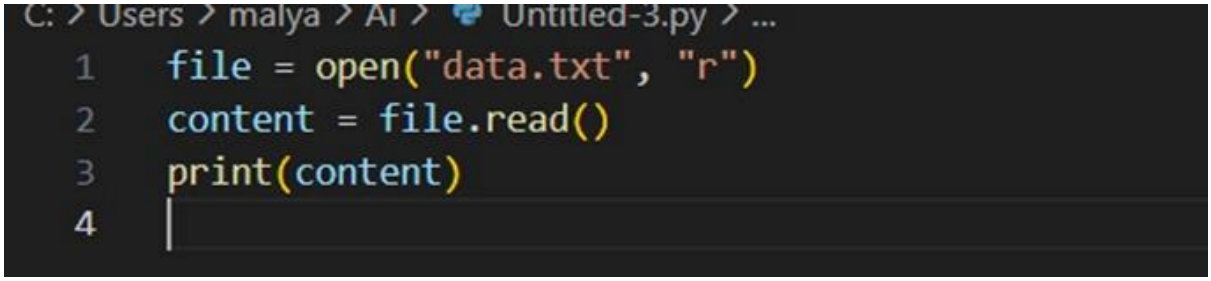
shows performance issues due to improper resource management.

Task Description

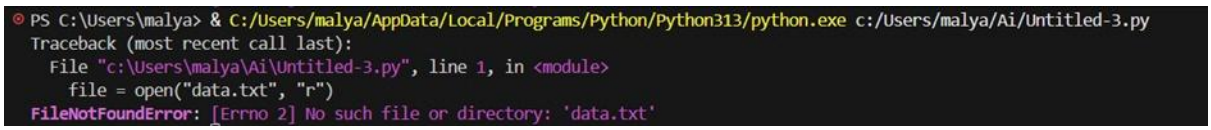
You are given a Python file-handling snippet that opens a file but does

not explicitly close it.

Faulty code:



Output:



Code after fixing;

```python
# Create sample data file if it doesn't exist
try:
    with open("data.txt", "w") as file:
        file.write("Hello, World!\nThis is sample data.\nFile handling in Python!")
except Exception as e:
    print(f"Error creating file: {e}")

# Read and display the file
try:
    with open("data.txt", "r") as file:
        content = file.read()
        print("File Content:")
        print(content)
except FileNotFoundError:
    print("Error: data.txt file not found!")
```

Output:

```
PS C:\Users\malya> & C:/Users/malya/AppData/Local/Programs/Python/Python313/python.exe c:/Users/malya/Ai/Untitled-3.py
File Content:
Hello, World!
This is sample data.
File handling in Python!
PS C:\Users\malya>
```

Task 4: Handling Runtime Errors Gracefully in Loops

Scenario

You are working on a data analysis script that processes a list of values.

Some values cause runtime errors, but the program should continue

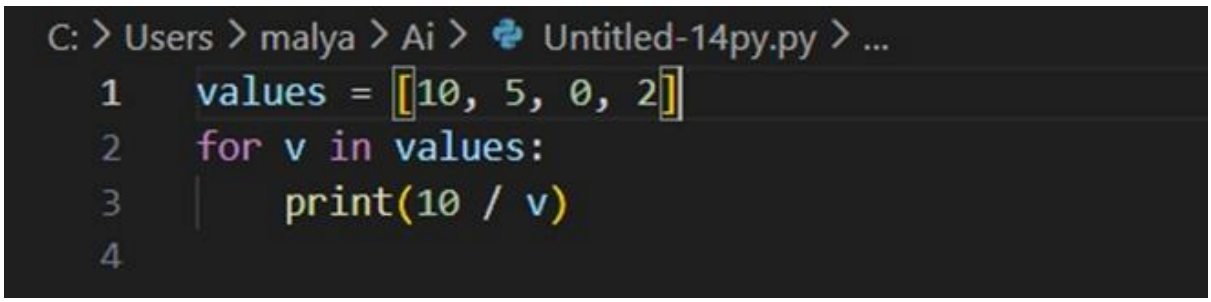
processing remaining data.

Task Description

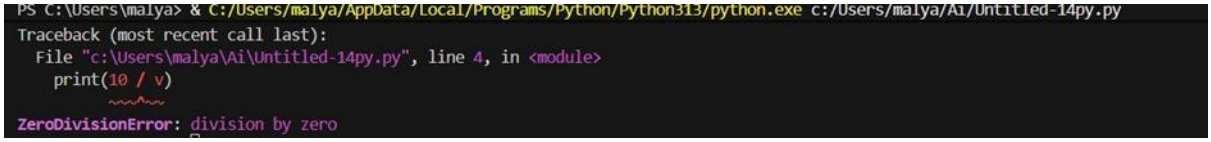
You are provided with a code snippet containing a ZeroDivisionError

inside a loop.

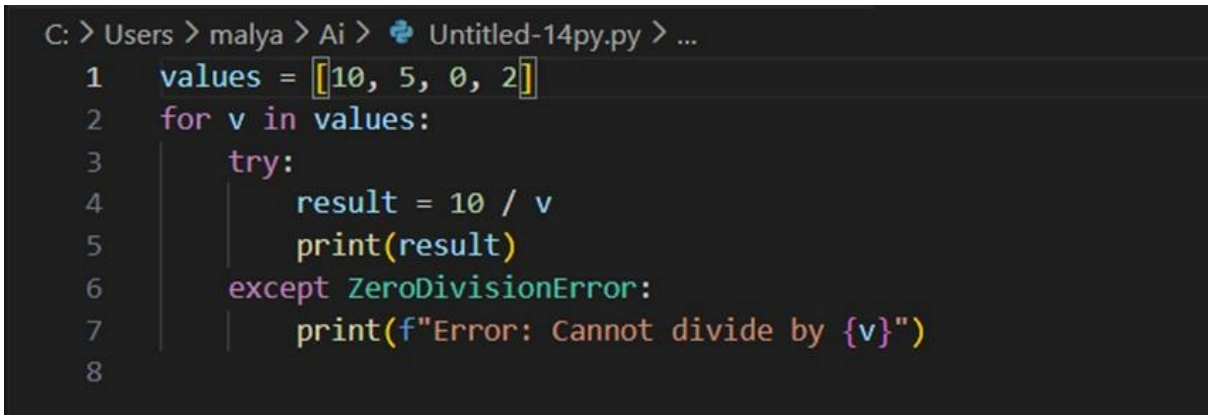
Faulty code:



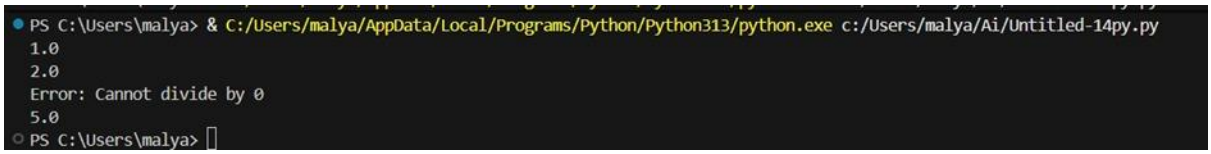
Output:



Codeafter fixing error:



Output:



Task 5: Debugging Class Initialization Errors

Scenario

A class written by a junior developer is throwing unexpected errors when

objects are created or attributes are accessed.

Task Description

You are given a Python class with:

• Incorrect __init__ parameters

• Missing or incorrect attribute references (e.g., missing self)

Faulty code:

C: > Users > malya > Ai > Untitled-5.py > ...

```
1   class Student:
2       def __init__(name, age):
3           self.name = name
4           self.age = age
5
6       def display():
7           print(name, age)
8
9   s = Student("Teja", 20)
10  s.display()
11
```

Output:

```
PS C:\Users\malya> & C:/Users/malya/AppData/Local/Programs/Python/Python313/python.exe c:/Users/malya/Ai/Untitled-5.py
Traceback (most recent call last):
  File "c:\Users\malya\Ai\Untitled-5.py", line 9, in <module>
    s = Student("Teja", 20)
TypeError: Student.__init__() takes 2 positional arguments but 3 were given
PS C:\Users\malya>
```

Code after fixing error:

C: > Users > malya > Ai > Untitled-5.py > Student

```
1   class Student:
2       def __init__(self, name, age):
3           self.name = name
4           self.age = age
5
6       def display(self):
7           print(f"{self.name}, {self.age}")
8
9   s = Student("Teja", 20)
10  s.display()
11
```

Output:


PS C:\Users\malya> & C:/Users/malya/AppData/Local/Programs/Python/Python313/python.exe c:/Users/malya/Ai/Untitled-5.py
Teja, 20