

## ASSIGNMENT-7.4

SHAIK SALIHA

2503A52L13

### Task Description #1:

• Introduce a buggy Python function that calculates the factorial of a number using recursion. Use Copilot or Cursor AI to detect and fix the logical or syntax errors

```
def factr(n):  
    if n == 0:  
        return 0  
    elif n == 1:  
        return 1  
    else:  
        return n * factr(n - 2)  
  
print(factr("5"))
```

### PROMPT:

Write a recursive function to calculate the factorial of a number in Python. Also, fix any bugs in the implementation if present. Additionally, include a comment or section for an Armstrong Number Checker using AI-generated Python code.

### CODE:

# Buggy recursive factorial function

```
def factorial(n):  
    if n == 0:  
        return 0 # Bug: should return 1 for factorial(0)  
    else:  
        return n * factorial(n - 1)
```

# Fixed version

```
def factorial_fixed(n):  
    if n == 0:
```

```

1  # Buggy recursive factorial function
2  def factorial(n):
3      if n == 0:
4          return 0 # Bug: should return 1 for factorial(0)
5      else:
6          return n * factorial(n - 1)
7
8  # Fixed version
9  def factorial_fixed(n):
10     if n == 0:
11         return 1
12     else:
13         return n * factorial_fixed(n - 1)
14
15     print(factorial(5)) # This will cause a recursion error
16     print(factorial_fixed(5)) # This will correctly return 120
17     # Armstrong Number Checker using AI-generated Python code

```

PROBLEMS

OUTPUT

DEBUG CONSOLE

TERMINAL

PORTS

powershell

+

⌵

⋮

+ CategoryInfo

: ParserError: (:) [], ParentContainsErrorRecordException

+ FullyQualifiedErrorId

: AmpersandNotAllowed

PS C:\Users\noush\OneDrive\Desktop\2403A52412ai> & C:\Users\noush\AppData\Local\Programs\Python\Python313\python.exe c:/Users/noush/OneDrive/Desktop/2403A52412ai/factorial.py

0

120

PS C:\Users\noush\OneDrive\Desktop\2403A52412ai>

Select Encoding

### OBSERVATION:

1. The code demonstrates a buggy and a corrected recursive factorial function in Python.
2. The buggy version returns 0 for factorial(0), while the fixed version correctly returns 1.

### **Task Description #2:**

- Provide a list sorting function that fails due to a type error (e.g., sorting list with mixed integers and strings). Prompt AI to detect the issue and fix the code for consistent sorting.

```
def sort_list(data):  
    return sorted(data)  
  
items = [3, "apple", 1, "banana", 2]  
print(sort_list(items))
```

### PROMPT:

Write a Python function that attempts to sort a list containing both integers and strings, causing a `TypeError`. Then, detect and fix the issue so the list can be sorted consistently.

### CODE:

# Function that fails due to `TypeError` when sorting mixed types

```
def sort_mixed_list(lst):
```

```
    return sorted(lst)
```

# Example usage (this will raise a `TypeError`)

```
data = [3, "apple", 1, "banana"]
```

```
print("Original list:", data)
```

```
try:
```

```
    print("Sorted (will fail):", sort_mixed_list(data))
```

```
except TypeError as e:
```

```
# Fixed version: convert all elements to strings before sorting

def sort_mixed_list_fixed(lst):
    return sorted(lst, key=str)

print("Sorted (fixed):", sort_mixed_list_fixed(data))
```

```

1  # Function that fails due to TypeError when sorting mixed types
2  def sort_mixed_list(lst):
3      return sorted(lst)
4
5  # Example usage (this will raise a TypeError)
6  data = [3, "apple", 1, "banana"]
7  print("Original list:", data)
8  try:
9      print("Sorted (will fail):", sort_mixed_list(data))
10 except TypeError as e:
11     print("TypeError:", e)
12
13 # Fixed version: convert all elements to strings before sorting
14 def sort_mixed_list_fixed(lst):
15     return sorted(lst, key=str)
16
17 print("Sorted (fixed):", sort_mixed_list_fixed(data))
18

```

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS powershell + - [ ] [X] ... [ ]

```

xed_list
    return sorted(lst)
TypeError: '<' not supported between instances of 'str' and 'int'
PS C:\Users\noush\OneDrive\Desktop\2403A52412ai> & C:\Users\noush\AppData\Local\Programs\Python\Python313\python.exe c:/Users/noush/OneDrive/Desktop/2403A52412ai/ass.py
Original list: [3, 'apple', 1, 'banana']
TypeError: '<' not supported between instances of 'str' and 'int'
Sorted (fixed): [1, 3, 'apple', 'banana']
PS C:\Users\noush\OneDrive\Desktop\2403A52412ai>

```

OBSERVATION:

This code demonstrates that sorting a list with mixed data types (integers and strings) in Python raises a `TypeError`. It then provides a fixed version of the sorting function that converts all elements to strings before sorting, allowing the mixed list to be sorted without errors.

### Task Description #3:

- Write a Python snippet for file handling that opens a file but forgets to close it. Ask Copilot or Cursor AI to improve it using the best practice (e.g., with `open()` block).

Code1

```
with open("example.txt", "w") as f:  
    f.write("Hello, world!")
```

Code2

```
f1 = open("data1.txt", "w")  
f2 = open("data2.txt", "w")  
  
f1.write("First file content\n")  
f2.write("Second file content\n")  
  
print("Files written successfully")
```

Code3

```
data = open("input.txt", "r").readlines()  
output = open("output.txt", "w")  
  
for line in data:  
    output.write(line.upper())  
  
print("Processing done")
```

Code4

```

f = open("numbers.txt", "r")
nums = f.readlines()

squares = []
for n in nums:
    n = n.strip()
    if n.isdigit():
        squares.append(int(n) * int(n))

f2 = open("squares.txt", "w")
for sq in squares:
    f2.write(str(sq) + "\n")

print("Squares written")

```

### PROMPT:

Write a Python snippet that demonstrates bad file handling by opening a file and forgetting to close it, then improve it using the best practice ([with open\(\)](#) block). Add print statements to confirm each file write operation.

### CODE:

# Bad practice: opens a file but forgets to close it

# Bad practice: opens a file but forgets to close it

```
file = open('example.txt', 'w')
```

```
file.write('Hello, world!')
```

# file.close() is missing

```
print("Wrote to example.txt using bad practice (file not closed explicitly)")
```

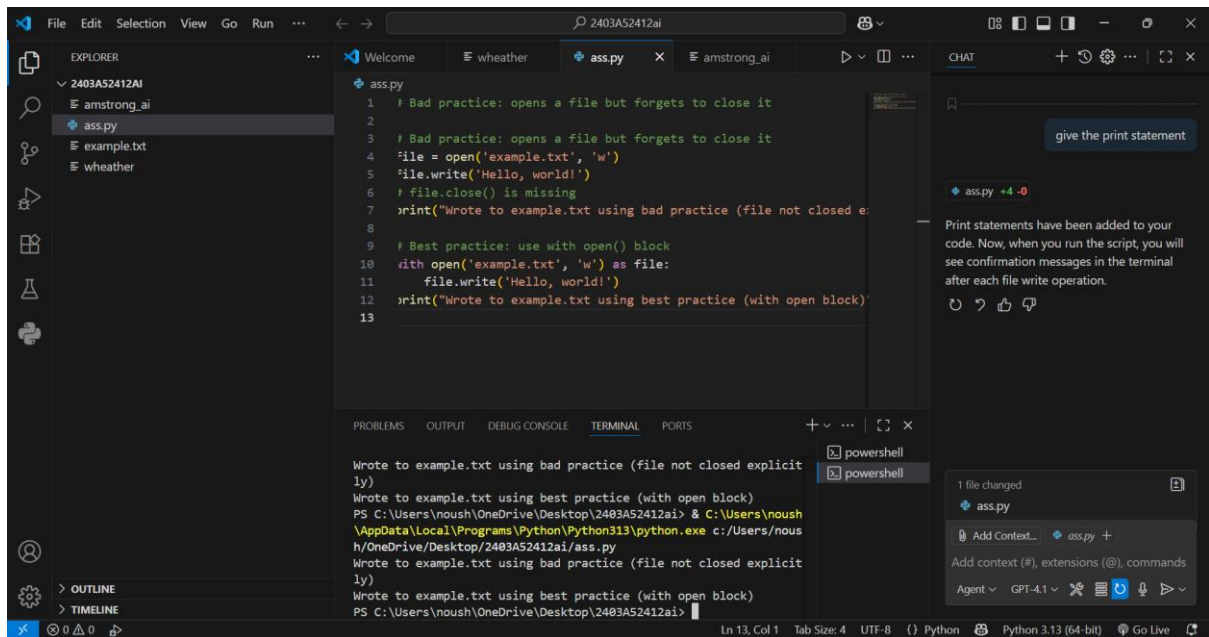
# Best practice: use with open() block

with open('example.txt', 'w') as file:

```
    file.write('Hello, world!')
```

```
print("Wrote to example.txt using best practice (with open block)")
```

### CODE AND OUTPUT:



## OBSERVATION:

1. The code first writes to a file without closing it, which is bad practice.
2. It then demonstrates the correct way using a with open() block, ensuring the file is properly closed.

## **Task Description #4:**

- Provide a piece of code with a ZeroDivisionError inside a loop. Ask AI to add error handling using try-except and continue execution safely.

```
def compute_ratios(values):
    results = []
    for i in range(len(values)):
        for j in range(i, len(values)):
            ratio = values[i] / (values[j] - values[i])
            results.append((i, j, ratio))
    return results

nums = [5, 10, 15, 20, 25]
print(compute_ratios(nums))
```

## PROMPT:

Write a Python loop that causes a ZeroDivisionError when dividing by zero, then improve the code by adding try-except error handling to continue execution safely.

## CODE:

# Code with ZeroDivisionError inside a loop

```
numbers = [5, 2, 0, 3]
```

```
for n in numbers:
```

```
    result = 10 / n # This will raise ZeroDivisionError when n is 0
```

```
    print(f"10 / {n} = {result}")
```

```
# Improved version with error handling
```

```
print("\nWith error handling:")
```

```
for n in numbers:
```

```
    try:
```

```
        result = 10 / n
```

```
        print(f"10 / {n} = {result}")
```

```
    except ZeroDivisionError:
```

```
        print(f"Cannot divide by zero for n = {n}")
```

CODE AND OUTPUT:



```
ass.py
1  # Code with ZeroDivisionError inside a loop
2  numbers = [5, 2, 0, 3]
3  for n in numbers:
4      result = 10 / n # This will raise ZeroDivisionError when n is 0
5      print(f"10 / {n} = {result}")
6
7  # Improved version with error handling
8  print("\nWith error handling:")
9  for n in numbers:
10     try:
11         result = 10 / n
12         print(f"10 / {n} = {result}")
13     except ZeroDivisionError:
14         print(f"Cannot divide by zero for n = {n}")
15
```

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS

10 / 5 = 2.0  
10 / 2 = 5.0  
Traceback (most recent call last):  
File "c:\Users\noush\OneDrive\Desktop\2403A52412ai\ass.py", line 4, in <module>  
result = 10 / n # This will raise ZeroDivisionError when n is 0  
ZeroDivisionError: division by zero  
PS C:\Users\noush\OneDrive\Desktop\2403A52412ai>

powerShell  
powerShell

#### OBSERVATION:

1. The code first demonstrates a ZeroDivisionError when dividing by zero inside a loop.
2. It then shows how to handle the error using try-except, allowing the loop to continue safely without crashing.

#### Task Description #5:

- Include a buggy class definition with incorrect `__init__` parameters or attribute references. Ask AI to analyze and correct the constructor and attribute usage.

##### Incorrect Code:

```
class StudentRecord:
    def __init__(self, name, id, courses=[]):
        self.studentName = names
        self.student_id = id
        self.courses = courseList

    def add_course(self, course):
        self.courses.append(course)
```

```

def get_summary(self):
    return f"Student: {self.studentName}, ID: {self.student_id}, Courses: {' '.join(self.courses)}"

class Department:
    def __init__(self, deptName, students=None):
        self.dept_name = deptName
        self.students = students

    def enroll_student(self, student):
        self.students.append(student)

    def department_summary(self):
        return f"Department: {self.dept_name}, Total Students: {len(self.student)}"

s1 = StudentRecord("Alice", 101, ["Math", "Science"])
d1 = Department("Computer Science")
d1.enroll_student(s1)
print(s1.get_summary())
print(d1.department_summary())

```

### Prompt:

Write a buggy Python class with incorrect **init** parameters and attribute references, then correct the constructor and attribute usage. Add code to create an object and display its information. Include a print statement to verify output.

### CODE:

```

print("Test")
# Buggy class definition
class Student:
    def __init__(self, namee, rollnumber): # Typo in parameter 'namee'
        self.nam = name # Typo in attribute 'nam'
        self.roll = rollnum # Typo in variable 'rollnum'

    def display(self):
        print(f"Name: {self.name}, Roll Number: {self.rollnumber}") # AttributeError
# Corrected class definition
class StudentFixed:
    def __init__(self, name, rollnumber):
        self.name = name
        self.rollnumber = rollnumber

    def display(self):
        print(f"Name: {self.name}, Roll Number: {self.rollnumber}")
s = StudentFixed("Bob", 101)
s.display()

```

## CODE AND OUTPUT:

```
ass.py
1 print("Test")
2 # Buggy class definition
3 class Student:
4     def __init__(self, namee, rollnumber): # Typo in parameter 'namee'
5         self.nam = name # Typo in attribute 'nam'
6         self.roll = rollnum # Typo in variable 'rollnum'
7
8     def display(self):
9         print(f"Name: {self.name}, Roll Number: {self.rollnumber}")
10
11 # Corrected class definition
12 class StudentFixed:
13     def __init__(self, name, rollnumber):
14         self.name = name
15         self.rollnumber = rollnumber
16
17     def display(self):
18         print(f"Name: {self.name}, Roll Number: {self.rollnumber}")
```

h/OneDrive/Desktop/2403A52412ai/ass.py  
PS C:\Users\noush\OneDrive\Desktop\2403A52412ai> & C:\Users\noush\AppData\Local\Programs\Python\Python313\python.exe c:/Users/noush/OneDrive/Desktop/2403A52412ai/ass.py  
PS C:\Users\noush\OneDrive\Desktop\2403A52412ai> & C:\Users\noush\AppData\Local\Programs\Python\Python313\python.exe c:/Users/noush/OneDrive/Desktop/2403A52412ai/ass.py  
Test  
Name: Bob, Roll Number: 101  
PS C:\Users\noush\OneDrive\Desktop\2403A52412ai>

## OBSERVATION:

1. The code demonstrates both a buggy and a corrected class definition, highlighting common mistakes in constructor and attribute naming.
2. The corrected class is instantiated and its method is called, and a print statement at the top confirms that output is working.

