LAB-7.4

MD ZIAUDDIN
2403A51271

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.

```
Users > Rishitha Reddy > OneDrive > Desktop > AIAC > Lab-7 > Task1.py > ...
def factr(n):
    # Convert string to integer if needed
    if isinstance(n, str):
        n = int(n)

    if n == 0:
        return 1  # Factorial of 0 is 1
    elif n == 1:
        return 1
    else:
        return n * factr(n - 1)  # Should be n-1, not n-2

print(factr("5"))
```

OUTPUT:

```
neDrive/Desktop/AIAC/Lab-7/Task1.py"
120
```

```
def factr(n):
    # Convert string to integer if needed
    if isinstance(n, str):
        n = int(n)

    if n == 0:
        return 1  # Factorial of 0 is 1
    elif n == 1:
        return 1
    else:
        return n * factr(n - 1)  # Should be n-1, not n-2

print(factr("5"))
```

OUTPUT:

```
sers/Rishitha Reddy/OneDrive/Desktop/AIAC/Lab-7/task1.1.py
120
PS C:\Users\Rishitha Reddy\OneDrive\Desktop\web folder>
```

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.

```python
def sort_list(data):
    # Separate numbers and strings
    numbers = [x for x in data if isinstance(x, (int, float))]
    strings = [x for x in data if isinstance(x, str)]

    # Sort each type separately
    numbers.sort()
    strings.sort()

    # Return combined sorted list (numbers first, then strings)
    return numbers + strings

items = [3, "apples", 1, "banana", 2]
print(sort_list(items))
```

OUTPUT:

```
neDrive/Desktop/AIAC/Lab-7/task2.py
[1, 2, 3, 'apples', 'banana']
PS C:\Users\Rishitha Reddy\OneDrive\Desktop\AIAC\Lab-4>
```

```python
def sort_list(data):
    # Separate numbers and strings
    numbers = [x for x in data if isinstance(x, (int, float))]
    strings = [x for x in data if isinstance(x, str)]

    # Sort each type separately
    numbers.sort()
    strings.sort()

    # Return combined sorted list (numbers first, then strings)
    return numbers + strings

items = [3, "apples", 1, "banana", 2]
print(sort_list(items))
```

OUTPUT:

```
PS C:\Users\Rishitha Reddy\OneDrive\Desktop\web folder> & "C:/Users/Rishitha Re
sers/Rishitha Reddy/OneDrive/Desktop/AIAC/Lab-7/Task2.1.py"
[1, 2, 3, 'apples', 'banana']
PS C:\Users\Rishitha Reddy\OneDrive\Desktop\web folder>
```

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).

```python
with open("example.txt","w") as f:
    f.write("Hello,world!")
    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")
    import os
    if not os.path.exists("input.txt"):
        with open("input.txt", "w") as temp_input:
            temp_input.write("Sample input line 1\nSample input line 2\n")
            data = open("input.txt", "r").readlines()
            output = open("output.txt", "w")
            for line in data:
                output.write(line.upper())
                print("Processing done")


    import os
    if not os.path.exists("numbers.txt"):
        with open("numbers.txt", "w") as nf:
            nf.write("1\n2\n3\n4\n5\n")  # Example numbers

    with open("numbers.txt", "r") as f:
        nums = f.readlines()
    squares = []
    for n in nums:
        n = n.strip()
        if n.isdigit():
            squares.append(int(n) * int(n))
    with open("squares.txt", "w") as f2:
        for sq in squares:
            f2.write(str(sq) + "\n")
    print("Squares written")
```

OUTPUT:

```
Files written successfully
Squares written
PS C:\Users\Rishitha Reddy\OneDrive\Desktop\AIAC\Lab-4>
```

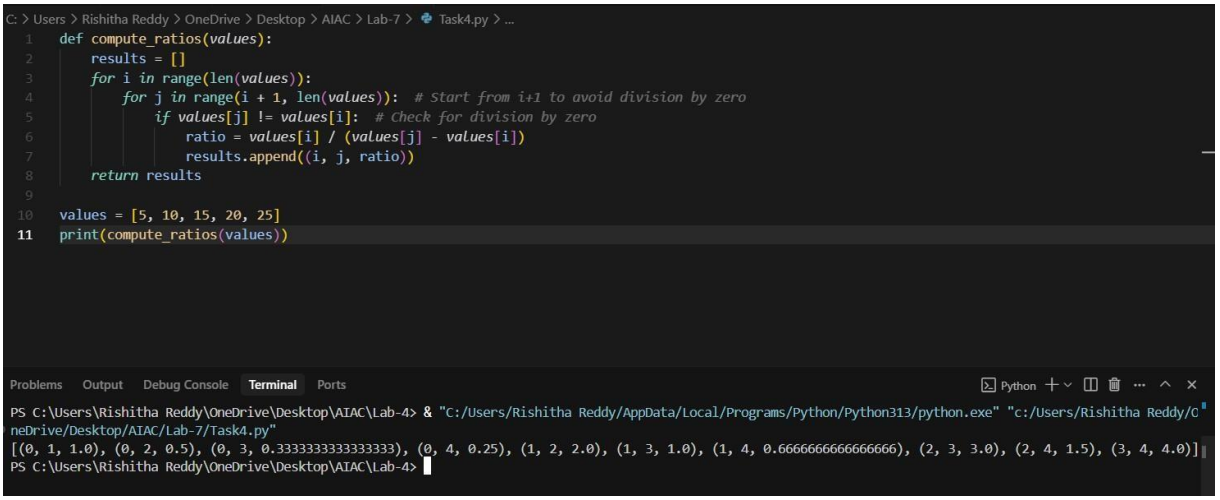
```python
with open("example.txt","w") as f:
    f.write("Hello,world!")
    f1=open("data1.txt","w")
    f2=open("data2.txt","w")
    f1.write("First file content\n")
    f2.write("Second file content\n")
    print("Files written successfully")
    import os
    if not os.path.exists("input.txt"):
        with open("input.txt", "w") as temp_input:
            temp_input.write("Sample input line 1\nSample input
            data = open("input.txt", "r").readlines()
            output = open("output.txt", "w")
            for line in data:
                output.write(line.upper())
                print("Processing done")


    import os
    if not os.path.exists("numbers.txt"):
        with open("numbers.txt", "w") as nf:
            nf.write("1\n2\n3\n4\n5\n")  # Example numbers

    with open("numbers.txt", "r") as f:
        nums = f.readlines()
    squares = []
    for n in nums:
        n = n.strip()
        if n.isdigit():
            squares.append(int(n) * int(n))
    with open("squares.txt", "w") as f2:
        for sq in squares:
            f2.write(str(sq) + "\n")
    print("Squares written")
```

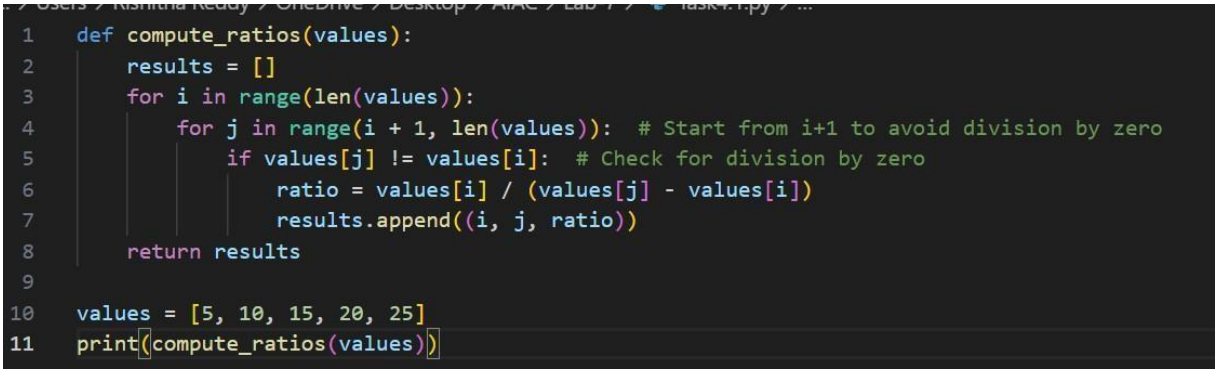
OUTPUT:

```
sers/Rishitha Reddy/OneDrive/Desktop/AIAC/Lab-7/task3.1.py"
Files written successfully
Squares written
PS C:\Users\Rishitha Reddy\OneDrive\Desktop\web folder>
```

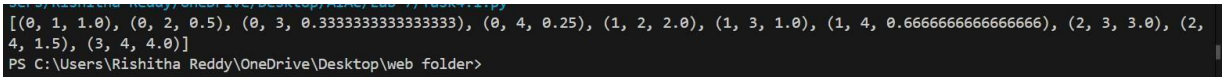
Task Description #4:

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





OUTPUT:



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

```python
class StudentRecord:
    def __init__(self, name, id, courses=None):
        if courses is None:
            courses = []
        self.student_name = name
        self.student_id = id
        self.courses = courses
    Ctrl+L to chat, Ctrl+K to generate
    def add_course(self, course):
        self.courses.append(course)

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


class Department:
    def __init__(self, dept_name, students=None):
        self.dept_name = dept_name
        if students is None:
            students = []
        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.students)}"


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

OUTPUT:

```
Student: Alice, ID: 101, Courses: Math, Science
Department: Computer Science, Total Students: 1
PS C:\Users\Rishitha Reddy\OneDrive\Desktop\AIAC\Lab-4>
```

```python
class StudentRecord:
    def __init__(self, name, id, courses=None):
        if courses is None:
            courses = []
        self.student_name = name
        self.student_id = id
        self.courses = courses

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

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


class Department:
    def __init__(self, dept_name, students=None):
        self.dept_name = dept_name
        if students is None:
            students = []
        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.students)}"


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

Ln 19, Col 29 Spaces: 4 UTF-8 CRLF {} Python

```
sers/Rishitha Reddy/OneDrive/Desktop/AIAC/Lab-7/Task5.1.py"
Student: Alice, ID: 101, Courses: Math, Science
Department: Computer Science, Total Students: 1
PS C:\Users\Rishitha Reddy\OneDrive\Desktop\web folder>
```