AI ASSISTED CODING

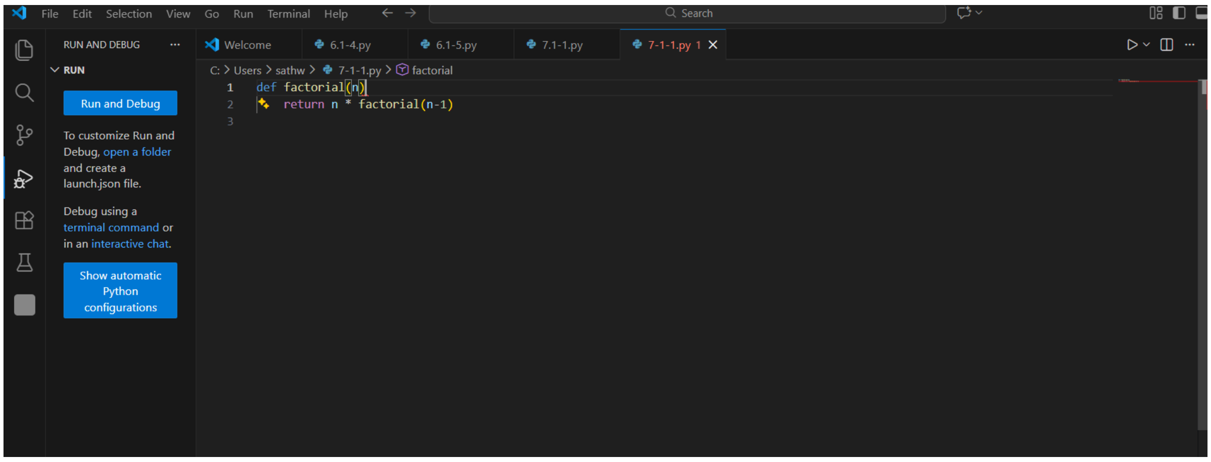
Assignment-7.1

HTNO:2303A510B1

Batch:14

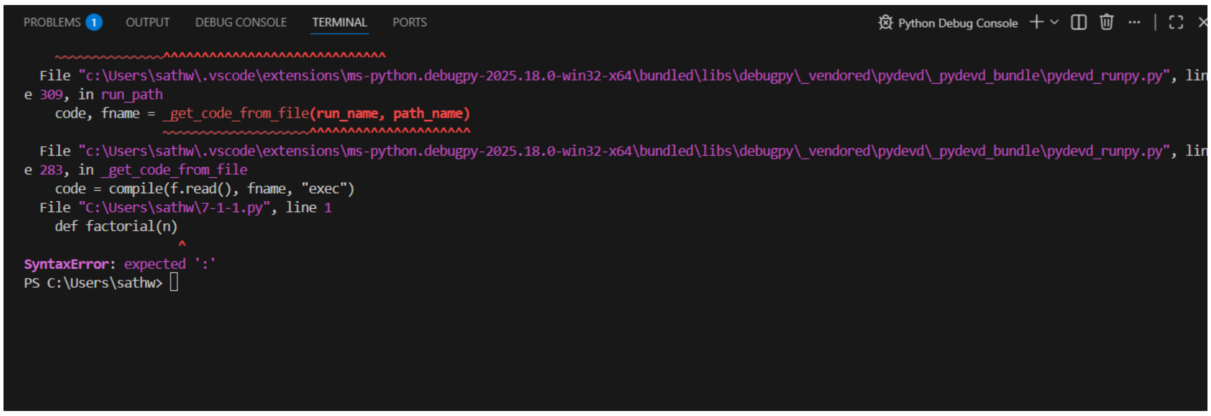
Name: B . Sathwik reddy

Task 1: Debugging a Recursive Calculation Module



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

Output:



```
    ~~~~~~~~~~~~~~^^^^^^^^^^^^^^^^^^^^^^^^^^^^^
  File "c:\Users\sathw\.vscode\extensions\ms-python.debugpy-2025.18.0-win32-x64\bundled\libs\debugpy\_vendored\pydevd\_pydevd_bundle\pydevd_runpy.py", lin
e 309, in run_path
    code, fname = _get_code_from_file(run_name, path_name)
                  ~~~~~~~~~~~~~~~~~~~^^^^^^^^^^^^^^^^^^^^
  File "c:\Users\sathw\.vscode\extensions\ms-python.debugpy-2025.18.0-win32-x64\bundled\libs\debugpy\_vendored\pydevd\_pydevd_bundle\pydevd_runpy.py", lin
e 283, in _get_code_from_file
    code = compile(f.read(), fname, "exec")
  File "C:\Users\sathw\7-1-1.py", line 1
    def factorial(n)
                    ^
SyntaxError: expected ':'
PS C:\Users\sathw>
```

After Debugging

I found the syntax error! You're missing a **colon (:)** at the end of the function definition line.

In Python, all function definitions must end with a colon to indicate the start of the function body. Here's the corrected code:

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

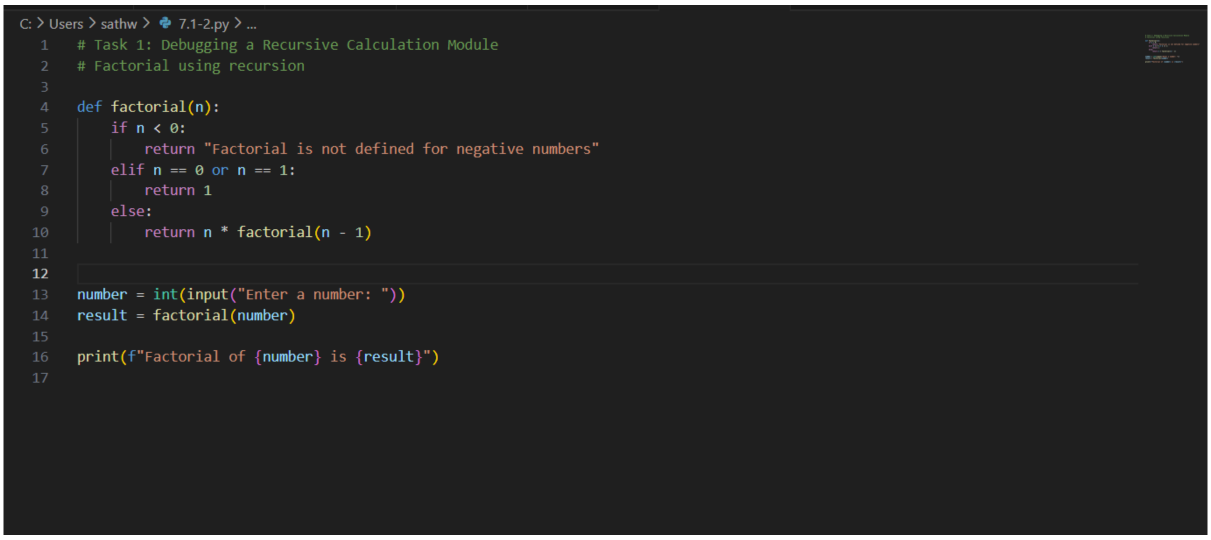
Key points:

- **The colon (:)** signals to Python that an indented block follows
- The indentation of the return statement shows it's part of the function body

Additional observation: This recursive function has an issue—it will cause infinite recursion because it never has a **base case** (a condition to stop recursing). You'd want to add a base case:

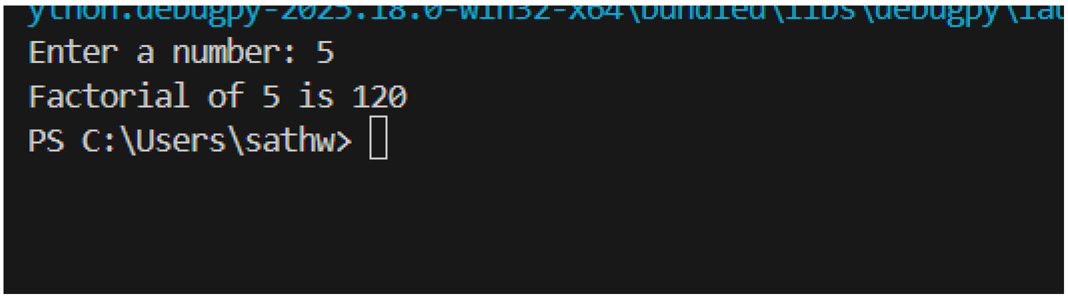
```
def factorial(n):
    if n <= 1:
        return 1
    return n * factorial(n-1)
```

This prevents the recursion from continuing indefinitely by returning 1 when n reaches 1 or less.



```python
# Task 1: Debugging a Recursive Calculation Module
# Factorial using recursion

def factorial(n):
    if n < 0:
        return "Factorial is not defined for negative numbers"
    elif n == 0 or n == 1:
        return 1
    else:
        return n * factorial(n - 1)


number = int(input("Enter a number: "))
result = factorial(number)

print(f"Factorial of {number} is {result}")
```

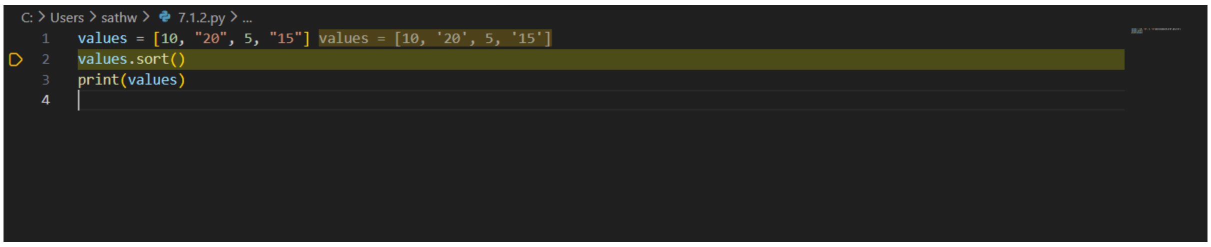
Output:



```
ython.debugpy-2025.18.0-win32-x64\bundled\libs\debugpy\la
Enter a number: 5
Factorial of 5 is 120
PS C:\Users\sathw>
```

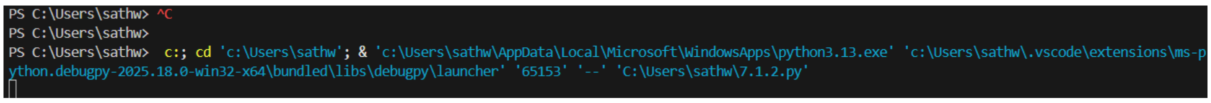
Task – 2:

Task 2: Fixing Data Type Errors in a Sorting Utility



```
C: > Users > sathw > 7.1.2.py > ...
values = [10, "20", 5, "15"] values = [10, '20', 5, '15']
values.sort()
print(values)

```

Output:



```
PS C:\Users\sathw> ^C
PS C:\Users\sathw>
PS C:\Users\sathw>  c:; cd 'c:\Users\sathw'; & 'c:\Users\sathw\AppData\Local\Microsoft\WindowsApps\python3.13.exe' 'c:\Users\sathw\.vscode\extensions\ms-p
ython.debugpy-2025.18.0-win32-x64\bundled\libs\debugpy\launcher' '65153' '--' 'C:\Users\sathw\7.1.2.py'
```

After Debugging:

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

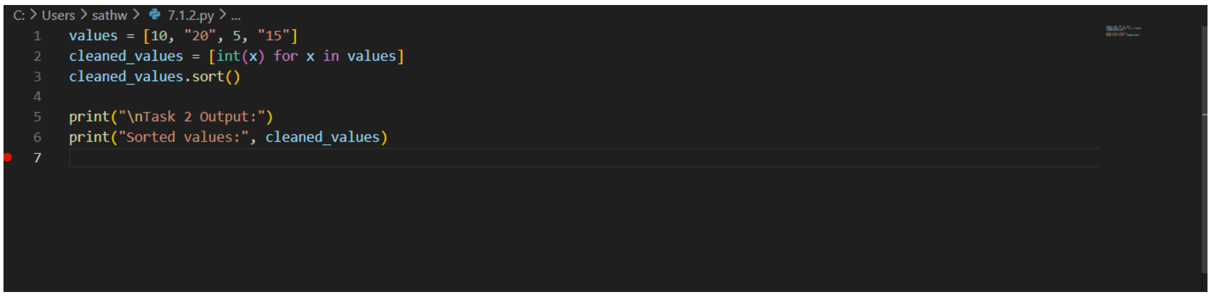
data = [10, "5", 3, "20", 1]

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.

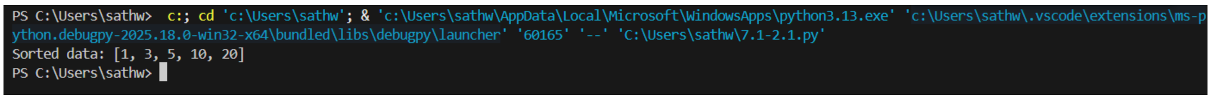
Why it fails:

- 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



```
C: > Users > sathw > 7.1.2.py > ...
values = [10, "20", 5, "15"]
cleaned_values = [int(x) for x in values]
cleaned_values.sort()

print("\nTask 2 Output:")
print("Sorted values:", cleaned_values)

```

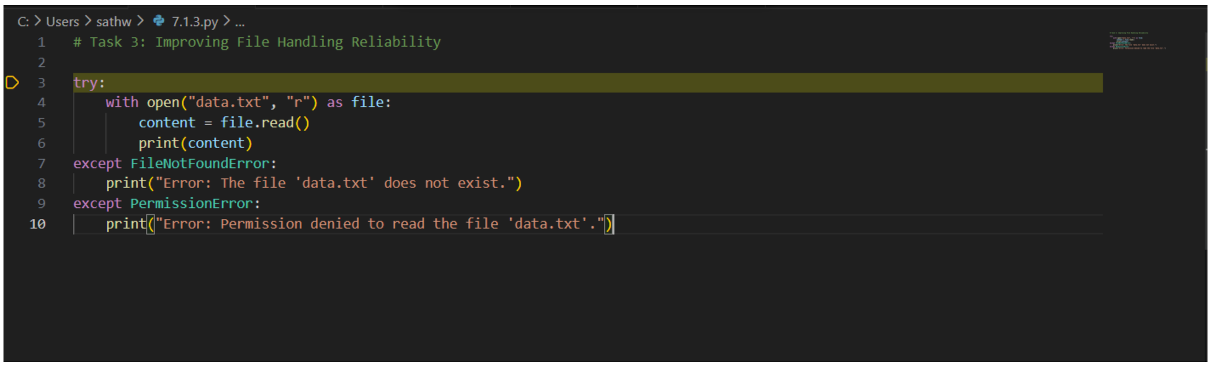
Output:



```
PS C:\Users\sathw>  c:; cd 'c:\Users\sathw'; & 'c:\Users\sathw\AppData\Local\Microsoft\WindowsApps\python3.13.exe' 'c:\Users\sathw\.vscode\extensions\ms-p
ython.debugpy-2025.18.0-win32-x64\bundled\libs\debugpy\launcher' '60165' '--' 'C:\Users\sathw\7.1-2.1.py'
Sorted data: [1, 3, 5, 10, 20]
PS C:\Users\sathw>
```

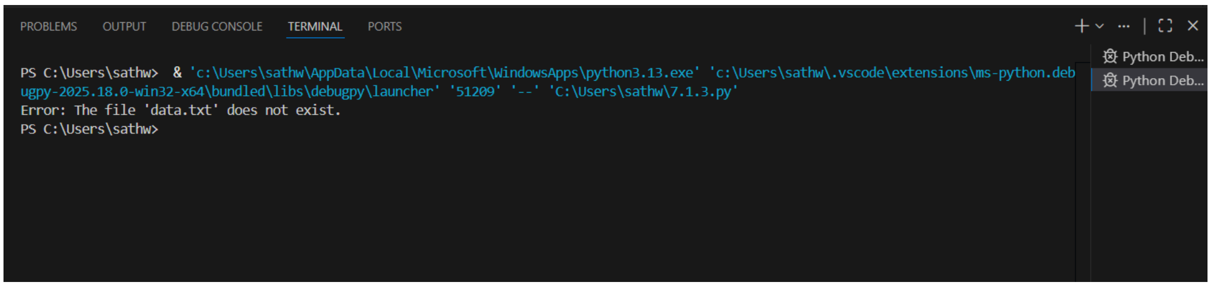
Task - 3:

Task 3: Improving File Handling Reliability


C: > Users > sathw > 7.1.3.py > ...

```python
# Task 3: Improving File Handling Reliability

try:
    with open("data.txt", "r") as file:
        content = file.read()
        print(content)
except FileNotFoundError:
    print("Error: The file 'data.txt' does not exist.")
except PermissionError:
    print("Error: Permission denied to read the file 'data.txt'.")
```

Output:



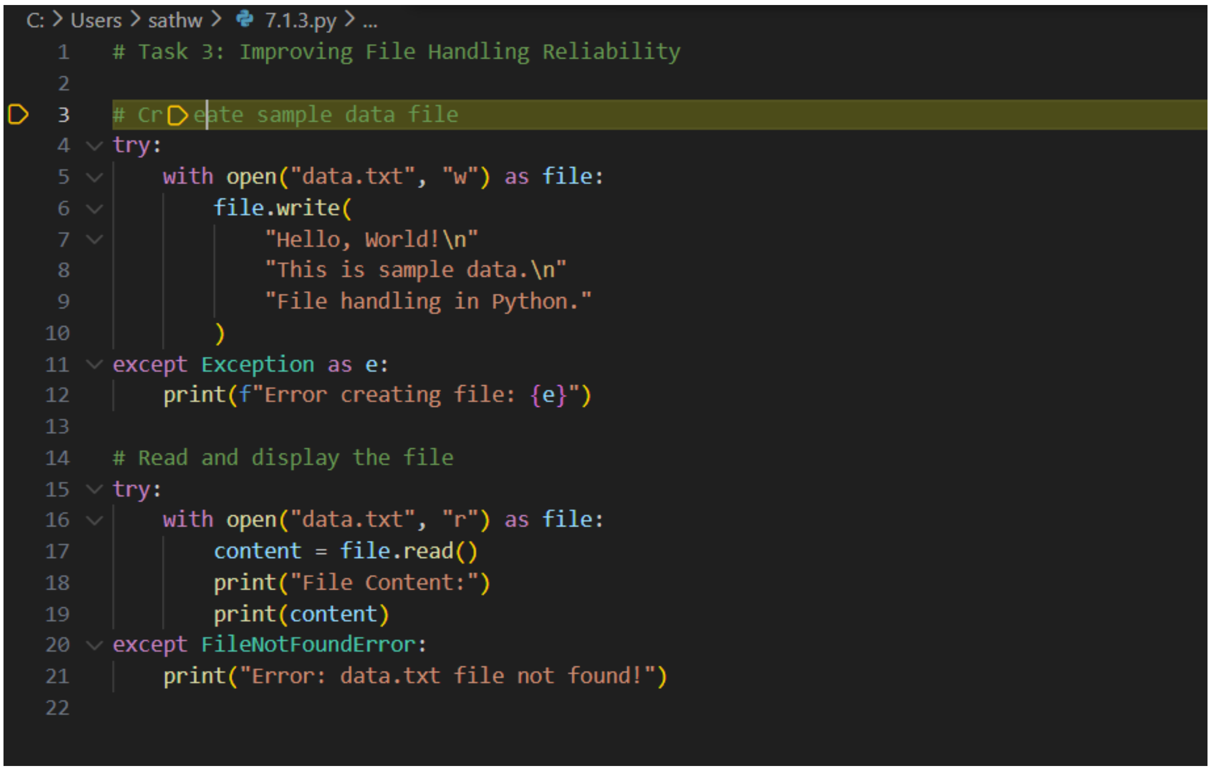
```
PS C:\Users\sathw>  & 'c:\Users\sathw\AppData\Local\Microsoft\WindowsApps\python3.13.exe' 'c:\Users\sathw\.vscode\extensions\ms-python.deb
ugpy-2025.18.0-win32-x64\bundled\libs\debugpy\launcher' '51209' '--' 'C:\Users\sathw\7.1.3.py'
Error: The file 'data.txt' does not exist.
PS C:\Users\sathw>
```

After Debugging:

What it does:

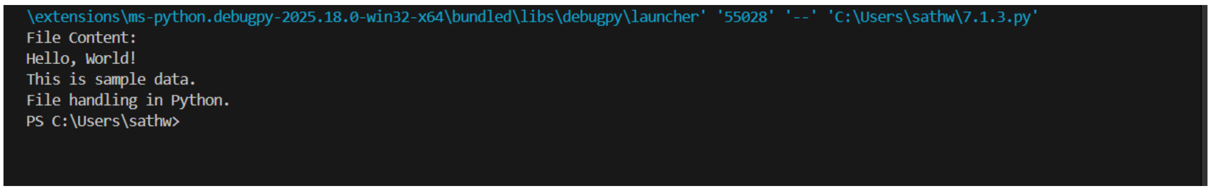
1. **open("data.txt", "r")** - Opens a file named data.txt in **read mode** ("r")
2. file.read() - Reads all the content from the file
3. print(content) - Displays the file content

Why it's failing:
The file data.txt doesn't exist in your working directory, so Python throws a FileNotFoundError.



```
C: > Users > sathw > 7.1.3.py > ...
# Task 3: Improving File Handling Reliability

# Create sample data file
try:
    with open("data.txt", "w") as file:
        file.write(
            "Hello, World!\n"
            "This is sample data.\n"
            "File 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!")
```

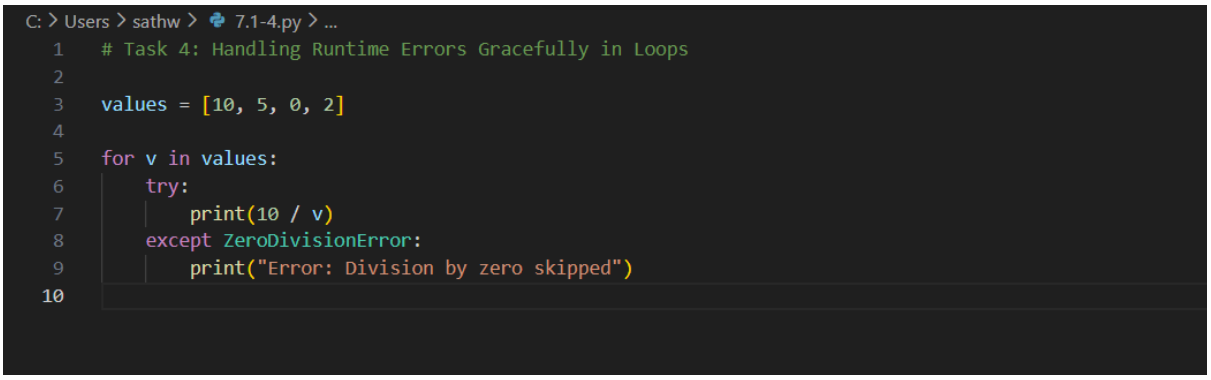
Output:



```
\extensions\ms-python.debugpy-2025.18.0-win32-x64\bundled\libs\debugpy\launcher' '55028' '--' 'C:\Users\sathw\7.1.3.py'
File Content:
Hello, World!
This is sample data.
File handling in Python.
PS C:\Users\sathw>
```

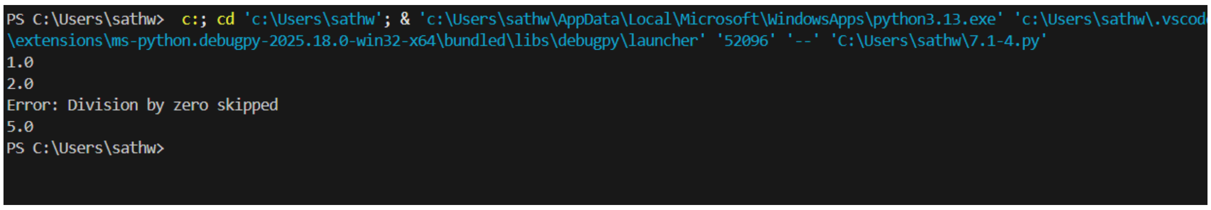
Task – 4:

Task 4: Handling Runtime Errors Gracefully in Loops



```
C: > Users > sathw > 7.1-4.py > ...
# Task 4: Handling Runtime Errors Gracefully in Loops

values = [10, 5, 0, 2]

for v in values:
    try:
        print(10 / v)
    except ZeroDivisionError:
        print("Error: Division by zero skipped")
```

Output :

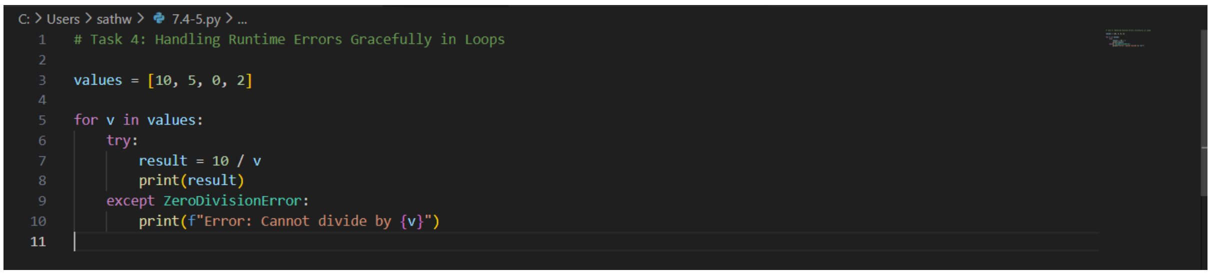

```
PS C:\Users\sathw>  c:; cd 'c:\Users\sathw'; & 'c:\Users\sathw\AppData\Local\Microsoft\WindowsApps\python3.13.exe' 'c:\Users\sathw\.vscod
\extensions\ms-python.debugpy-2025.18.0-win32-x64\bundled\libs\debugpy\launcher' '52096' '--' 'C:\Users\sathw\7.1-4.py'
1.0
2.0
Error: Division by zero skipped
5.0
PS C:\Users\sathw>
```

After Debugging:

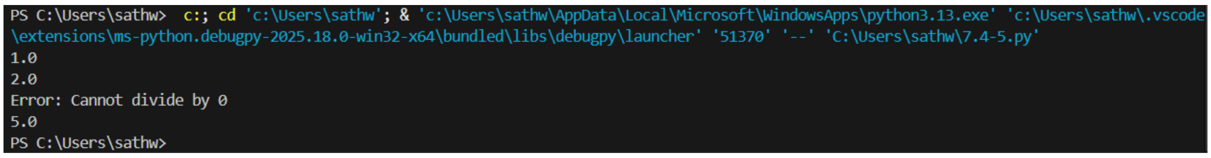
What it does:

- Loops through each value in the list: 10, 5, 0, 2
- Divides 10 by each value and prints the result

Why it's failing:
When v = 0, Python tries to execute 10 / 0, which causes a **ZeroDivisionError** (you cannot divide by zero).

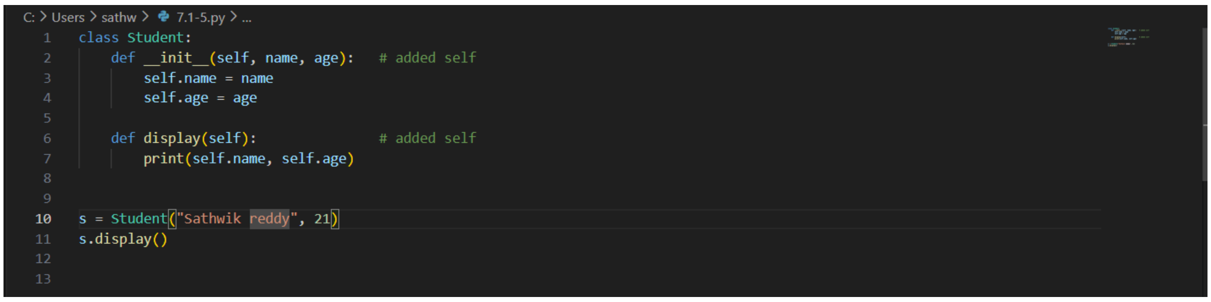

C: > Users > sathw > 7.4-5.py > ...
```python
# Task 4: Handling Runtime Errors Gracefully in Loops

values = [10, 5, 0, 2]

for v in values:
    try:
        result = 10 / v
        print(result)
    except ZeroDivisionError:
        print(f"Error: Cannot divide by {v}")
```

Output:

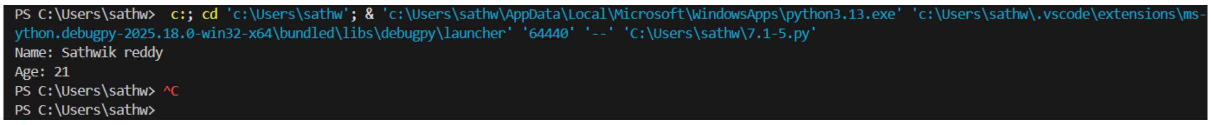

```
PS C:\Users\sathw>  c:; cd 'c:\Users\sathw'; & 'c:\Users\sathw\AppData\Local\Microsoft\WindowsApps\python3.13.exe' 'c:\Users\sathw\.vscode
\extensions\ms-python.debugpy-2025.18.0-win32-x64\bundled\libs\debugpy\launcher' '51370' '--' 'C:\Users\sathw\7.4-5.py'
1.0
2.0
Error: Cannot divide by 0
5.0
PS C:\Users\sathw>
```

Task -5:

Task 5: Debugging Class Initialization Errors


C: > Users > sathw > 7.1-5.py > ...
```python
class Student:
    def __init__(self, name, age):   # added self
        self.name = name
        self.age = age

    def display(self):               # added self
        print(self.name, self.age)


s = Student("Sathwik reddy", 21)
s.display()
```

Output :



```
PS C:\Users\sathw>  c:; cd 'c:\Users\sathw'; & 'c:\Users\sathw\AppData\Local\Microsoft\WindowsApps\python3.13.exe' 'c:\Users\sathw\.vscode\extensions\ms-python.debugpy-2025.18.0-win32-x64\bundled\libs\debugpy\launcher' '64440' '--' 'C:\Users\sathw\7.1-5.py'
Name: Sathwik reddy
Age: 21
PS C:\Users\sathw> ^C
PS C:\Users\sathw>
```

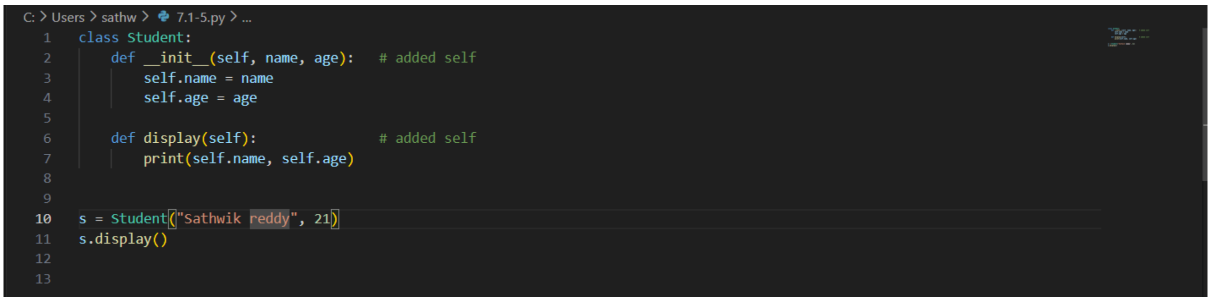
After debugging:

What it's trying to do:

- Create a Student class with name and age attributes
- Initialize a student object and display their details

Why it's failing:

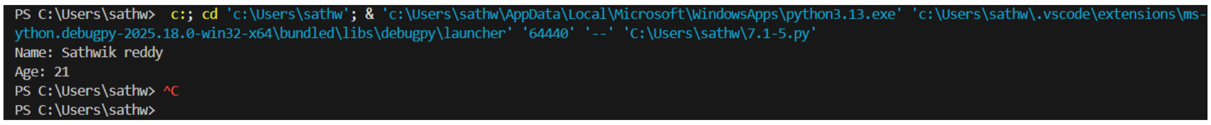
1. __init__ **method:** Missing self parameter
   - Should be: def __init__(self, name, age):
2. display() **method:** Missing self parameter AND references to variables
   - Should be: def display(self):
   - Should reference: self.name and self.age



C: > Users > sathw > 7.1-5.py > ...

```python
class Student:
    def __init__(self, name, age):   # added self
        self.name = name
        self.age = age

    def display(self):               # added self
        print(self.name, self.age)


s = Student("Sathwik reddy", 21)
s.display()
```

Output:



```
PS C:\Users\sathw>  c:; cd 'c:\Users\sathw'; & 'c:\Users\sathw\AppData\Local\Microsoft\WindowsApps\python3.13.exe' 'c:\Users\sathw\.vscode\extensions\ms-python.debugpy-2025.18.0-win32-x64\bundled\libs\debugpy\launcher' '64440' '--' 'C:\Users\sathw\7.1-5.py'
Name: Sathwik reddy
Age: 21
PS C:\Users\sathw> ^C
PS C:\Users\sathw>
```