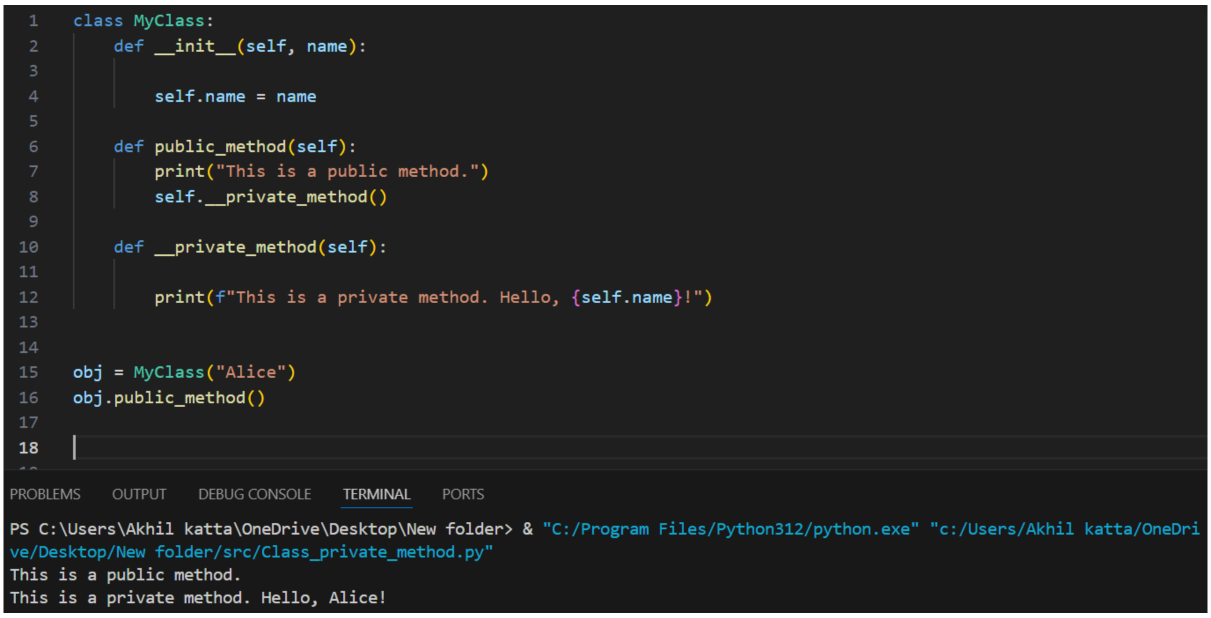
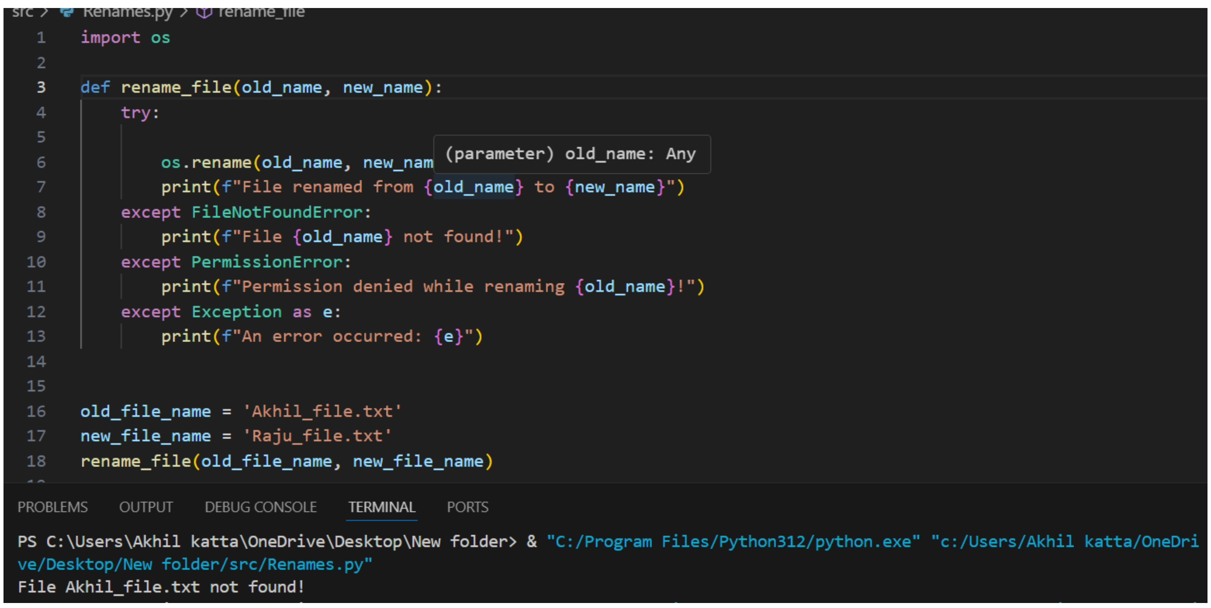
Python class with a private method



```python
class MyClass:
    def __init__(self, name):

        self.name = name

    def public_method(self):
        print("This is a public method.")
        self.__private_method()

    def __private_method(self):

        print(f"This is a private method. Hello, {self.name}!")


obj = MyClass("Alice")
obj.public_method()

```

```
PS C:\Users\Akhil katta\OneDrive\Desktop\New folder> & "C:/Program Files/Python312/python.exe" "c:/Users/Akhil katta/OneDri
ve/Desktop/New folder/src/Class_private_method.py"
This is a public method.
This is a private method. Hello, Alice!
```

Program that renames a file

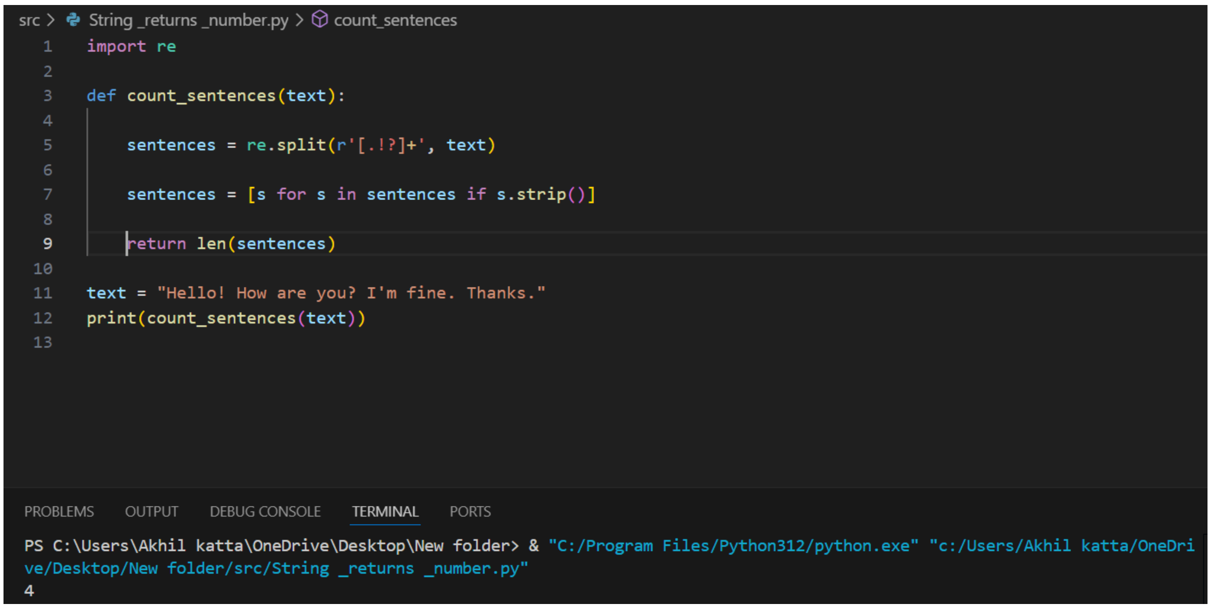


src > Renames.py > rename_file

```python
import os

def rename_file(old_name, new_name):
    try:

        os.rename(old_name, new_nam
        print(f"File renamed from {old_name} to {new_name}")
    except FileNotFoundError:
        print(f"File {old_name} not found!")
    except PermissionError:
        print(f"Permission denied while renaming {old_name}!")
    except Exception as e:
        print(f"An error occurred: {e}")


old_file_name = 'Akhil_file.txt'
new_file_name = 'Raju_file.txt'
rename_file(old_file_name, new_file_name)
```

```
PS C:\Users\Akhil katta\OneDrive\Desktop\New folder> & "C:/Program Files/Python312/python.exe" "c:/Users/Akhil katta/OneDri
ve/Desktop/New folder/src/Renames.py"
File Akhil_file.txt not found!
```

String and returns the number of sentences

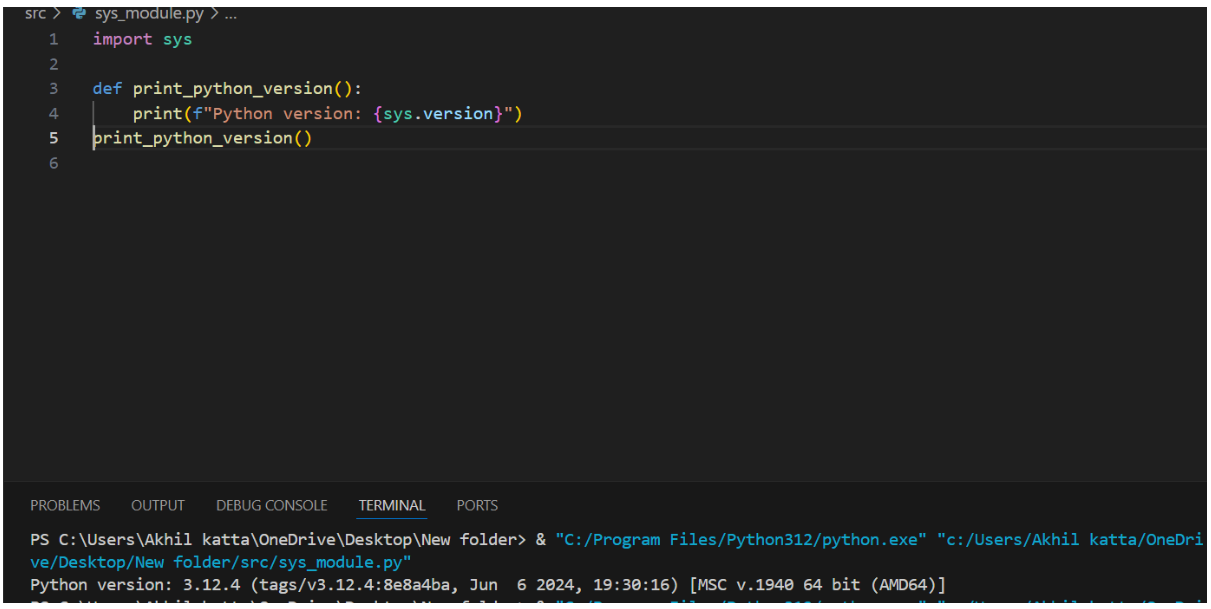


src > String _returns _number.py > count_sentences

```python
import re

def count_sentences(text):

    sentences = re.split(r'[.!?]+', text)

    sentences = [s for s in sentences if s.strip()]

    return len(sentences)

text = "Hello! How are you? I'm fine. Thanks."
print(count_sentences(text))
```

```
PS C:\Users\Akhil katta\OneDrive\Desktop\New folder> & "C:/Program Files/Python312/python.exe" "c:/Users/Akhil katta/OneDri
ve/Desktop/New folder/src/String _returns _number.py"
4
```

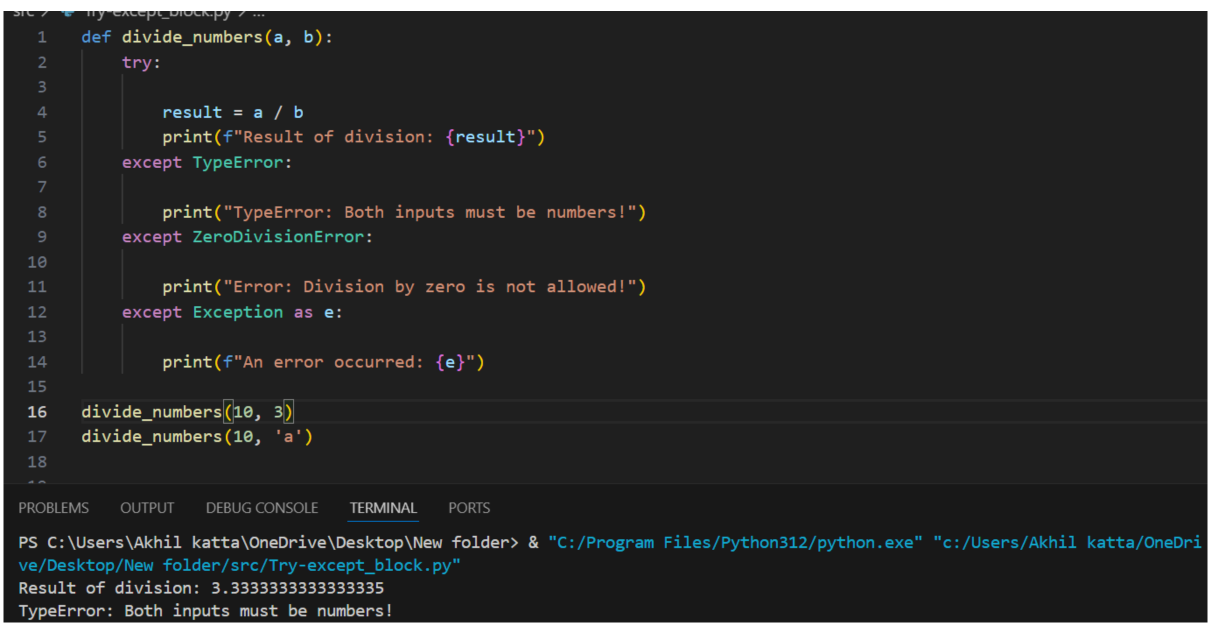
The sys module and prints the Python version



```python
import sys

def print_python_version():
    print(f"Python version: {sys.version}")
print_python_version()

```

```
PS C:\Users\Akhil katta\OneDrive\Desktop\New folder> & "C:/Program Files/Python312/python.exe" "c:/Users/Akhil katta/OneDri
ve/Desktop/New folder/src/sys_module.py"
Python version: 3.12.4 (tags/v3.12.4:8e8a4ba, Jun  6 2024, 19:30:16) [MSC v.1940 64 bit (AMD64)]
```

That Uses a try-except block to handle a Type Error



```python
def divide_numbers(a, b):
    try:

        result = a / b
        print(f"Result of division: {result}")
    except TypeError:

        print("TypeError: Both inputs must be numbers!")
    except ZeroDivisionError:

        print("Error: Division by zero is not allowed!")
    except Exception as e:

        print(f"An error occurred: {e}")

divide_numbers(10, 3)
divide_numbers(10, 'a')

```

```
PS C:\Users\Akhil katta\OneDrive\Desktop\New folder> & "C:/Program Files/Python312/python.exe" "c:/Users/Akhil katta/OneDri
ve/Desktop/New folder/src/Try-except_block.py"
Result of division: 3.3333333333333335
TypeError: Both inputs must be numbers!
```