Hands-on_TryException

Exercise 1: Handle division by zero

➢ Ask the user to enter two numbers and divide them

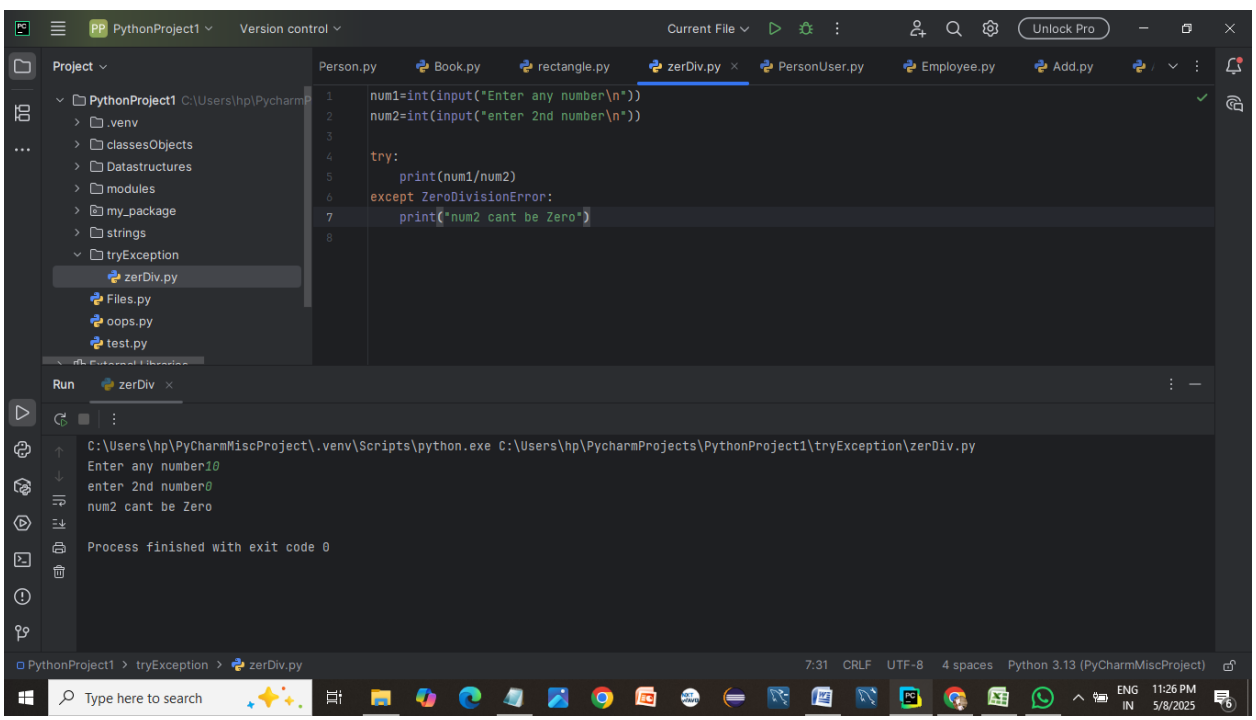
➢ Use try-except to handle ZeroDivisionError

Code:

```
num1=int(input("Enter any number\n"))
num2=int(input("enter 2nd number\n"))

try:
    print(num1/num2)
except ZeroDivisionError:
    print("num2 cant be Zero")
```

Output



Exercise 2: Handle invalid input

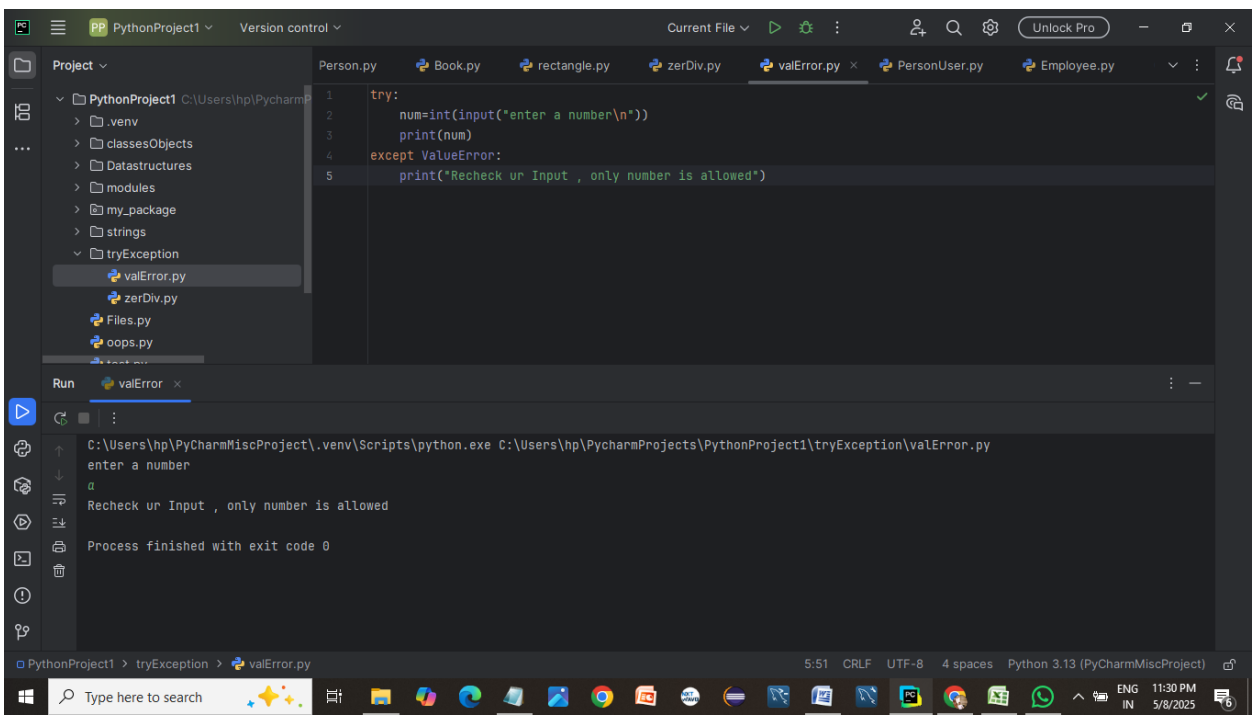
➢ Ask the user to enter a number and convert it to an integer

➢ Catch ValueError if input is not a number

Code)

```
try:
    num=int(input("enter a number\n"))
    print(num)
except ValueError:
    print("Recheck ur Input , only number is allowed")
```

Output



Exercise 3: Ask the user for a file name and try to open it

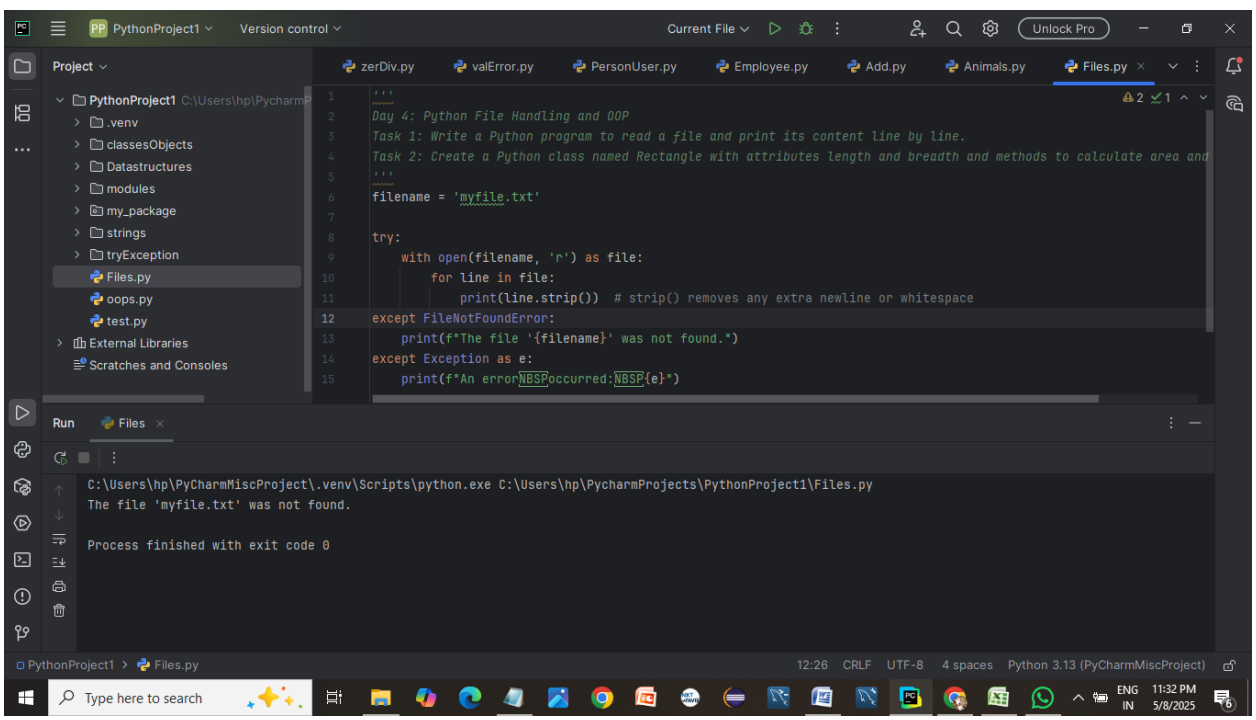
➢ Handle FileNotFoundError and general Exception

Code)

```
filename = 'myfile.txt'

try:
    with open(filename, 'r') as file:
        for line in file:
            print(line.strip())  # strip() removes any extra newline or
whitespace
except FileNotFoundError:
    print(f"The file '{filename}' was not found.")
except Exception as e:
    print(f"An error occurred: {e}")
```

Output)



Exercise 4: Combine multiple except blocks for different errors

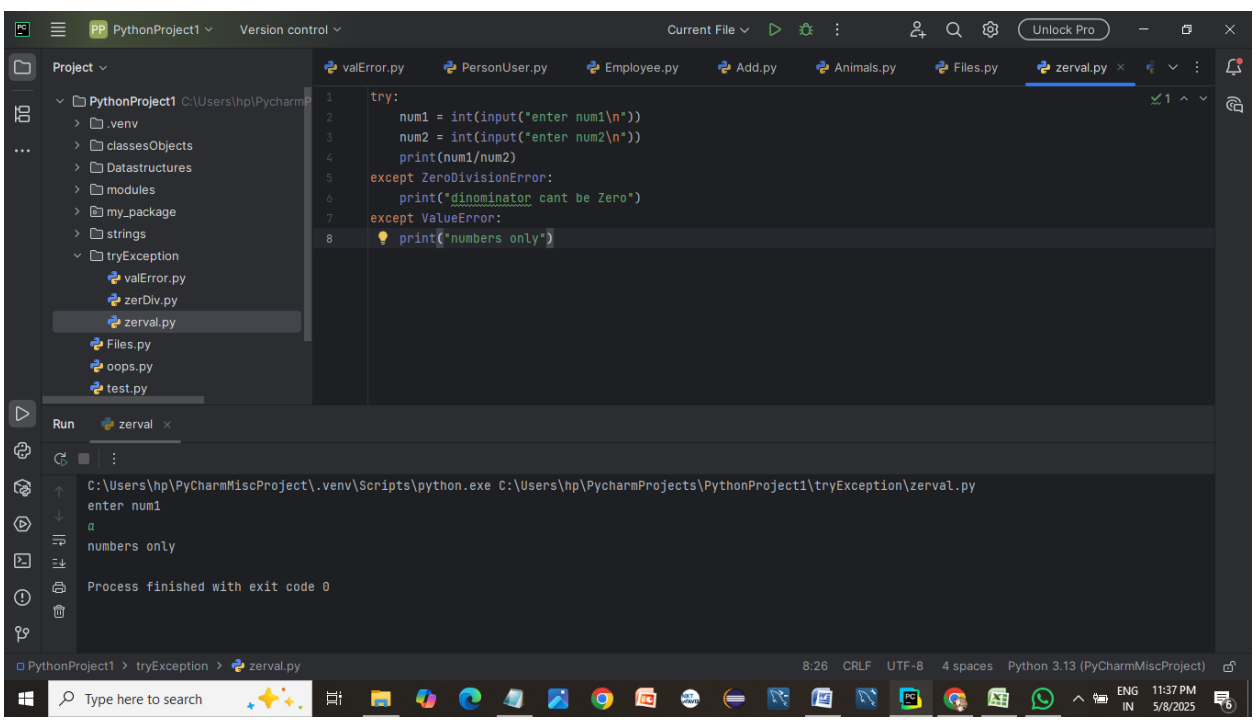
➢ Try converting input to float and dividing

➢ Handle ValueError and ZeroDivisionError separately

Code)

```
try:
    num1 = int(input("enter num1\n"))
    num2 = int(input("enter num2\n"))
    print(num1/num2)
except ZeroDivisionError:
    print("dinominator cant be Zero")
except ValueError:
    print("numbers only")
```

output)



Exercise 5: Use try-except-else-finally

➢ Prompt for a number and divide it by 2

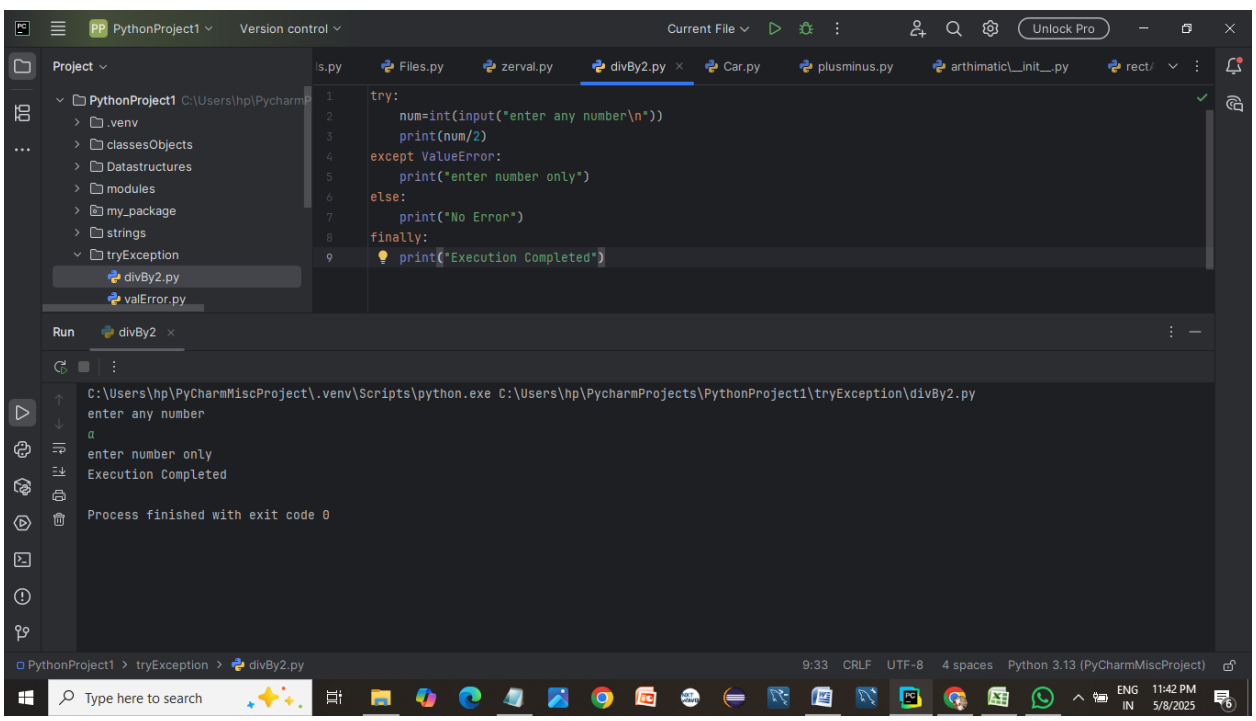
➢ Use else to confirm no error occurred

➢ Use finally to print "Execution complete"

Code)

```
try:
    num=int(input("enter any number\n"))
    print(num/2)
except ValueError:
    print("enter number only")
else:
    print("No Error")
finally:
    print("Execution Completed")
```

output)



Exercise 6: Create a custom exception called `NegativeNumberError`

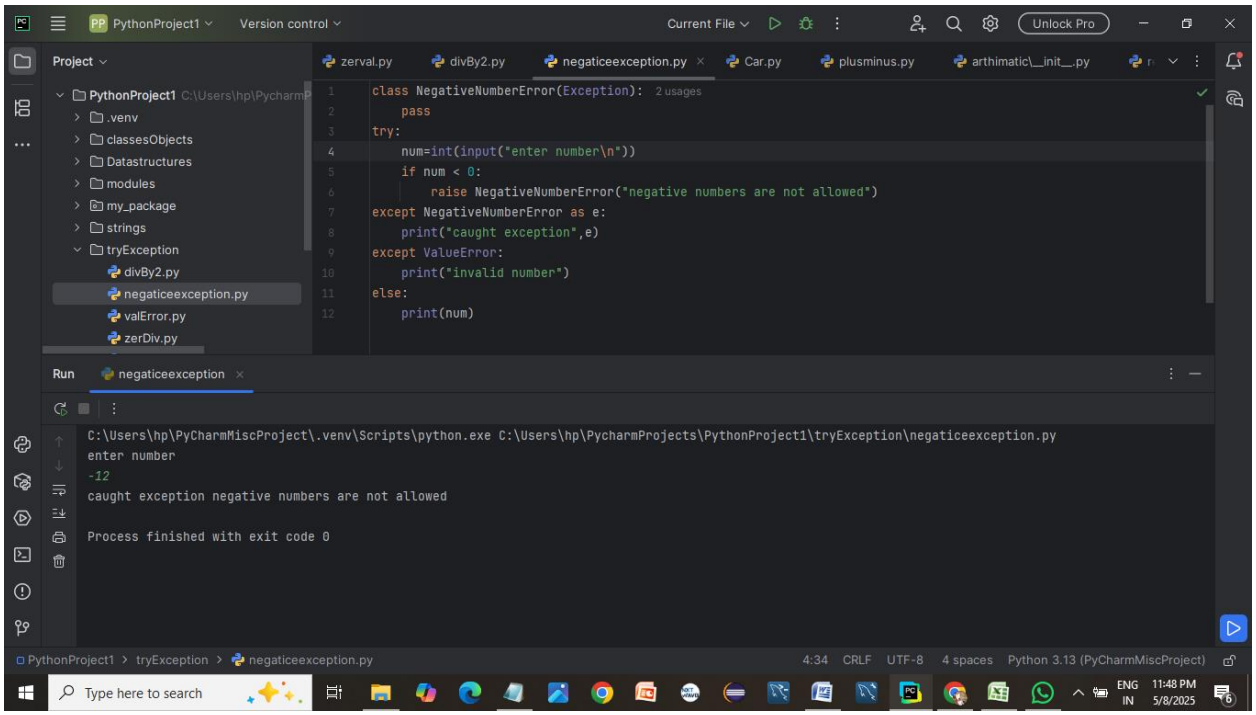
➢ Raise it if the user inputs a negative number

➢ Handle it in a try-except block

Code)

```
class NegativeNumberError(Exception):
    pass
try:
    num=int(input("enter number"))
    if num < 0:
        raise NegativeNumberError("negative numbers are not allowed")
except NegativeNumberError as e:
    print("caught exception",e)
except ValueError:
    print("invalid number")
else:
    print(num)
```

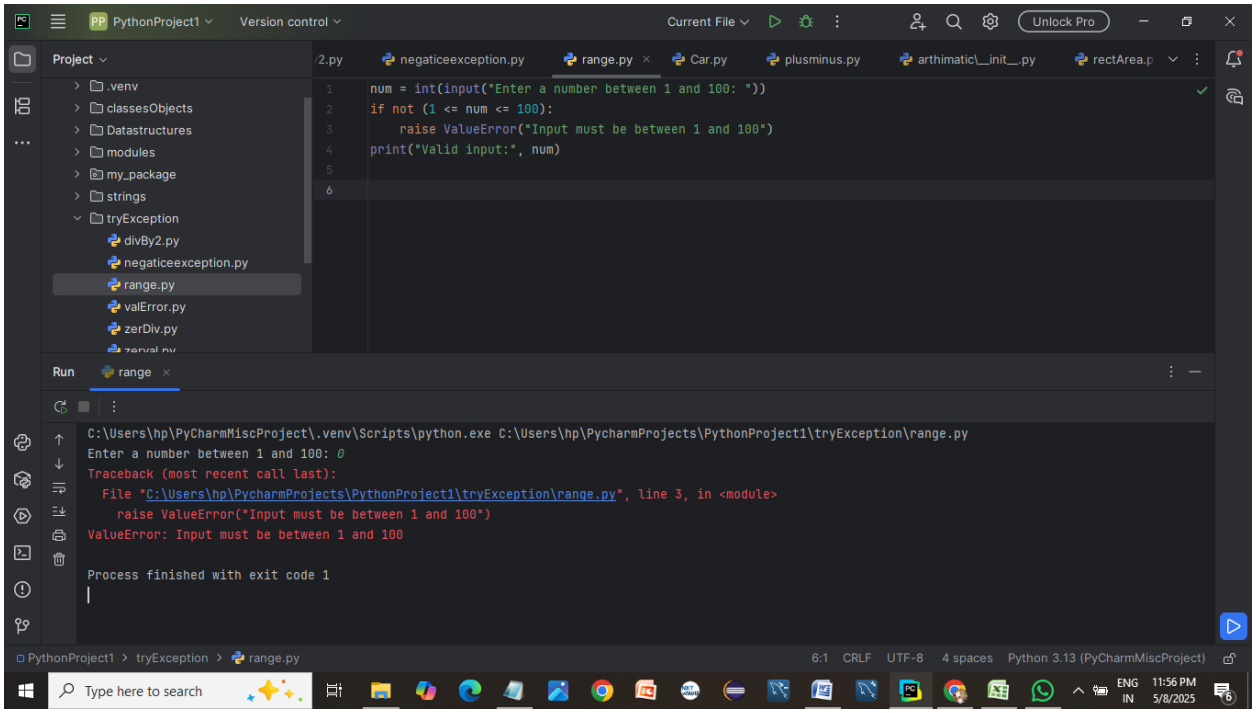
Output)



Exercise 7: Write a function that raises a `ValueError` if input is not between 1 and 100

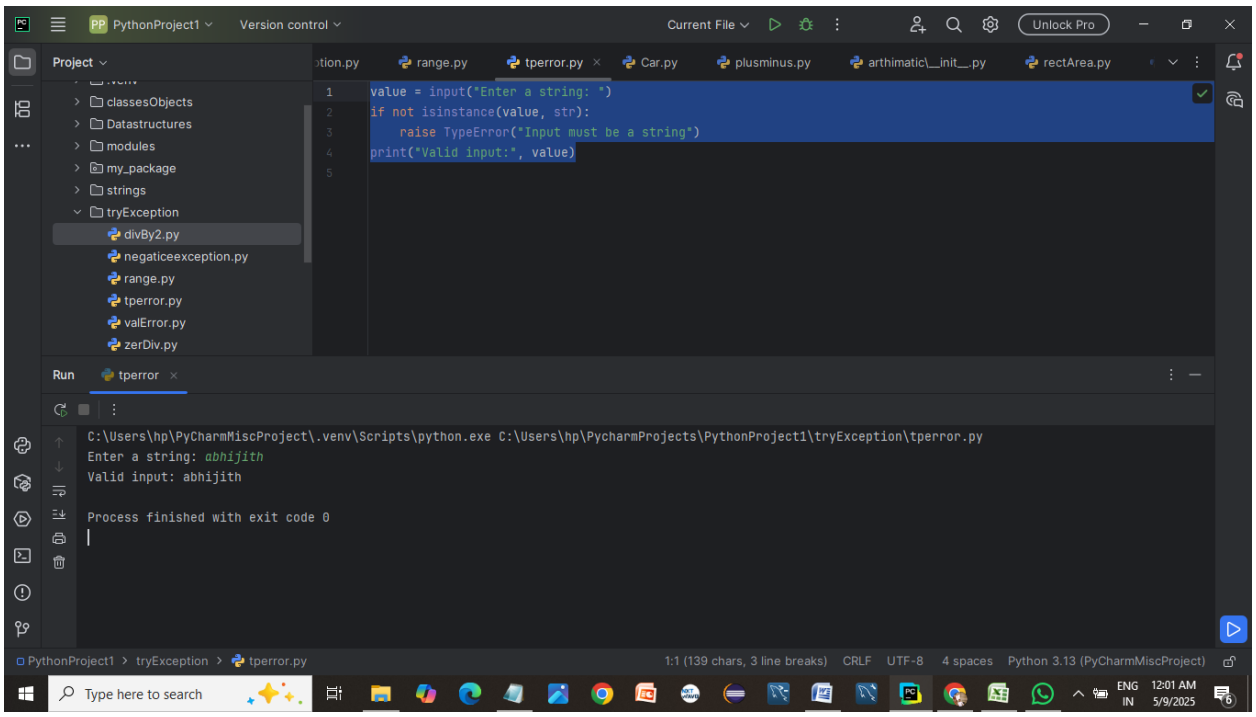
Exercise 8: Write a function that raises a `TypeError` if input is not a string

```
num = int(input("Enter a number between 1 and 100: "))
if not (1 <= num <= 100):
    raise ValueError("Input must be between 1 and 100")
print("Valid input:", num)
```



```python
value = input("Enter a string: ")
if not isinstance(value, str):
    raise TypeError("Input must be a string")
print("Valid input:", value)
```

Output)



Exercise 9: Build a simple calculator with exception handling

➢ Handle invalid inputs, division by zero, and unsupported operations

Code)

```
try:
    num1=int(input("Enter 1st number\n"))
    num2=int(input("Enter 2nd number\n"))
    oper=input("Select any operand from list \n+\n-\n*\n/\n%\n")
    if(oper == "+"):
        print("Sum of {} and {} is {}".format(num1,num2,num1+num2))
    elif(oper == "-"):
        print("Differencr of {} and {} is {}".format(num1,num2,num1-num2))
    elif(oper == "*"):
        print("Product of {} and {} is {}".format(num1, num2, num1 * num2))
    elif(oper == "/"):
        print("division of {} and {} is {}".format(num1, num2, num1 / num2))
    elif (oper == "%"):
        print("mod of {} and {} is {}".format(num1, num2, num1 % num2))
    else:
        print("Enter proper operator")
except ZeroDivisionError:
    print("denominator cant be 0")
except ValueError:
    print("invalid literal for int")
```

Output)