

Hands-on TryException

Exercise 1: Handle division by zero

➤ Ask the user to enter two numbers and divide them

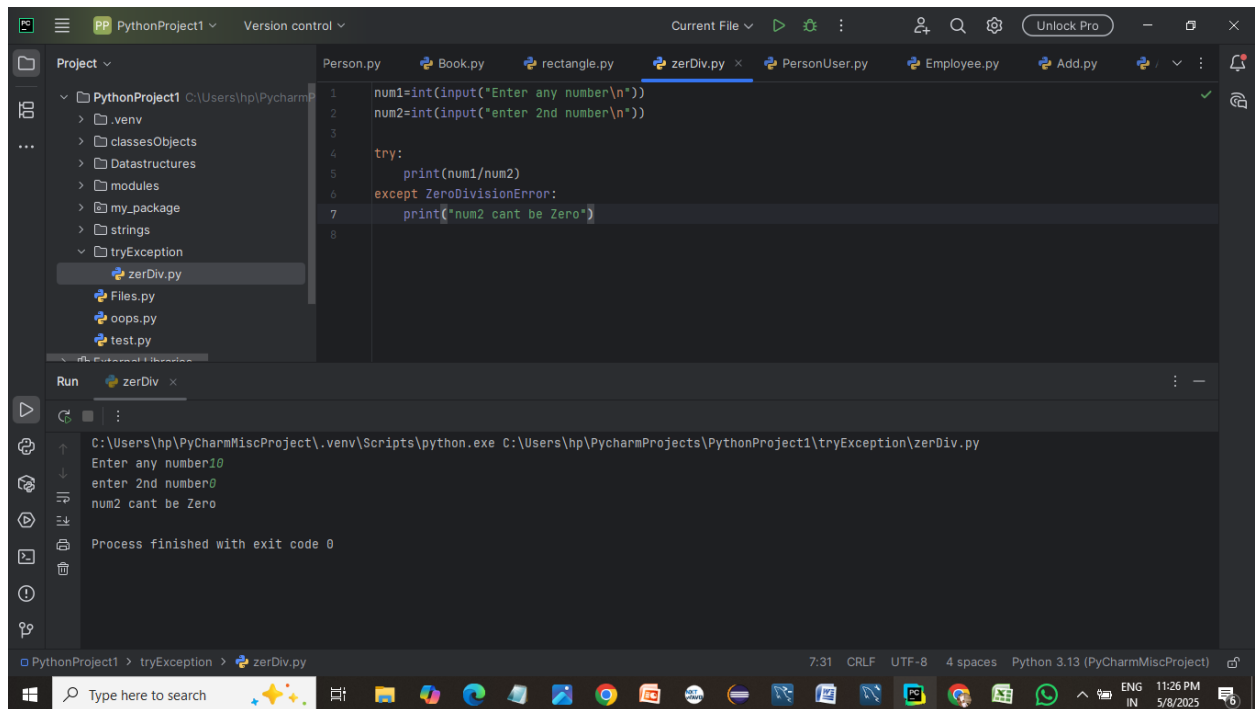
➤ Use try-except to handle ZeroDivisionError

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



The screenshot shows the PyCharm IDE interface. The top toolbar includes icons for file operations, search, and settings, along with a 'Unlock Pro' button. The left sidebar displays the project structure for 'PythonProject1', with 'tryException/zerDiv.py' selected. The main editor window shows the Python code for 'zerDiv.py' with line numbers 1 through 8. The code is as follows:

```
1 num1=int(input("Enter any number\n"))
2 num2=int(input("enter 2nd number\n"))
3
4 try:
5     print(num1/num2)
6 except ZeroDivisionError:
7     print("num2 cant be Zero")
8
```

Below the editor is the 'Run' console. It shows the command used to execute the script: `C:\Users\hp\PyCharmMiscProject\.venv\Scripts\python.exe C:\Users\hp\PycharmProjects\PythonProject1\tryException\zerDiv.py`. The output of the program is displayed as:

```
Enter any number10
enter 2nd number0
num2 cant be Zero
```

The console also indicates 'Process finished with exit code 0'. The bottom status bar shows the file encoding as UTF-8, 4 spaces indentation, and the Python version as 3.13 (PyCharmMiscProject). The Windows taskbar at the very bottom shows the time as 7:31 and the date as 5/8/2025.

Exercise 2: Handle invalid input

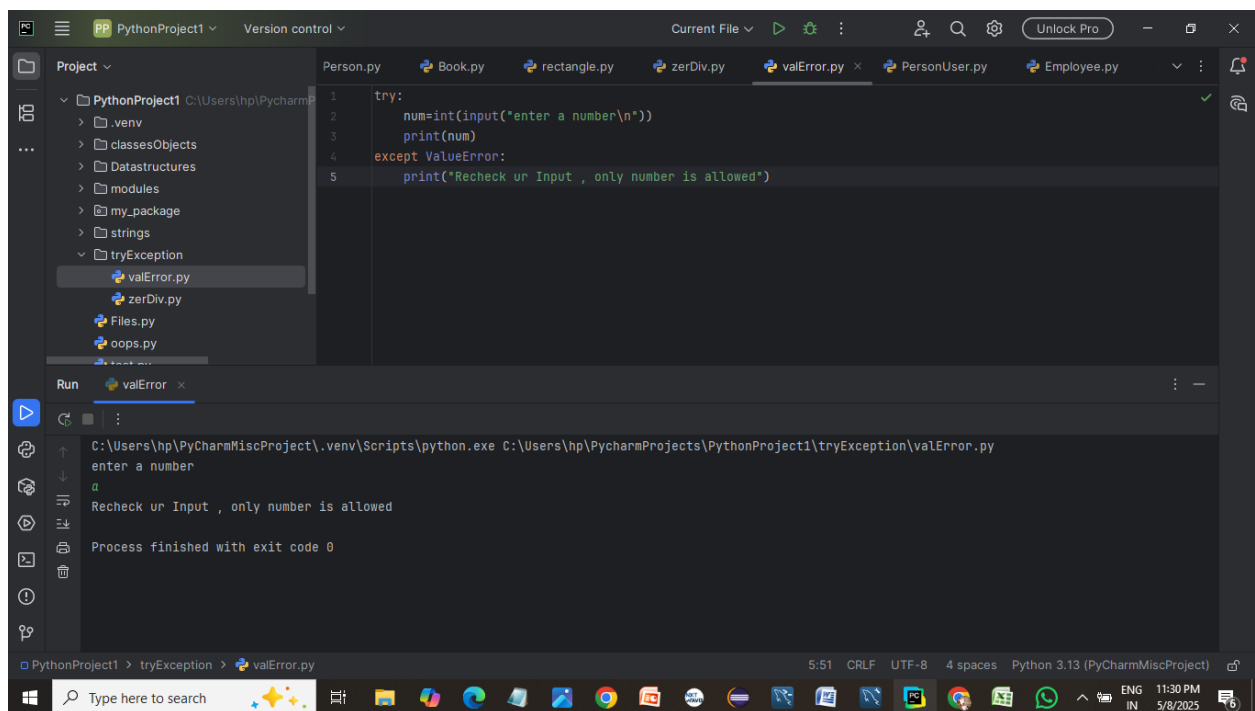
➤ Ask the user to enter a number and convert it to an integer

➤ 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

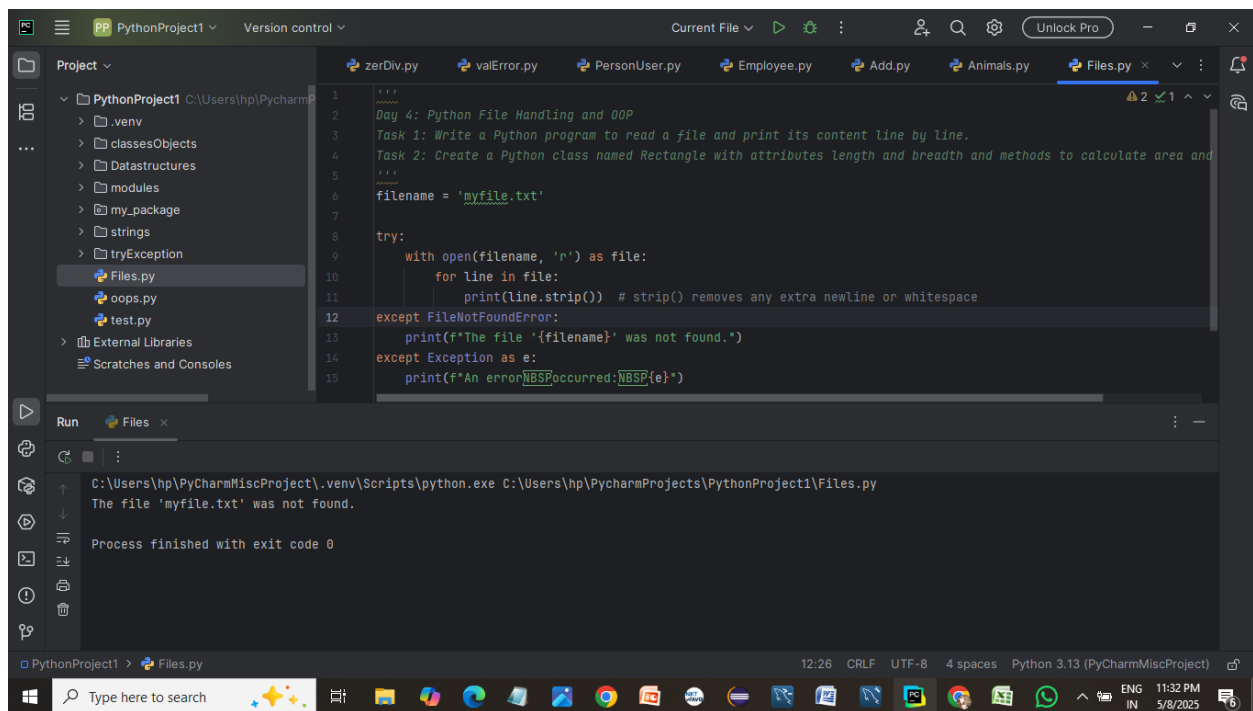
➤ Handle FileNotFoundError and general Exception

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)



```
1  """
2  Day 4: Python File Handling and OOP
3
4  Task 1: Write a Python program to read a file and print its content line by line.
5  Task 2: Create a Python class named Rectangle with attributes length and breadth and methods to calculate area and
6
7  """
8  filename = 'myfile.txt'
9
10 try:
11     with open(filename, 'r') as file:
12         for line in file:
13             print(line.strip()) # strip() removes any extra newline or whitespace
14 except FileNotFoundError:
15     print(f"The file '{filename}' was not found.")
16 except Exception as e:
17     print(f"An error occurred: {e}")
```

Run C:\Users\hp\PyCharmMiscProject\.venv\Scripts\python.exe C:\Users\hp\PyCharmProjects\PythonProject1\Files.py
The file 'myfile.txt' was not found.
Process finished with exit code 0

Exercise 4: Combine multiple except blocks for different errors

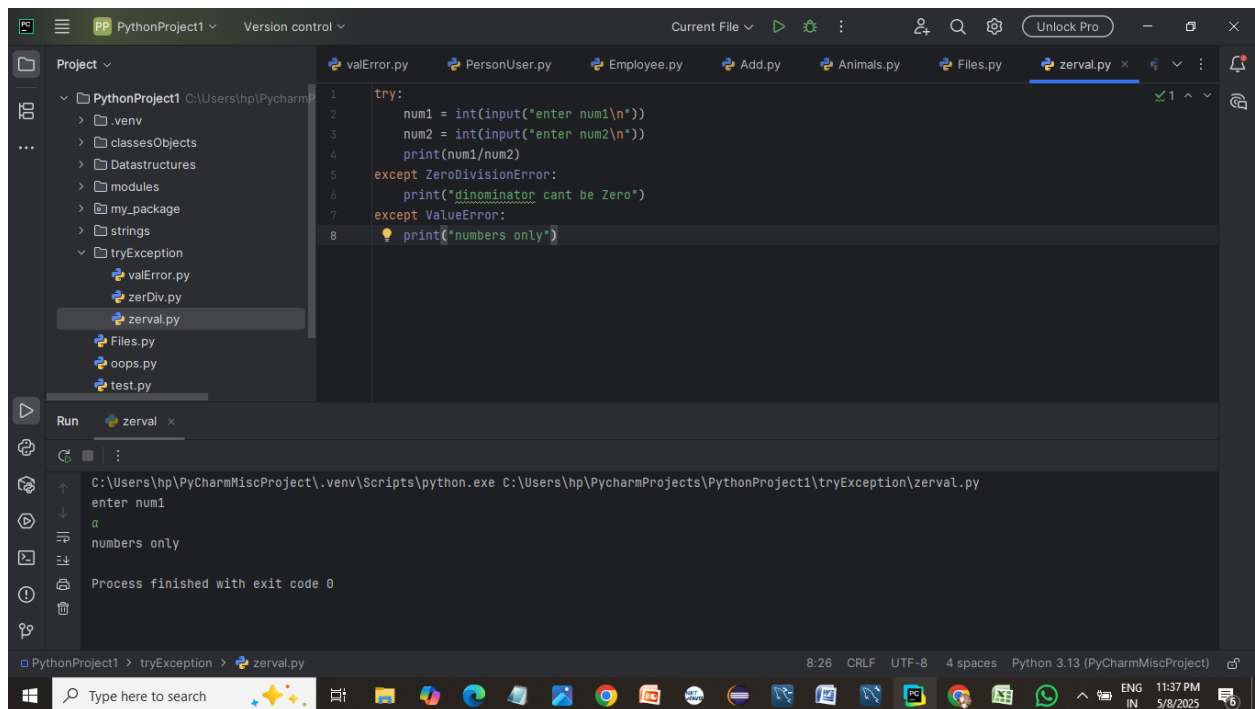
➤ Try converting input to float and dividing

➤ 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)



The screenshot shows the PyCharm IDE interface. The main editor window displays the following Python code:

```
1 try:
2     num1 = int(input("enter num1\n"))
3     num2 = int(input("enter num2\n"))
4     print(num1/num2)
5 except ZeroDivisionError:
6     print("dinominator cant be Zero")
7 except ValueError:
8     print("numbers only")
```

The left sidebar shows the project structure, with the file `zerval.py` selected under the `tryException` directory.

The bottom panel shows the Run output for the script `zerval`. The output is as follows:

```
C:\Users\hp\PyCharmMiscProject\.venv\Scripts\python.exe C:\Users\hp\PyCharmProjects\PythonProject1\tryException\zerval.py
enter num1
a
numbers only
Process finished with exit code 0
```

The status bar at the bottom indicates the file encoding is UTF-8, the line ending is CRLF, and the Python version is 3.13.

Exercise 5: Use try-except-else-finally

➤ Prompt for a number and divide it by 2

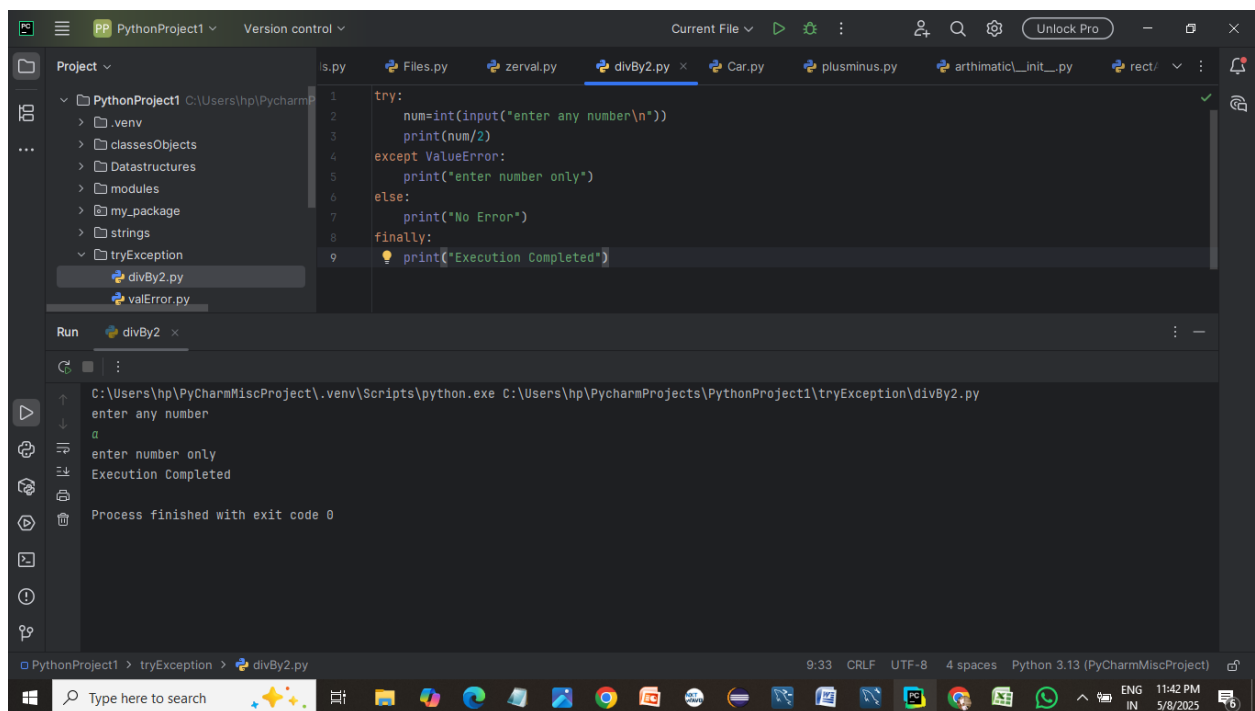
➤ Use else to confirm no error occurred

➤ 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)



The screenshot displays the PyCharm IDE interface. The top toolbar includes icons for file operations, running, and debugging. The left sidebar shows the project structure with folders like .venv, classesObjects, Datastructures, modules, my_package, strings, and tryException. The main editor window shows the code for divBy2.py, which is the same code as in the previous block. The Run window at the bottom shows the execution output: 'enter any number', 'a', 'enter number only', and 'Execution Completed'. The status bar at the bottom indicates the file is Python 3.13 (PyCharmMiscProject) and the system clock shows 11:42 PM on 5/8/2025.

Exercise 6: Create a custom exception called `NegativeNumberError`

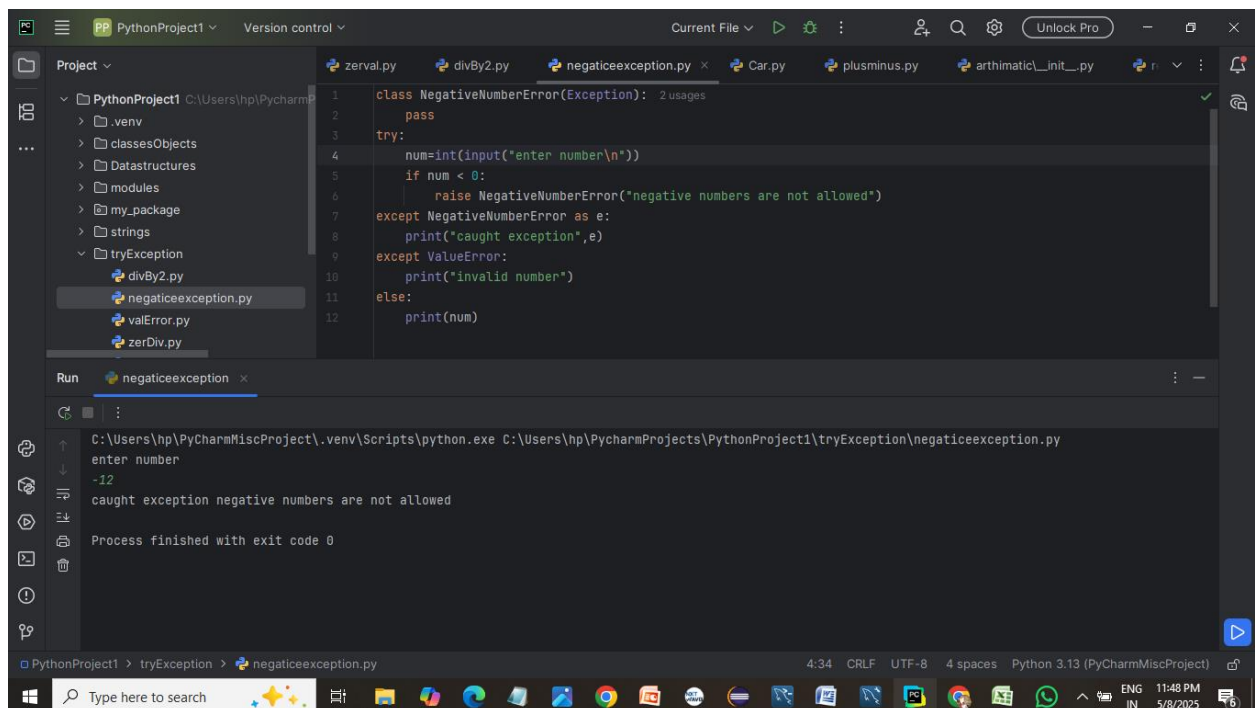
➤ Raise it if the user inputs a negative number

➤ 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)



The screenshot shows the PyCharm IDE interface. The top toolbar includes icons for file operations, running, and debugging, along with a 'Unlock Pro' button. The 'Project' sidebar on the left shows a tree view of the project structure, including folders like '.venv', 'classesObjects', 'Datastructures', 'modules', 'my_package', 'strings', and 'tryException'. The 'tryException' folder is expanded, showing files like 'divBy2.py', 'negativeexception.py', 'valError.py', and 'zerDiv.py'. The 'negativeexception.py' file is selected and its code is displayed in the main editor. The code is identical to the one shown in the 'Code)' block. Below the editor, the 'Run' window shows the execution of 'negativeexception.py'. The output is as follows:

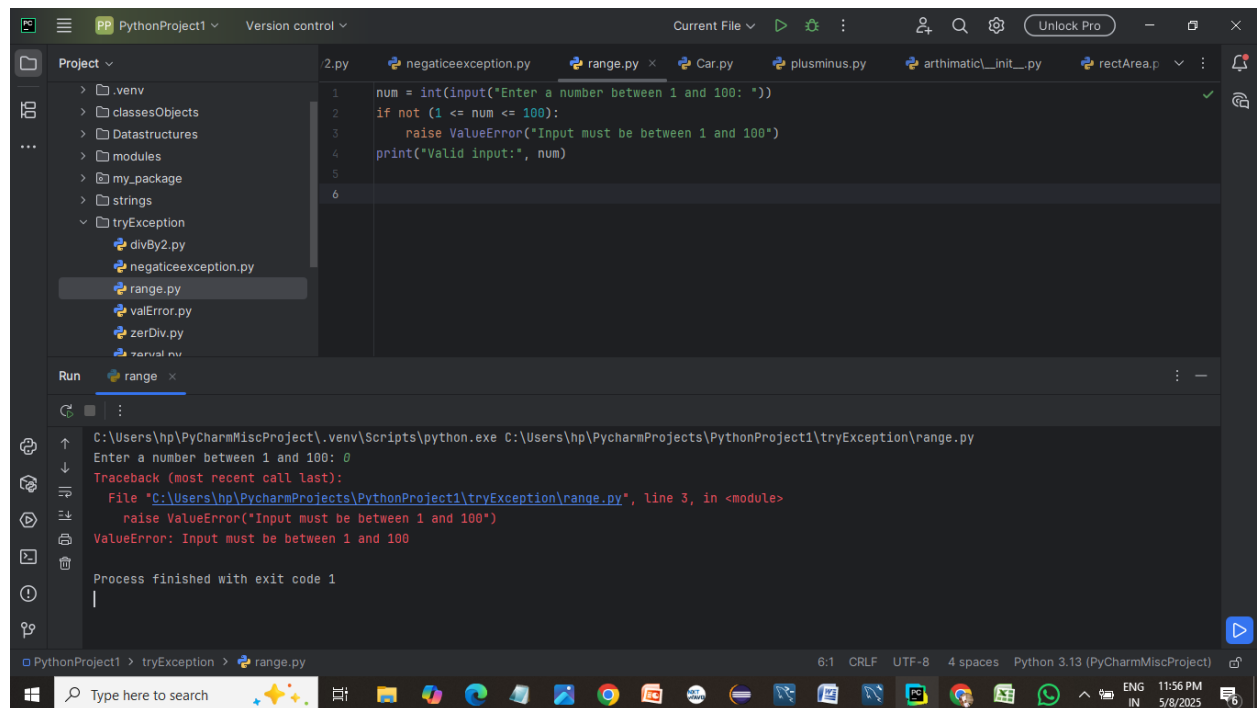
```
C:\Users\hp\PyCharmMiscProject\.venv\Scripts\python.exe C:\Users\hp\PyCharmProjects\PythonProject1\tryException\negativeexception.py
enter number
-12
caught exception negative numbers are not allowed
Process finished with exit code 0
```

The bottom status bar indicates the file encoding is UTF-8, the line length is 4 spaces, and the Python version is 3.13 (PyCharmMiscProject). The system tray at the bottom shows the date and time as 5/8/2025, 11:48 PM.

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

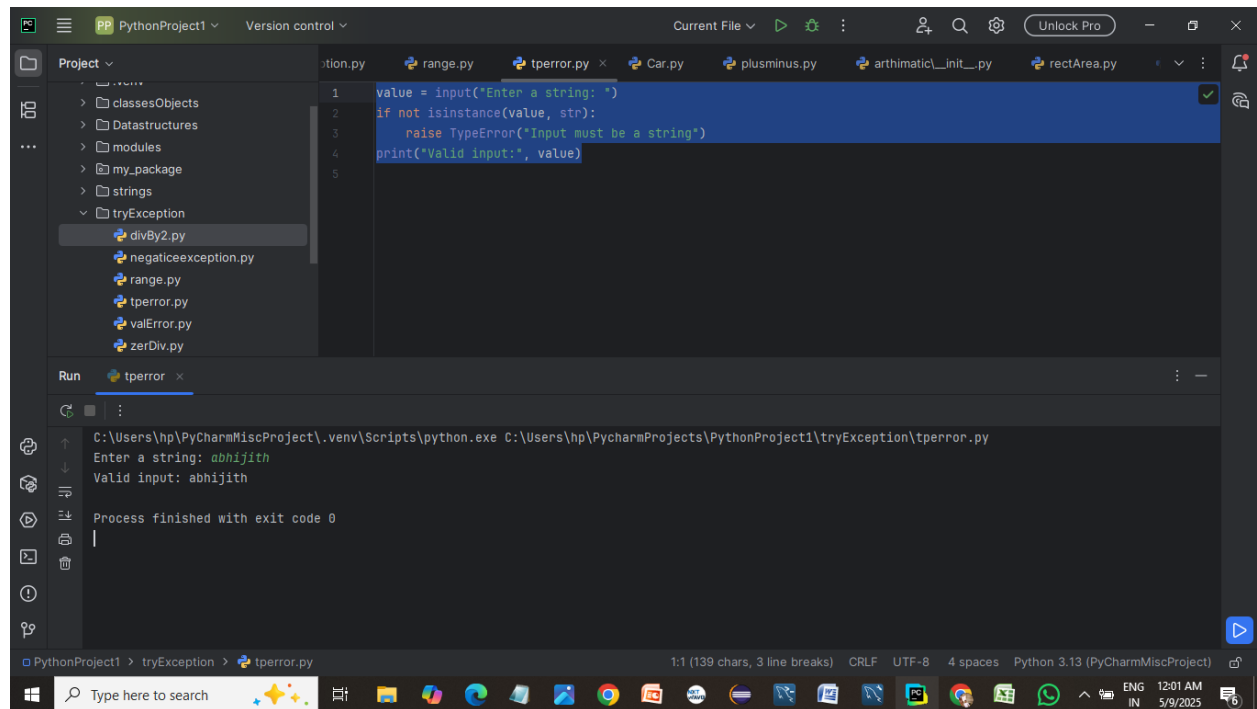
Exercise 8: Write a function that raises a `TypeError` if input is not a string

```
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)
```



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

Output)



The screenshot displays the PyCharm IDE interface. The top toolbar shows the 'Run' button (a green play icon). The left sidebar contains a 'Project' view with a tree structure showing folders like 'classesObjects', 'Datastructures', 'modules', 'my_package', 'strings', and 'tryException'. The 'tryException' folder is expanded, showing files: 'divBy2.py', 'negaticeexception.py', 'range.py', 'tperror.py', 'valError.py', and 'zerDiv.py'. The 'tperror.py' file is selected and its contents are displayed in the main editor window. The code in the editor is the same as shown in the first block. Below the editor is a 'Run' console window. It shows the command executed: 'C:\Users\hp\PyCharmMiscProject\.venv\Scripts\python.exe C:\Users\hp\PyCharmProjects\PythonProject1\tryException\tperror.py'. The output of the program is displayed below the command: 'Enter a string: abhijith' followed by 'Valid input: abhijith'. At the bottom, a status bar indicates the file is 'tperror.py' with 139 characters, 3 line breaks, CRLF line endings, UTF-8 encoding, 4 spaces indentation, and Python 3.13 interpreter.

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

Run console output:

```
C:\Users\hp\PyCharmMiscProject\.venv\Scripts\python.exe C:\Users\hp\PyCharmProjects\PythonProject1\tryException\tperror.py
Enter a string: abhijith
Valid input: abhijith
Process finished with exit code 0
```

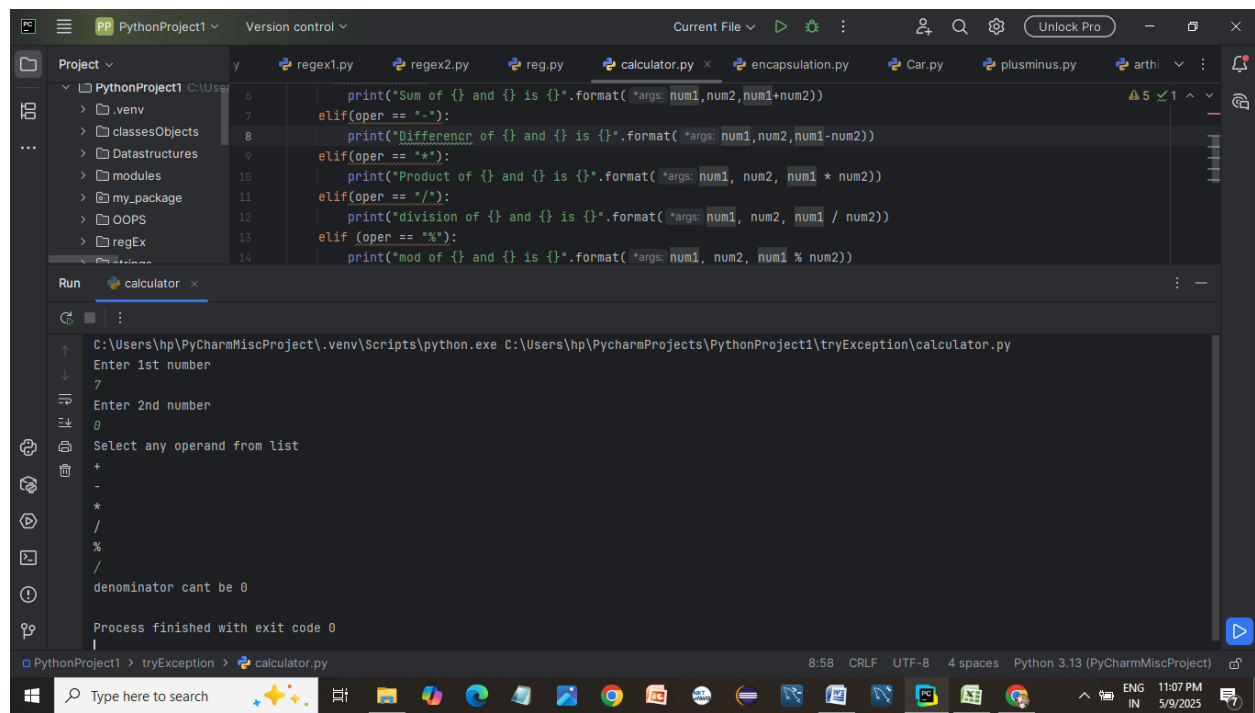

Exercise 9: Build a simple calculator with exception handling

➤ 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)



```
PythonProject1
venv
classesObjects
Datastructures
modules
my_package
OOPS
regEx
calculator.py
encapsulation.py
Car.py
plusminus.py
arhti

6 print("Sum of {} and {} is {}".format(*args: num1,num2,num1+num2))
7 elif(oper == "-"):
8     print("Differencr of {} and {} is {}".format(*args: num1,num2,num1-num2))
9 elif(oper == "*"):
10    print("Product of {} and {} is {}".format(*args: num1, num2, num1 * num2))
11 elif(oper == "/"):
12    print("division of {} and {} is {}".format(*args: num1, num2, num1 / num2))
13 elif(oper == "%"):
14    print("mod of {} and {} is {}".format(*args: num1, num2, num1 % num2))

Run calculator x

C:\Users\hp\PyCharmMiscProject\.venv\Scripts\python.exe C:\Users\hp\PyCharmProjects\PythonProject1\tryException\calculator.py
Enter 1st number
7
Enter 2nd number
0
Select any operand from list
+
-
*
/
%
/
denominator cant be 0
Process finished with exit code 0
```