**Task 9 : Implement Exception and exception Handling in Python**

**Task9.1: Division by Zero Exception**

**Aim:**  
To write a Python program that handles division by zero error using exception handling.

**Algorithm:**

1. Start the program.
2. Accept numerator and denominator from the user.
3. Use try block to perform division.
4. If denominator is zero, catch ZeroDivisionError in except block.
5. Display the appropriate error message.
6. End the program.

Program

try:

num1 = int(input("Enter numerator: "))

num2 = int(input("Enter denominator: "))

result = num1 / num2

print("Result:", result)

except ZeroDivisionError:

print("Error: Division by zero is not allowed!")

**Sample I/O**

**Enter numerator: 10**

**Enter denominator: 2**

**Result: 5.0**

**Enter numerator: 10**

**Enter denominator: 0**

**Error: Division by zero is not allowed!P**

**Task 9.2: Handling Multiple Exceptions**

**Aim:**  
To write a Python program that demonstrates handling of multiple exceptions such as invalid input and unexpected errors.

**Algorithm:**

1. Start the program.
2. Accept a number from the user.
3. Use try block to calculate the square of the number.
4. If the input is not a number, handle ValueError.
5. If any other error occurs, handle it using a general Exception.
6. Display the result or error message.
7. End the program.

Program

try:

num = int(input("Enter a number: "))

print("Square:", num \*\* 2)

except ValueError:

print("Error: Invalid input, please enter a number!")

except Exception as e:

print("Unexpected error:", e)

Sample I/O

Enter a number: 6

Square: 36

Enter a number: hello

Error: Invalid input, please enter a number!

**Task 9.3: Using finally Block**

**Aim:**  
To write a Python program that demonstrates the use of the finally block in exception handling.

**Algorithm:**

1. Start the program.
2. Try to open and read from a file.
3. If the file is not found, handle the FileNotFoundError.
4. Use the finally block to print a completion message (it executes always).
5. End the program.

Program

try:

file = open("sample.txt", "r")

content = file.read()

print(content)

except FileNotFoundError:

print("Error: File not found!")

finally:

print("Execution completed (finally block runs always).")

Sample I/O

(sample.txt contains: "Hello Python")

Hello Python

Execution completed (finally block runs always).

Error: File not found!

Execution completed (finally block runs always).

**Task 9.4: User-Defined Exception**

**Aim:**  
To write a Python program that demonstrates user-defined exceptions.

**Algorithm:**

1. Start the program.
2. Define a custom exception class (e.g., NegativeNumberError).
3. Accept a number from the user.
4. If the number is negative, raise the user-defined exception.
5. Catch the exception in the except block and display the error message.
6. If no error, print the number entered.
7. End the program.

**Program**

class NegativeNumberError(Exception):

pass

try:

num = int(input("Enter a positive number: "))

if num < 0:

raise NegativeNumberError("Negative number entered!")

print("You entered:", num)

except NegativeNumberError as e:

print("Error:", e)

**Sample I/O**

Enter a positive number: 15

You entered: 15

Enter a positive number: -8

Error: Negative number entered!