**Que 1 : Introduction to exceptions and how to handle them using try, except, and finally.**

**What is an exception?**

An exception is an **error** that occurs during the execution of a program.

It **disrupts** the normal flow of the program.

**Examples of exceptions**:

* Dividing by zero → ZeroDivisionError
* Accessing a variable that doesn’t exist → NameError
* Opening a file that doesn’t exist → FileNotFoundError

**Why handle exceptions?**

If you don’t handle exceptions, your program will stop running when an error occurs.  
By **handling exceptions**, you can:  
✅ Prevent the program from crashing  
✅ Show a friendly error message  
✅ Continue execution after the error

**Handling Exceptions with try, except, and finally**

**try** → Block of code where you think an error might occur.

**except** → Block that runs if an error happens in the try block.

**finally** → Block that always runs, whether an error happens or not (used for cleanup tasks like closing files).

**Que 2 : Understanding multiple exceptions and custom exceptions.**

**1. Understanding Multiple Exceptions**

Sometimes, a piece of code can raise different types of exceptions.  
You can handle them in **two main ways**:

**A. Multiple except Blocks**

You handle each exception separately.

try:

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

y = 10 / x

print("Result:", y)

except ZeroDivisionError:

print("Error: Cannot divide by zero.")

except ValueError:

print("Error: Please enter a valid integer.")

except Exception as e: # Catches any other error

print("Unexpected error:", e)

**B. Handling Multiple Exceptions in One Line**

If the handling logic is the same for multiple exceptions.

try:

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

y = 10 / x

except (ZeroDivisionError, ValueError):

print("Error: Either division by zero or invalid input.")

**2. Custom Exceptions**

Python lets you create your **own exception classes** for specific application needs.  
They are useful when:

* Built-in exceptions don’t describe the problem well.
* You want clearer, more descriptive error handling.

**Creating a Custom Exception**

A custom exception is just a class that inherits from Exception.

class NegativeNumberError(Exception):

"""Custom exception for negative numbers."""

pass

try:

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

if num < 0:

raise NegativeNumberError("Negative numbers are not allowed!")

print("You entered:", num)

except NegativeNumberError as e:

print("Custom Exception:", e)