## 🐍 Error Handling in Python (Simple Explanation)

### ✅ What is Error Handling?

Error handling in Python means managing situations where your code runs into problems (called **exceptions**) so your program doesn’t crash.

**Common errors:**

* Dividing by zero
* File not found
* Invalid input (like entering text when a number is expected)

We use error handling to catch and fix these problems.

### 📦 Python Uses try, except, else, and finally

* try: Write code that might cause an error.
* except: Handle the error if it happens.
* else: Run this if there is **no error**.
* finally: Always runs (whether there is an error or not).

### ✅ Basic Syntax

try:  
 # risky code  
except SomeError:  
 # code to handle the error  
else:  
 # runs if no error  
finally:  
 # runs no matter what

### 🔍 Example 1: Divide by Zero Error

try:  
 result = 10 / 0  
except ZeroDivisionError:  
 print("You can't divide by zero!")

**Output:**

You can't divide by zero!

### 🔍 Example 2: File Not Found Error

try:  
 f = open("data.txt", "r")  
 content = f.read()  
 f.close()  
except FileNotFoundError:  
 print("File not found!")

**Output:**

File not found!

### 🔍 Example 3: Using else and finally

try:  
 num = int(input("Enter a number: "))  
 result = 100 / num  
except ValueError:  
 print("That's not a number!")  
except ZeroDivisionError:  
 print("You can't divide by zero!")  
else:  
 print("Result is", result)  
finally:  
 print("This always runs.")

**Possible Output 1 (if user enters** ``**):**

Result is 4.0  
This always runs.

**Possible Output 2 (if user enters** ``**):**

That's not a number!  
This always runs.

### 📌 Extra Notes

* You can handle **multiple exceptions** using multiple except blocks.
* Use finally for cleanup actions like closing files or releasing resources.
* Use else only if you want to run code when there are **no exceptions**.

### ✅ Want More?

You can also raise your own exceptions using raise or create custom error classes. Let me know if you’d like that included!