Python Exception Handling – Beginner Friendly Guide

1. What is an Exception?

An exception is a runtime error that interrupts your program. Exception handling lets you catch these errors and keep the program running smoothly.

Common Exception Types:
- ValueError — Invalid value provided (e.g., converting letters to numbers)
- TypeError — Operation performed on wrong type
- ZeroDivisionError — Dividing by zero
- FileNotFoundError — Accessing a file that doesn't exist
- KeyError — Accessing missing dictionary key

2. Basic Try–Except Example

```
try:
    num = int("hello")
    print(num)
except ValueError:
    print("Error: Cannot convert text to number!")
```

3. Multiple Except Blocks

```
try:
    value = 10 / 0
except ZeroDivisionError:
    print("You tried dividing by zero!")
except Exception as error:
    print("Some other error occurred:", error)
```

4. Try–Except–Else Block

The **else** block runs only when the try block succeeds without errors.

```
try:
    x = int("20")
except ValueError:
    print("Invalid input!")
else:
    print("Success! Converted value:", x)
```

5. Try–Except–Finally Block

The **finally** block always executes—useful for closing files or cleaning up resources.

```
try:
    file = open("unknown_file.txt")
except FileNotFoundError:
    print("File not found!")
finally:
    print("Cleaning up... This runs no matter what.")
```

6. Raising Your Own Exceptions

```
age = -1

if age < 0:
    raise ValueError("Age cannot be negative!")
```

7. Custom Exceptions

```
class SalaryError(Exception):
    pass

salary = -1000
if salary < 0:
    raise SalaryError("Salary cannot be negative!")
```

8. Practical Real-World Examples

A. Reading a File Safely

```
try:
    with open("data.txt") as f:
        content = f.read()
        print(content)
except FileNotFoundError:
    print("Oops! The file does not exist.")
```

B. Safe User Input

```
while True:
    try:
        num = int(input("Enter a number: "))
        print("You entered:", num)
        break
    except ValueError:
        print("Please enter digits only!")
```

Summary of Key Concepts

- Use try–except to handle errors and avoid program crashes
- Multiple except blocks let you handle different errors
- Use else for code that runs only if no error occurs
- Use finally for cleanup actions
- Raise custom exceptions for better control