Python Exception Handling: try, except, else, finally

# 1. try Block

Purpose:  
- Place the code that might raise an exception.  
  
Syntax:  
try:  
 # Code that might throw an exception  
  
When to Use:  
- When there’s a risk of runtime errors (file operations, type conversion, division, etc.)

# 2. except Block

Purpose:  
- Runs only if an exception occurs in the try block.  
  
Syntax:  
except ExceptionType:  
 # Code that handles the exception  
  
To catch any exception:  
except:  
 # General exception handling  
  
When to Use:  
- Handle errors like ZeroDivisionError, ValueError, etc., to avoid program crashes.

# 3. else Block

Purpose:  
- Executes only if the try block didn't raise an exception.  
  
Syntax:  
try:  
 # Risky code  
except:  
 # Handle error  
else:  
 # Run this if no error occurred  
  
When to Use:  
- For code that should run only if try is successful.

# 4. finally Block

Purpose:  
- Always executes whether or not an exception was raised.  
  
Syntax:  
try:  
 # Risky code  
except:  
 # Handle error  
finally:  
 # Always runs  
  
When to Use:  
- Cleanup actions (close files, release resources, logging, etc.)

# Example Code:

def divide(a, b):  
 try:  
 result = a / b  
 except ZeroDivisionError:  
 print("Error: Cannot divide by zero!")  
 else:  
 print(f"Division successful: {result}")  
 finally:  
 print("Execution completed.")  
  
divide(10, 2)  
divide(10, 0)