EXCEPTION HANDLING IN PYTHON

What is Exception Handling?

Exception handling is a mechanism in Python to handle runtime errors (exceptions) so that the program doesn't crash unexpectedly. It allows us to manage errors gracefully using try-except blocks.

What is an Exception?

An exception is an error that occurs during program execution, disrupting the normal flow of the program.

Example:

```
num = 10 / 0  # ZeroDivisionError
```

Type	Description	Example
Syntax Error	Mistake in code structure (detected before execution).	if True print("Hello") (Missing :)
Exception	Occurs during execution (even if syntax is correct).	10 / 0 (ZeroDivisionError)

Handling Exceptions using try-except

The try block contains code that may raise an exception, and the except block handles the error.

Example:

```
try:
    num = int(input("Enter a number: "))
    result = 10 / num
    print("Result:", result)
except ValueError:
    print("Error: Invalid input! Enter a number.")
```

Handling Multiple Exceptions

Using multiple except blocks:

```
try:
    num = int(input("Enter a number: "))
    result = 10 / num
except ZeroDivisionError:
    print("Cannot divide by zero!")
except ValueError:
    print("Invalid input! Please enter a number.")
```

Using a single except block with a tuple:

```
try:
    num = int(input("Enter a number: "))
    result = 10 / num
except (ZeroDivisionError, ValueError) as e:
    print("Error:", e)
```

Using else with try-except

The else block runs only if there's no exception.

Example:

```
try:
    num = int(input("Enter a number: "))
    result = 10 / num
except ZeroDivisionError:
    print("Cannot divide by zero!")
except ValueError:
    print("Invalid input!")
else:
    print("Result:", result)
```

Using finally (Always Executes)

The finally block runs no matter what, even if an exception occurs.

Example:

```
try:
    file = open("test.txt", "r")
    data = file.read()
except FileNotFoundError:
    print("File not found!")
finally:
    print("Closing file (if it was opened).")
```

Raising Exceptions (raise)

You can manually raise exceptions using raise.

Example:

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

Custom Exceptions (class)

You can create your own exceptions by defining a new class that inherits from Exception.

Example:

```
class NegativeNumberError(Exception):
    pass

num = int(input("Enter a positive number: "))
if num < 0:
    raise NegativeNumberError("Negative numbers are not allowed!")
```

Full Exception Handling Structure

```
try:
    num = int(input("Enter a number: "))
    result = 10 / num
except ZeroDivisionError:
    print("Cannot divide by zero!")
except ValueError:
    print("Invalid input! Enter a number.")
else:
    print("Result:", result)
finally:
    print("Execution completed.")
```

Common Built-in Exceptions

Exception	Meaning
ZeroDivisionError	Division by zero error
ValueError	Invalid value (e.g., converting a string to an integer)
TypeError	Operation between incompatible types
IndexError	Accessing an out-of-range list index
KeyError	Accessing a non-existent dictionary key
FileNotFoundError	Trying to open a file that doesn't exist
AttributeError	Accessing an undefined attribute of an object
ImportError	Importing a non-existent module
NameError	Using a variable that is not defined