Python Exception Handling

Exception handling is a crucial aspect of Python programming. Let's explore how to handle errors gracefully and ensure smooth execution.

by Harish Kumar Sharma

What Are Exceptions?

Exceptions in Python are events that occur during the execution of a program, disrupting its normal flow. They can be errors or other exceptional conditions.

The Try-Except Block

The try-except block is used to handle exceptions. The code within the try block is executed, and if an exception occurs, it is caught and processed in the except block.

Handling Specific Exceptions

We can specify different exception types to handle specific errors. By catching and handling specific exceptions, we can customize our response based on the type of error.

The Else Clause

The else clause in exception handling is executed only if no exceptions are raised in the try block. It allows us to define code that should be executed when the try block finishes successfully.

The Finally Block

The finally block is always executed, regardless of whether an exception occurred or not. It is placed after the try and except blocks and is commonly used for cleaning up resources.

Raising Exceptions

In Python, we can intentionally raise exceptions using the raise statement. This allows us to create custom exceptions and handle them appropriately in our code.

Common Exceptions and Their Descriptions

Exception	Description
TypeError	Raised when an operation or function is applied to an object of an inappropriate type.
ValueError	Raised when a function receives an argument of the correct type but with an invalid value.
IndexError	Raised when trying to access an index that is outside the bounds of a list or other sequence.
FileNotFoundError	Raised when a file or directory is requested but cannot be found.