Error and Exception Handling in Python

By Bhimashankar Takalki

Error and exception handling in Python is a crucial aspect of writing robust and reliable code. Python provides a robust mechanism for handling errors and exceptions using try, except, else, and finally blocks.

try, except, else, and finally Blocks:

- The try block is used to enclose the code that might raise an exception.
- The except block is used to handle the exception raised in the try block.
- The else block is executed if no exceptions are raised in the try block.
- The finally block is always executed, regardless of whether an exception occurred.

```
try:
    # Code that might raise an exception
    result = 10 / 0
except ZeroDivisionError as e:
    # Handle specific exception
    print("Error:", e)
except Exception as e:
    # Handle any other exception
    print("An error occurred:", e)
else:
    # Execute if no exception occurs
    print("No exception occurred")
finally:
    # Always executed, regardless of exceptions
    print("Finally block executed")
```

Handling Specific Exceptions:

You can handle specific exceptions using multiple except blocks, each handling a different type of exception.

```
try:
    # Code that might raise an exception
    file = open('nonexistent.txt', 'r')
except FileNotFoundError as e:
    # Handle file not found exception
    print("File not found:", e)
except IOError as e:
    # Handle I/O error
    print("I/O error:", e)
except Exception as e:
    # Handle any other exception
    print("An error occurred:", e)
```

Raising Exceptions:

You can raise exceptions explicitly using the raise statement.

```
try:
    # Code that might raise an exception
    age = int(input("Enter your age: "))
    if age < 0:
        raise ValueError("Age must be a positive number")
except ValueError as e:
    # Handle value error
    print("Invalid age:", e)
```

Handling Exceptions with else Block:

The else block is executed if no exceptions are raised in the try block.

```
try:
    # Code that might raise an exception
    result = 10 / 2
except ZeroDivisionError as e:
    # Handle specific exception
    print("Error:", e)
else:
    # Execute if no exception occurs
    print("Result:", result)
```

finally Block:

The finally block is always executed, regardless of whether an exception occurred in the try block.

```
try:
    # Code that might raise an exception
    file = open('example.txt', 'r')
    data = file.read()
except IOError as e:
    # Handle I/O error
    print("I/O error:", e)
finally:
    # Always executed, regardless of exceptions
    file.close()
```