Exeptions

Ba Nguyễn

assert

raise

try

except

else

finally

Errors

Syntax Errors (Parsing Error)

Không tuân theo cấu trúc (cú pháp) hợp lệ trong Python gây dừng chương trình

Logical Errors (Exceptions)

Lỗi phát hiện trong quá trình thực thi chương trình, lỗi không gây dừng chương trình, nhưng thay đổi luồng hoạt động của nó

Built-in Exceptions

Khi một lỗi xảy ra, Python sẽ tự động đưa ra loại exception phù hợp và thông báo lỗi chi tiết.

Python bao gồm rất nhiều built-in exceptions khác nhau, cũng như cho phép khai báo các exceptions tùy chỉnh.

Danh sách đầy đủ tại đây

Built-in Exceptions

`AssertionError`	Khi câu lệnh **assert** không thành công
`AttributeError`	Khi tham chiếu hoặc gán giá trị cho thuộc tính
`ImportError`	Khi không tìm thấy module
`IndexError`	Khi index vượt quá phạm vi (chuỗi, list, ...)
`KeyError`	Khi key không tồn tại trong Dictionary
`KeyboardInterrupt`	Khi người dùng dừng chương trình (Ctrl + C)
`NameError`	Khi không tìm thấy một biến

Built-in Exceptions

`SyntaxError`	Khi trình phân tích phát hiện lỗi cú pháp
`IndentationError`	Khi phần thụt lề không hợp lệ
`TypeError`	Kiểu dữ liệu không hợp lệ
`ValueError`	Khi đúng kiểu dữ liệu, nhưng giá trị không phù hợp
`ZeroDivisionError`	Khi thực hiện phép chia cho 0
`FileNotFoundError`	Khi không tìm thấy file yêu cầu

Raising Exceptions

```
number = int(input("Enter a positive integer:"))

if number <= 0:
    # Có thể chủ động ném ra một exception
    # khi một điều kiện nhất định xảy ra
    # Cung cấp thêm các tham số để xác định vấn đề
    raise ValueError(f"{number} is not a positive number")
```

Raising Exceptions

```
number = int(input("Enter a positive integer:"))

# Cũng có thể sử dụng assert để đánh giá một điều kiện
# Nếu điều kiện đúng, chương trình tiếp tục
# Nếu điều kiện sai, ném ra một AssertionError
assert number > 0, f"{number} is not a positive number"
```

Handling Exception

```
try:
    "Các câu lệnh có thể gây ra lỗi"
except:
    "Mã xử lý khi có lỗi"
else:                                   # optional
    "Mã xử lý khi không có lỗi"
finally:                                # optional
    "Mã thực thi cuối cùng, bất kể lỗi hay không"
```

Handling Specific Exception

```
try:
    pass
except ValueError as e:
    "Xử lý 1 exception cụ thể"
except (TypeError, ZeroDivisionError) as e:
    "Xử lý nhiều exceptions cùng lúc"
except:
    "Xử lý các exceptions khác"
```

Clean-up Actions

```
try:
    pass
finally:
    """Thực hiện các thao tác dọn dẹp
    như đóng file, ngắt kết nối CSDL, ...

    Finally đảm bảo việc thực hiện các thao tác này"""
```

User-defined Exceptions

```
class CustomError(Exception):
    """User-defined exception

    Attributes:
        msg -- explanation of the error
    """

    def __init__(self, msg):
        self.msg = msg
```

Summary

- Exception được đưa ra trong quá trình thực thi, nếu không được xử lý sẽ khiến chương trình bị dừng đột ngột
- Sử dụng khối lệnh try except để xử lý exceptions, thực hiện các hành động phù hợp
- Có thể chủ động đưa ra một exception hoặc tạo một loại exception mới khi phù hợp với một điều kiện chỉ định
- Nên bóc tách và xử lý các exception riêng biệt
- Sử dụng finally để thực hiện các thao tác dọn dẹp