Xử lý ngoại lệ trong Python

Thời gian dự kiến cần thiết: 10 phút

Mục tiêu

1. Hiểu ngoại lệ
2. Phân biệt lỗi với ngoại lệ
3. Làm quen với các ngoại lệ phổ biến của Python
4. Quản lý ngoại lệ hiệu quả

Trong thế giới lập trình, sai sót và những tình huống không mong muốn là điều chắc chắn xảy ra. Python, ngôn ngữ lập trình phổ biến và linh hoạt, trang bị cho các nhà phát triển bộ công cụ mạnh mẽ để quản lý các tình huống không lường trước này thông qua các trường hợp ngoại lệ và xử lý lỗi.

Ngoại lệ là gì?

Trường hợp ngoại lệ là cảnh báo khi có điều gì đó không mong muốn xảy ra trong khi chạy chương trình. Đó có thể là một lỗi trong mã hoặc một tình huống không được lên kế hoạch. Python có thể tự động đưa ra các cảnh báo này, nhưng chúng ta cũng có thể cố ý kích hoạt chúng bằng cách sử dụng lệnh raise. Điều thú vị là chúng ta có thể ngăn chương trình của mình gặp sự cố bằng cách xử lý các ngoại lệ.

Lỗi so với ngoại lệ

Đợi đã, sự khác biệt giữa lỗi và ngoại lệ là gì? Vâng, `errors`thường là những vấn đề lớn xuất phát từ máy tính hoặc hệ thống. Chúng thường làm cho chương trình ngừng hoạt động hoàn toàn. Mặt khác, `exceptions`chúng giống như những vấn đề mà chúng ta có thể kiểm soát hơn. Chúng xảy ra do chúng tôi đã làm gì đó trong mã của mình và thường có thể sửa được, vì vậy chương trình sẽ tiếp tục.

Đây là sự khác biệt giữa `Errors and exceptions`: -

Diện mạo	Lỗi	Ngoại lệ
Nguồn gốc	Lỗi thường do môi trường, phần cứng hoặc hệ điều hành gây ra.	Các ngoại lệ thường là kết quả của việc thực thi mã có vấn đề trong chương trình.
Thiên nhiên	Lỗi thường nghiêm trọng và có thể dẫn đến lỗi chương trình hoặc chấm dứt bất thường.	Các trường hợp ngoại lệ thường ít nghiêm trọng hơn và có thể được phát hiện và xử lý để ngăn chặn việc chấm dứt chương trình.
Sự điều khiển	Lỗi thường không được chương trình phát hiện hoặc xử lý.	Các ngoại lệ có thể được phát hiện bằng cách sử dụng các khối thử ngoại trừ và được xử lý một cách khéo léo, cho phép chương trình tiếp tục thực thi.
Ví dụ	Các ví dụ bao gồm “SyntaxError” do cú pháp không chính xác hoặc “NameError” khi một biến không được xác định.	Các ví dụ bao gồm “ZeroDivisionError” khi chia cho 0 hoặc “FileNotFoundError” khi cố mở một tệp không tồn tại.
Phân loại	Lỗi không được phân loại thành các loại.	Các ngoại lệ được phân loại thành nhiều lớp khác nhau, chẳng hạn như “ArithmeticError”, “IOError”, ValueError, v.v., dựa trên bản chất của chúng.

Các ngoại lệ phổ biến trong Python

Dưới đây là một số ví dụ về các trường hợp ngoại lệ mà chúng tôi thường gặp và có thể xử lý bằng công cụ này:

- **ZeroDivisionError:** Lỗi này phát sinh khi cố gắng chia một số cho 0. Phép chia cho số 0 không được xác định trong toán học, gây ra lỗi số học. Ví dụ:
  Ví dụ:

```
1. 1
2. 2
3. 3
```

```
1. result = 10 / 0
2. print(result)
3. # Raises ZeroDivisionError
```

Đã sao chép!

- **ValueError:** Lỗi này xảy ra khi sử dụng giá trị không phù hợp trong mã. Một ví dụ về điều này là khi cố gắng chuyển đổi một chuỗi không phải là số thành số nguyên:
  Ví dụ:

```
1. 1
2. 2
```

```
1. num = int("abc")
2. # Raises ValueError
```

Đã sao chép!

- **FileNotFoundError:** Ngoại lệ này xảy ra khi cố gắng truy cập vào một tệp không tồn tại.
  Ví dụ:

```
1. 1
2. 2
```

```
1. with open("nonexistent_file.txt", "r") as file:
2.         content = file.read()   # Raises FileNotFoundError
```

Đã sao chép!

- **IndexError:** IndexError xảy ra khi một chỉ mục được sử dụng để truy cập một phần tử trong danh sách nằm ngoài phạm vi chỉ mục hợp lệ.
  Ví dụ:

```
1. 1
2. 2
3. 3
```

```
1. my_list = [1, 2, 3]
2. value = my_list[1]  # No IndexError, within range
3. missing = my_list[5]  # Raises IndexError
```

Đã sao chép!

- **KeyError:** KeyError phát sinh khi cố gắng truy cập vào khóa không tồn tại trong từ điển.
  Ví dụ:

```
1. 1
2. 2
3. 3
```

```
1. my_dict = {"name": "Alice", "age": 30}
2. value = my_dict.get("city")  # No KeyError, using .get() method
3. missing = my_dict["city"]  # Raises KeyError
```

Đã sao chép!

- **TypeError:** TypeError xảy ra khi một đối tượng được sử dụng theo cách không tương thích. Một ví dụ bao gồm việc cố gắng nối một chuỗi và một số nguyên:
  Ví dụ:

```
1. 1
2. 2
```

```
1. result = "hello" + 5
2. # Raises TypeError
```

Đã sao chép!

- **AttributionError:** AttributionError xảy ra khi một thuộc tính hoặc phương thức được truy cập trên một đối tượng không sở hữu thuộc tính hoặc phương thức cụ thể đó. Ví dụ:
  Ví dụ:

```
1. 1
2. 2
3. 3
```

```
1. text = "example"
2. length = len(text)  # No AttributeError, correct method usage
3. missing = text.some_method()  # Raises AttributeError
```

Đã sao chép!

- **ImportError:** Lỗi này xảy ra khi cố gắng nhập một mô-đun không khả dụng. Ví dụ:`import non_existent_module`

> Lưu ý: **Hãy nhớ rằng, những trường hợp ngoại lệ bạn sẽ gặp phải không chỉ giới hạn ở những trường hợp này. Có nhiều hơn nữa trong Python. Tuy nhiên, không cần phải lo lắng. Bằng cách sử dụng kỹ thuật được cung cấp bên dưới và làm theo cú pháp đúng, bạn sẽ có thể xử lý mọi trường hợp ngoại lệ xảy ra theo cách của mình.**

Xử lý ngoại lệ:

Python có một công cụ tiện dụng được gọi là `try and except`giúp chúng ta quản lý các ngoại lệ.

Thử và Ngoại trừ : Bạn có thể sử dụng khối thử và ngoại trừ để ngăn chương trình của bạn gặp sự cố do ngoại lệ.

Đây là cách chúng hoạt động:

1. Mã có thể dẫn đến ngoại lệ được chứa trong khối thử.
2. Nếu một ngoại lệ xảy ra, mã sẽ trực tiếp chuyển sang khối ngoại trừ.
3. Trong khối ngoại trừ, bạn có thể xác định cách xử lý ngoại lệ một cách khéo léo, như hiển thị thông báo lỗi hoặc thực hiện các hành động thay thế.
4. Sau khối ngoại trừ, chương trình tiếp tục thực thi đoạn mã còn lại.

Ví dụ: Đang cố chia cho số 0

```
1. 1
2. 2
3. 3
4. 4
```

```
5. 5
6. 6
7. 7
8. số 8
9. 9
```

```
1. # using Try- except
2. try:
3.     # Attempting to divide 10 by 0
4.     result = 10 / 0
5. except ZeroDivisionError:
6.     # Handling the ZeroDivisionError and printing an error message
7.     print("Error: Cannot divide by zero")
8. # This line will be executed regardless of whether an exception occurred
9. print("outside of try and except block")
```

Đã sao chép!

Bước tiếp theo

Khi chúng ta đọc xong phần này, bạn đã sẵn sàng chuyển sang phần tiếp theo nơi bạn sẽ thực hành xử lý lỗi. Để học tập tốt hơn, hãy thử các loại dữ liệu khác nhau trong phòng thí nghiệm. Bằng cách này, bạn sẽ gặp phải nhiều lỗi khác nhau và học cách xử lý chúng một cách hiệu quả. Kiến thức này sẽ giúp bạn viết mã mạnh hơn và đáng tin cậy hơn trong tương lai.

(Các) tác giả

Akansha Yadav