21. Xử lý ngoại lệ

Một ngoại lệ là một lỗi xảy ra trong quá trình thực hiện một chương trình. Các ngoại lệ được biết đến với các lập trình viên không phải là những trường hợp không phù hợp với quy tắc chung. Tên "ngoại lệ" trong khoa học máy tính cũng có ý nghĩa này: Nó hàm ý rằng vấn đề (ngoại lệ) không xảy ra thường xuyên, ví dụ ngoại lệ là "ngoại lệ đối với quy tắc". Xử lý ngoại lệ là một cấu trúc trong một số ngôn ngữ lập trình để xử lý các lỗi tự động. Nhiều ngôn ngữ lập trình như C++, Objective-C, PHP, Java, Ruby, Python, và nhiều ngôn ngữ khác đã được xây dựng để hỗ trợ cho việc xử lý ngoại lệ.

Xử lý lỗi thường được giải quyết bằng cách lưu trạng thái tại thời điểm xảy ra lỗi và làm gián đoạn luồng thông thường của chương trình để thực hiện một hàm hoặc đoạn mã đặc biệt, được gọi là trình xử lý ngoại lệ. Tùy thuộc vào loại lỗi ("phép chia cho số không", "lỗi mở tập tin" vv) xảy ra, trình xử lý lỗi có thể "khắc phục" sự cố và chương trình có thể được tiếp tục sau đó với dữ liệu đã lưu trước đó.

Xử lý ngoại lệ trong Python.

Ngoại lệ xử lý trong Python rất giống với Java. Mã, chứa nguy cơ ngoại lệ, được nhúng trong khối “try-except” trong python hay “try-catch” trong java . Nó là có thể để tùy chỉnh các trường hợp ngoại lệ, thậm chí ta cố tình hay buộc chương trình bắn ra một ngoại lệ.

Hãy xem xét một ví dụ đơn giản. Giả sử chúng ta muốn yêu cầu người dùng nhập một số nguyên, raw_input() sẽ đợi ta nhập giá trị từ bàn phím, đầu vào mong đợi của chuỗi là một số nguyên. Nếu đầu vào không phải là một số nguyên hợp lệ, chúng ta sẽ bắn ra một giá trị ValueError. Chúng tôi hiển thị điều này trong đoạn mã sau:

Đoạn mã lỗi xảy ra khi tôi nhập vào chuỗi “ref” - không phải là một số nguyên-trong trường hợp khi ta không bắt ngoại lệ:

```
n = int(raw_input("Please enter a number: "))


Output:


Please enter a number: ref

Traceback (most recent call last):

  File "C:/Users/cuongtran/PycharmProjects/BigData/course.py", line 1, in <module>

    n = int(raw_input("Please enter a number: "))

ValueError: invalid literal for int() with base 10: 'ref'
```

Cùng mong muốn trên ta dùng thêm đoạn mã sử dụng block “try-except” để bắt ngoại lệ.

```
while True:
    try:
        n = raw_input("Please enter an integer: ")
        n = int(n)
        break
    except ValueError:
        print("No valid integer! Please try again ...")


Output:
```

```
Please enter an integer: ref

No valid integer! Please try again ...

Please enter an integer: 23
```

Bắt nhiều ngoại lệ:

Khi đoạn mã có thể có nhiều hơn một ngoại lệ. Nhưng khi xẩy ra ngoại lệ, chỉ có 1 ngoại lệ được dùng.

Ví dụ tiếp theo của chúng tôi hiển thị một khối “try-except”, trong đó chúng ta mở một tập tin để đọc, đọc một dòng từ tập tin này và chuyển đổi dòng này thành một số nguyên. Có ít nhất hai trường hợp ngoại lệ có thể: IOError hoặc ValueError.

```
f = open('integers.txt', 'w')
f.write('12 345 cuong\n')
f.close()
try:
    f = open('integers.txt')
    s = f.readline()
    i = int(s.strip())
except (IOError, ValueError):
    print "An I/O error or a ValueError occurred"
except:
    print "An unexpected error occurred"
    raise


Output:


An I/O error or a ValueError occurred
```

Clean-up (try-finally)

Cho đến nay, câu lệnh “try” đã luôn được kết hợp với “except”. Nhưng cũng có một cách khác để sử dụng nó. Câu lệnh “try” có thể được theo sau bởi một “finally”. “finally” được gọi là clean-up action hoặc kết thúc (termination) vì chúng phải được thực hiện trong mọi trường hợp, nghĩa là mệnh đề "finally " luôn được thực hiện bất kể trường hợp ngoại lệ xảy ra hay không trong khối “try”.

Một ví dụ đơn giản để chứng minh nó:

```
def test_try_finall():
    try:
        x = float(raw_input("Your number: "))
        inverse = 1.0 / x
    finally:
         print("There may or may not have been an exception.")

test_try_finall()
test_try_finall()

Output:
```

```
Your number: 23

There may or may not have been an exception.

Your number: 0

Traceback (most recent call last):

  File "C:/Users/cuongtran/PycharmProjects/BigData/course.py", line 10, in <module>

    test_try_finall()

  File "C:/Users/cuongtran/PycharmProjects/BigData/course.py", line 4, in test_try_finall

    inverse = 1.0 / x

ZeroDivisionError: float division by zero

There may or may not have been an exception.
```

Nhìn vào kết quả, dù có xẩy ra ngoại lệ hay không thì đoạn mã print("There may or may not have been an exception.") luôn được thực hiện.

Kết hợp try-except-finally

Ta viết lại ví dụ phía trên:

```
def test_try_except_finall():
    try:
        x = float(raw_input("Your number: "))
        inverse = 1.0 / x
    except ValueError:
        print "You should have given either an int or a float"
    except ZeroDivisionError:
        print "Infinity"
    finally:
        print("There may or may not have been an exception.")

test_try_except_finall()
test_try_except_finall()


Output:


Your number: 23

There may or may not have been an exception.

Your number: 0

Infinity

There may or may not have been an exception.
```

Mệnh đề “else” trong xử lý ngoại lệ:

“try-except” có một mệnh đề tùy chọn khác. Một khối khác phải được đặt sau tất cả các mệnh đề ngoại lệ. Một mệnh đề “else” sẽ được thực hiện nếu mệnh đề “try“ không đưa ra ngoại lệ nào.

Ví dụ sau mở tệp và đọc tất cả các dòng vào một danh sách có tên "văn bản":

Ta so sánh đoạn mã sau: inverse chỉ được in ra nếu không có lỗi nào xẩy ra.

```
def test_try_else():
    try:
        x = float(raw_input("Your number: "))
        inverse = 1.0 / x
    except ValueError:
        print "You should have given either an int or a float"
    except ZeroDivisionError:
        print "Infinity"
    else:
        print "inverse = ", inverse

test_try_else()
test_try_else()

Output:

Your number: 23

inverse =  0.0434782608696

Your number: 0

Infinity
```

Dùng mệnh đề “assert”.

Thường assert dùng để phục vụ testing hoặc debug.

Và nó chỉ bắn ra ngoại lệ nếu như điều kiện là False

Chúng ta có thể mô tả nó như sau trong Python.

if not <some_test>:

raise AssertionError(<message>)

và cách dùng tương đương với mệnh đề “assert” là:

assert <some_test>, <message>

ví dụ:

```
x = 5
y = 10
assert x > y, "x has to be greater than y"


Traceback (most recent call last):

  File "C:/Users/cuongtran/PycharmProjects/BigData/course.py", line 3, in <module>

    assert x > y, "x has to be greater than y"

AssertionError: x has to be greater than y
```

Kết luận

Ngoại lệ là không thể tránh khỏi khi chúng ta lập trình. Chúng không phải là những lỗi xảy ra thường xuyên và nằm ngoài các quy tắc thông thường. Nhiều ngôn ngữ lập trình hỗ trợ xử lý lỗi ngoại lệ. Trong Python thì việc xử lý ngoại lệ khá giống với Java. Đoạn mã có khả năng xảy ra ngoại lệ sẽ được đặt trong khối “try-except” hoặc “try-except-finally”. Ngoại lệ sẽ được xử lý bằng đoạn mã nằm trong “except”. Chúng ta có thể khai báo một tuple các ngoại lệ nếu chúng được xử lý với cùng một đoạn mã, hoặc tự tạo một custom exception. Ngoài ra Python còn hỗ trợ một mệnh đề tùy chọn khác là “try-except-else”. Mệnh đề else sẽ được thực hiện nếu mệnh đề try không đưa ra ngoại lệ nào.

Như vậy chúng tôi đã giới thiệu một loạt những khái niệm và phương thức cơ bản trong Python kể từ đầu khóa học đến giờ. Đó là những kiến thức cốt lõi để các bạn bước đầu tiếp cận với những nội dung mới hơn ở các bài học sau về Data Analysis cũng như các bộ thư viện mạnh mẽ trong Python.