LẬP TRÌNH PYTHON

Bài 8: Ngoại lệ và xử lý ngoại lệ

Tóm tắt nội dung bài trước

- Kiểu dữ liệu từ điển là kiểu dữ liệu lấy cảm hứng từ từ điển trong cuộc sống
  - Từ điển là một tập các mục, một mục là một cặp key và value
  - Các key phải đôi một khác nhau, phải là kiểu dữ liệu bất biến
  - Có thể tra cứu value thông qua phép toán chỉ mục với key
  - Dữ liệu trong từ điển không có tính thứ tự, nhưng có thể duyệt bằng vòng lặp
- Module và Package là cơ chế Python sử dụng để kiểm soát mã nguồn hiệu quả hơn
  - Một file mã nguồn = một module, có thể tái sử dụng bởi import
  - Một thư mục chứa mã nguồn = một package
- Module math chứa khá nhiều các hàm toán học cơ bản

Nội dung

1. Ngoại lệ là gì?
2. Xử lý ngoại lệ
3. Một số loại ngoại lệ thường gặp
4. Tự sinh ngoại lệ
5. Bài tập

Phần 1

Ngoại lệ là gì?

Ngoại lệ là gì?

- Việc một chương trình máy tính hoạt động không hoàn hảo là không thể tránh khỏi
- Thường thì giới lập trình chia lỗi thành 3 nhóm
  1. Lỗi khi viết chương trình: hệ quả là chương trình không chạy được khi gặp dòng lệnh sai, nếu là thông dịch (hoặc không dịch được, nếu là biên dịch)
  2. Lỗi khi chương trình chạy: hệ quả là phải thực hiện lại
     - Chẳng hạn như nhập liệu không đúng, thì phải nhập lại
  3. Ngoại lệ: vẫn là lỗi, xảy ra khi có một bất thường và khiến một chức năng không thể thực hiện được
     - Chẳng hạn như đang ghi dữ liệu ra một file, nhưng file đó lại bị một tiến trình khác xóa mất
- Ngoại lệ = lỗi? Đúng, nhưng không hẳn

Ngoại lệ là gì?

- Ranh giới giữa ngoại lệ và lỗi khá mong manh, thậm chí khó phân biệt trong nhiều tình huống
- Vấn đề: Cách chia lỗi thành 3 nhóm có khuynh hướng cho rằng môi trường thực thi của chương trình là thân thiện và hoàn hảo
- Python quan điểm nên chia lỗi thành 2 loại
  - Syntax error: viết sai cú pháp, khiến chương trình thông dịch không dịch được, trong trường hợp này lập trình viên phải viết lại mã, không có phương án nào khác
  - Exception: xảy ra bất thường không như thiết kế
    - Như vậy xử lý exception sẽ khiến chương trình ổn định và hoạt động tốt trong mọi tình huống
    - Trường hợp này lập trình viên phải có phương án khắc phục

Ngoại lệ là gì?

- Ví dụ về syntax error:

```
>>> while True print('Hello world')
  File "<stdin>", line 1
    while True print('Hello world')
                     ^
SyntaxError: invalid syntax
```

- Ví dụ về exception:

```
>>> 10 * (1/0)
Traceback (most recent call last):
  File "<stdin>", line 1, in <module>
ZeroDivisionError: division by zero
```

- Có vẻ như syntax error cũng chỉ là một exception!!!

Phần 2

Xử lý ngoại lệ

“xử lý” ngoại lệ

```
while True:
    try:
        x = int(input("Nhập số X: "))
        break
    except ValueError:
        print("Lỗi, hãy nhập lại.")
print("X =", x)
```

Vòng lặp nhập X cho đến khi người dùng nhập vào đúng giá trị số

Khối nhập X (có thể nhập lỗi)

Xử lý khi lỗi xảy ra

Cú pháp try-except-else-finally

- Cú pháp:

```
try:
except:
else:
finally:
```

- Công việc của từng khối:
  - Khối "try": đoạn mã có khả năng gây lỗi, khi lỗi xảy ra, khối này sẽ bị dừng ở dòng gây lỗi
  - Khối "except": đoạn mã xử lý lỗi, chỉ thực hiện nếu có lỗi xảy ra
  - Khối "else": có thể xuất hiện ngay sau khối except cuối cùng, đoạn mã sẽ được thực hiện nếu không có except nào được thực hiện (đoạn try không có lỗi)
  - Khối "finally": còn được gọi là khối clean-up, luôn được thực hiện dù có xảy ra lỗi hay không

Cú pháp try-except-finally

- Chú ý:
  - Khối try chỉ có 1 khối duy nhất, phải viết đầu tiên
  - Khối finally có thể có hay không, nếu có thì khối này phải viết cuối cùng
  - Khối except có thể không viết, có một khối, hoặc nhiều khối except khác nhau (để xử lý nhiều tình huống lỗi khác nhau)
  - Một khối except có thể xử lý một loại lỗi, nhiều loại lỗi hoặc tất cả các loại lỗi
  - Nếu không xử lý triệt để lỗi có thể "ném" trả lại lỗi này bằng lệnh "raise"
  - Có thể phát sinh một ngoại lệ bằng lệnh "raise <lỗi>"
  - Khi lỗi xảy ra, một biến chứa lỗi được phát sinh và "ném" xuống phần các khối except, khối except đầu tiên "bắt" được ngoại lệ sẽ xử lý nó

Quy tắc "bắt" ngoại lệ: lọt sàng xuống nia

```
except (NameError, TypeError):    # xử lý 2 loại lỗi
    print("Name or Type error")
except IOError as e:              # lấy biến lỗi, đặt tên là e
    print(e)
    raise                         # "ném" trả lại lỗi này
except ValueError:                # xử lý lỗi Value
    print("Value error")
except:                           # xử lý mọi lỗi khác
    print("An error occurred")
    raise NameError("Ko bit")     # tạo ra một lỗi "Ko bit"
else:                             # thực hiện nếu không có lỗi
    print("OK")
```

Nhiệm vụ của khối finally: xử lý dự phòng

- Việc chấm dứt đoạn mã một cách bất ngờ (trong khối try hoặc except) đem lại một số vấn đề
  - Chẳng hạn như trong khối try ta tạo một tập tin tạm thời để chứa dữ liệu trung gian, dùng xong sẽ xóa tập tin đó đi
  - Khi đoạn mã chấm dứt đột ngột vì ngoại lệ: lệnh xóa tập tin tạm không thực hiện
  - Hậu quả: sau một thời gian sử dụng chương trình, máy tính của người dùng có xuất hiện rất nhiều tập tin "rác"
  - Thực tế vấn đề này đã xảy ra với phần mềm Microsoft Word
- Khối finally: còn được gọi là khối "dọn dẹp" (clean-up)
  - Python đảm bảo rằng đoạn mã trong khối luôn được thực hiện dù có ngoại lệ xảy ra hay không
  - Trong ví dụ trên: ta sẽ đặt phần mã xóa tập tin tạm vào finally

Phần 3

Một số loại ngoại lệ thường gặp

Một số loại ngoại lệ thường gặp

Ngoại lệ	Lý do gây ra
SystemExit	Xảy ra khi chương trình gọi hàm **sys.exit()**
KeyboardInterrupt	Xảy ra khi người dùng nhấn phím ngắt để cố gắng kết thúc chương trình (thường là Ctrl-C hoặc Delete)
GeneratorExit	Xảy ra khi chương trình đóng một bộ sinh (generator) bằng hàm **close()** Chú ý: đây không thực sự là một lỗi
Exception	Lớp cha của mọi lớp lỗi, ngoại trừ **KeyboardInterrupt**, **SystemExit** và **GeneratorExit**
StopIteration	Xảy ra khi cố gắng đọc đối tượng tiếp theo từ iterator, nhưng khi đó iterator đã ở cuối của kiểu tuần tự nên không có đối tượng cần đọc
StopAsyncIteration	Xảy ra khi cố gắng đọc đối tượng tiếp theo từ một iterator bất đồng bộ, nhưng khi đó iterator chưa đồng bộ được đối tượng tiếp theo

Một số loại ngoại lệ thường gặp

Ngoại lệ	Lý do gây ra
ArithmeticError	Lớp cha của mọi lỗi xử lý tính toán
FloatingPointError	Lớp con của **ArithmeticError** *(hiện đã bỏ không dùng, chỉ giữ lại để tương thích với các mã cũ)*
OverflowError	Lớp con của **ArithmeticError**, xảy ra khi kết quả của phép toán số học quá lớn hoặc quá nhỏ đến mức không thể biểu diễn
ZeroDivisionError	Lớp con của **ArithmeticError**, xảy ra khi thực hiện phép chia hoặc phép đồng dư với số 0
AssertionError	Xảy ra khi câu lệnh kiểm tra điều kiện **assert** thất bại
AttributeError	Xảy ra khi không thể gán thuộc tính hoặc tham chiếu
BufferError	Xảy ra khi không thể tạo vùng đệm hoặc bị tràn vùng đệm

Một số loại ngoại lệ thường gặp

Ngoại lệ	Lý do gây ra
EOFError	Xảy ra khi cố gắng đọc thêm dữ liệu mặc dù đã đến cuối tập tin (không thể đọc thêm được gì nữa)
ImportError	Xảy ra khi chương trình khai báo sử dụng thành phần trong thư viện (một hàm, một biến hoặc một lớp) nhưng không tìm thấy thành phần này
ModuleNotFoundError	Kiểu con của **ImportError**, xảy ra khi chương trình khai báo sử dụng một thư viện nhưng Python không thể nạp được thư viện đó trên máy hiện tại (thường do không tìm thấy mã máy của thư viện đó)
LookupError	Lớp cha của các lỗi xảy ra khi tìm kiếm dữ liệu
IndexError	Kiểu con của **LookupError**, xảy ra khi giá trị chỉ mục (index) nằm ngoài phạm vi của một biến kiểu tuần tự (string, list, tuple,…)
KeyError	Kiểu con của **LookupError**, xảy ra khi không tìm thấy khóa trong từ điển

Một số loại ngoại lệ thường gặp

Ngoại lệ	Lý do gây ra
MemoryError	Xảy ra khi hết bộ nhớ nên chương trình không thể tiếp tục thực thi hoặc bộ nhớ bị phân mảnh đến mức không thể tạo được các biến cần thiết để tiếp tục thực thi chương trình
NameError	Xảy ra khi cố gắng truy cập một biến không tồn tại
UnboundLocalError	Kiểu con của **NameError**, xảy ra khi tham chiếu đến một biến cục bộ nhưng biến cục bộ đó không tồn tại
ReferenceError	Xảy ra khi tham chiếu bị lỗi, chẳng hạn như truy cập một biến hoặc một thuộc tính đã bị bộ dọn rác (garbage collector) xóa đi
RuntimeError	Xảy ra một lỗi thực thi chung chung, không rơi vào nhóm các lỗi đã biết
NotImplementedError	Kiểu con của **RuntimeError**, xảy ra khi chương trình cố gắng thực thi một phương thức hoặc một hàm, nhưng vì lý do nào đó phần thân hàm (phương thức) chưa được viết

Một số loại ngoại lệ thường gặp

Ngoại lệ	Lý do gây ra

OSError	Xảy ra khi thực hiện một hàm liên quan đến hệ thống (chẳng hạn như đọc ghi đĩa) nhưng gặp lỗi.
	
	*Để giữ tương thích với các phiên bản trước, lỗi này còn có các tên khác như **IOError**, **EnvironmentError** và **WindowsError***
	
	Đây là một lớp lớn cung cấp nhiều thông tin lỗi khi làm việc với hệ điều hành, các lớp con của lớp này gồm:
	
	- **BlockingIOError**
	- **ChildProcessError**
	- **ConnectionError**
	  ○ **BrokenPipeError**
	  ○ **ConnectionAbortedError**
	  ○ **ConnectionRefusedError**
	  ○ **ConnectionResetError**
	- **FileExistsError**
	- **FileNotFoundError**
	- **InterruptedError**
	- **IsADirectoryError**
	- **NotADirectoryError**
	- **PermissionError**
	- **ProcessLookupError**
	- **TimeoutError**

Một số loại ngoại lệ thường gặp

Ngoại lệ	Lý do gây ra
RecursionError	Kiểu con của **RuntimeError**, xảy ra khi gọi đệ quy quá nhiều lớp (độ sâu quá lớn)
SyntaxError	Xảy ra khi cố gắng chạy một lệnh viết sai cú pháp
IndentationError	Kiểu con của **SyntaxError**, xảy ra khi cố gắng chạy một lệnh viết thụt lề không chính xác
TabError	Kiểu con của **IndentationError**, xảy ra khi thụt lề sử dụng các dấu tab và dấu space không nhất quán
SystemError	Xảy ra khi trình thông dịch gặp các lỗi nội bộ
TypeError	Xảy ra khi chương trình cố gắng chuyển một đối tượng sang một kiểu khác nhưng không phù hợp (chẳng hạn cố gắng đổi một chuỗi tên riêng sang dạng số nguyên)

Một số loại ngoại lệ thường gặp

Ngoại lệ	Lý do gây ra
ValueError	Xảy ra khi hàm hoặc phương thức hoặc phép toán nhận được một đối số có kiểu đúng nhưng giá trị không phù hợp
UnicodeError	Kiểu con của **ValueError**, xảy ra khi có lỗi liên quan đến quá trình xử lý dữ liệu unicode
UnicodeEncodeError	Kiểu con của **UnicodeError**, xảy ra khi lỗi liên quan đến quá trình mã hóa dữ liệu
UnicodeDecodeError	Kiểu con của **UnicodeError**, xảy ra khi lỗi liên quan đến quá trình giải mã dữ liệu
UnicodeTranslateError	Kiểu con của **UnicodeError**, xảy ra khi lỗi liên quan đến quá trình chuyển đổi dữ liệu (kể cả chuyển đổi code page)

Một số loại ngoại lệ thường gặp

Ngoại lệ	Lý do gây ra
Warning	Lớp cha của các loại cảnh báo, những lỗi nhẹ hoặc có tiềm năng xảy ra trong tương lai. Các lớp con của lớp này gồm: - **UserWarning** - **DeprecationWarning** - **PendingDeprecationWarning** - **SyntaxWarning** - **RuntimeWarning** - **FutureWarning** - **ImportWarning** - **UnicodeWarning** - **BytesWarning** - **ResourceWarning**

Phần 4

Tự sinh ngoại lệ

Từ khóa raise

- Python cung cấp từ khóa raise sử dụng khi cần phát sinh một ngoại lệ
  - Nếu chỉ viết "raise": cách viết này chỉ đúng trong khối except, khi ta không xử lý được ngoại lệ hiện tại và "ném trả" ngoại lệ về cho chương trình cha
  - Nếu viết "raise <biến>": phát sinh một ngoại lệ và <biến> sẽ chứa các thông tin báo lỗi về ngoại lệ xảy ra
    - Trong trường hợp này, <biến> nên có kiểu Exception hoặc kế thừa từ Exception, kiểu càng cụ thể thì càng cung cấp nhiều thông tin cho quá trình sửa lỗi
  - Lập trình viên có thể tạo một kiểu ngoại lệ mới, kĩ thuật này sử dụng hướng đối tượng là chủ đề nằm ngoài bài giảng này nên sẽ không được đề cập tới

Ví dụ về phát sinh ngoại lệ

```
try:
    a = int(input("Nhập một số nguyên dương nhỏ hơn 100: "))
    # sinh lỗi khi số quá bé
    if a <= 0:
        raise ValueError("Bạn đã nhập một số quá nhỏ")
    # sinh lỗi khi số quá lớn
    if a >= 100:
        raise ValueError("Bạn cần nhập số nhỏ hơn 100")
# bắt lỗi:
# - nhập không phải số nguyên
# - nhập số quá lớn
# - nhập số quá bé
except ValueError as ex:
    print(ex)
```

Phần 5

Bài tập

Bài tập

1. Viết chương trình nhập 2 số nguyên a và b, sau đó tính và in ra giá trị phân số a/b. Chú ý xử lý ngoại lệ trong các tình huống dưới đây:
   - Người dùng nhập a hoặc b không phải số nguyên
   - Người dùng nhập b = 0
2. Viết chương trình yêu cầu người dùng nhập a, b và c là độ dài 3 cạnh của một tam giác. Xử lý ngoại lệ trong các tình huống sau:
   - Người dùng nhập a, b hoặc c không phải là kiểu số
   - Người dùng nhập giá trị 0 hoặc số âm cho a, b hoặc c
   - Người dùng nhập a, b, c dương nhưng không thể là cạnh của một tam giác

Bài tập

3. Viết chương trình cho phép người dùng nhập liên tiếp các số nguyên dương, yêu cầu:
   - Việc nhập liệu sẽ kết thúc bình thường (không exception) khi người dùng nhập số 0
   - Phát sinh exception và thông báo "Lỗi số âm!!!" khi người dùng nhập một số nguyên âm bất kỳ
   - Phát sinh exception và thông báo "Lỗi nhập số" khi người dùng nhập vào chuỗi không phải số nguyên
   - Phát sinh exception và thông báo "Lỗi nhập lặp" khi người dùng nhập 4 số nguyên dương liên tiếp giống nhau
   - Phát sinh exception và thông báo "Lỗi nhập chẵn" khi người dùng nhập 5 số chẵn liên tiếp
   - Sau khi phát sinh exception, xử lý và thông báo, chương trình cần tiếp tục hoạt động như bình thường