



Trường ĐH Khoa Học Tự Nhiên Tp. Hồ Chí Minh

TRUNG TÂM TIN HỌC



Lập trình Python cơ bản

Bài 10: Xử lý ngoại lệ (Exception)



Phòng Lập Trình - Mạng

2023

Nội dung

1. Lỗi cú pháp (Syntax Error)
2. Lỗi ngoại lệ (Exception Error)
3. Assertions
4. Các exceptions có sẵn trong Python (*Standard Exceptions*)
5. Exception do người dùng tự định nghĩa (*User Defined Exception*)

1. Lỗi cú pháp (Syntax Error)

- ☐ **Lỗi cú pháp (Syntax Error):** Là các lỗi lập trình khi chương trình bị viết sai cú pháp quy định. Lỗi này còn biết đến như lỗi phân tích (*parsing error*).

```
n = int(input('Nhập cùu chương muốn in: '))
for i in range(1, 11)
    print('%i x %2i = %2i' % (n, i, n * i))
```

```
File "k:/GIANG_DAY/_PORTAL/Chuyen_de/LDS1_2022/Demo/x1_ngoaile.py", line 4
for i in range(1, 11)
^
```

```
SyntaxError: invalid syntax
```

1. Lỗi cú pháp (Syntax Error)

```
n = int(input('Nhập cửu chương muốn in: '))
for i in range(1, 11)
    print('%i x %2i = %2i' % (n, i, n * i))

File "k:/GIANG_DAY/_PORTAL/Chuyen_de/LDS1_2022/Demo/xl Ngoai_le.py", line 4
    for i in range(1, 11)
               ^
SyntaxError: invalid syntax
```

- Bộ phân tích lặp lại dòng gây lỗi và hiển thị một mũi tên nhỏ (^) trỏ vào điểm đầu tiên lỗi được phát hiện (lỗi nằm ở phía sau dấu hiệu mũi tên).
- Trong ví dụ trên, lỗi được phát hiện tại dòng lệnh for, ngay sau hàm range(), vì thiếu một dấu hai chấm (":"). Python hiển thị Tên tập tin python và số thứ tự của dòng xảy ra lỗi để hỗ trợ trong việc tìm lỗi.

Nội dung

1. Lỗi cú pháp (Syntax Error)
2. Lỗi ngoại lệ (Exception Error)
3. Assertions
4. Các exceptions có sẵn trong Python (*Standard Exceptions*)
5. Exception do người dùng tự định nghĩa (*User Defined Exception*)

2. Lỗi ngoại lệ (Exception Error)

- ❑ Khi có phát sinh 1 exception và exception này không (hoặc chưa) được xử lý, chương trình sẽ bị ngắt tại nơi gây ra exception. Ngược lại nếu đã được người lập trình xử lý thì khi exception phát sinh, chương trình sẽ xử lý theo hướng của người lập trình đã cài đặt sẵn.
- ❑ Ngoài tính năng debug, Python cung cấp hai tính năng rất quan trọng để xử lý lỗi trong chương trình là
 - Assertions
 - Exception Handling

Nội dung

1. Lỗi cú pháp (Syntax Error)
2. Lỗi ngoại lệ (Exception Error)
3. Assertions
4. Các exceptions có sẵn trong Python (*Standard Exceptions*)
5. Exception do người dùng tự định nghĩa (*User Defined Exception*)

3. Assertions

- ❑ Cú pháp: **assert(Expression), [, Arguments]** thông_báo_lỗi
- ❑ Giải thích:
 - Khi thấy một *assertion statement*, Python đánh giá các *expression* kèm theo. Nếu *expression* có kết quả là:
 - **True**: lệnh đi ngay sau lệnh assert sẽ tiếp tục được thực hiện.
 - **False**: Python sẽ phát ra một *AssertionError Exception* bằng cách dùng *ArgumentExpression* làm đối số cho *AssertsError*.

3. Assertions

- ❑ Cú pháp: **assert(Expression), [, Arguments]** thông_báo_lỗi
- ❑ Giải thích:
 - Các ngoại lệ của *AssertionError* có thể được bắt và xử lý như bất kỳ ngoại lệ nào khác bằng cách sử dụng câu lệnh *try-except*, nhưng nếu không được xử lý, chúng sẽ chấm dứt chương trình và tạo ra một *traceback*.
 - Về cơ bản, *traceback* được sử dụng để xuất dấu vết của một chương trình sau khi một exception xảy ra. *traceback* bao gồm thông báo lỗi, số dòng gây ra lỗi và *call stack* của *function* gây ra lỗi.

3. Assertions

- ❑ Cú pháp: **assert(Expression), [, Arguments]** thông_báo_lỗi

- ❑ Ví dụ:

```
def uscln(a, b):
    assert (a > 0 and b > 0), 'a và b phải lớn hơn 0'
    while a != b:
        if a > b:
            a -= b
        else:
            b -= a
    return a

x = 9
y = 0
print(uscln(x, y))
```

```
Traceback (most recent call last):
  File "k:/GIANG_DAY/_PORTAL/Chuyen_de/LDS1_2022/Demo/:
    print(uscln(x, y)) # 3
  File "k:/GIANG_DAY/_PORTAL/Chuyen_de/LDS1_2022/Demo/:
    assert (a > 0 and b > 0), 'a và b phải lớn hơn 0'
AssertionError: a và b phải lớn hơn 0
```

Nội dung

1. Lỗi cú pháp (Syntax Error)
2. Lỗi ngoại lệ (Exception Error)
3. Assertions
4. Các exceptions có sẵn trong Python (Standard Exceptions)
5. Exception do người dùng tự định nghĩa (User Defined Exception)

4. Các exceptions có sẵn trong Python

❑ Các exceptions có sẵn trong Python:

Stt	Exception	Mô tả
1	Exception	Đây là lớp cơ sở (base class) cho tất cả các exception. Exception này sẽ xuất hiện khi có bất cứ một lỗi nào xảy ra.
2	StopIteration	Xuất hiện khi phương thức next() của iterator không trả đến một đối tượng nào.
3	SystemExit	Xuất hiện khi dùng phương thức sys.exit()
4	StandardError	Lớp cơ sở cho tất cả các exception, ngoại trừ StopIteration và SystemExit.
5	ArithmetricError	Lớp cơ sở cho tất cả các lỗi xảy ra khi tính toán các số.
6	OverflowError	Xuất hiện khi thực hiện tính toán và giá trị tính toán vượt quá ngưỡng giới hạn cho phép của kiểu dữ liệu.

4. Các exceptions có sẵn trong Python

❑ Các exceptions có sẵn trong Python:

Số thứ tự	Exception	Mô tả
7	FloatingPointError	Xuất hiện khi tính toán các số kiểu float thất bại.
8	ZeroDivisionError	Xuất hiện khi thực hiện phép chia hoặc chia lấy dư một số cho 0.
9	AssertionError	Xuất hiện trong trường hợp lệnh assert thất bại.
10	AttributeError	Xuất hiện khi không tồn tại thuộc tính hoặc thiếu tham số truyền vào cho thuộc tính.
11	EOFError	Xuất hiện khi không có dữ liệu từ hàm input() hoặc raw_input() hay lỗi do thao tác trên file khi con trỏ đang ở cuối file.
12	ImportError	Xuất hiện khi lệnh import thất bại.

4. Các exceptions có sẵn trong Python

❑ Các exceptions có sẵn trong Python:

Stt	Exception	Mô tả
13	KeyboardInterrupt	Xuất hiện khi người dùng gián đoạn việc thực hiện chương trình, thường bằng cách nhấn Ctrl + C.
14	LookupError	Lớp cơ sở cho tất cả các lỗi về lookup.
15	IndexError	Xuất hiện khi index không tồn tại trong list, string,...
16	KeyError	Xuất hiện khi key không tồn tại trong dictionary.
17	NameError	Xuất hiện khi một biến không tồn tại trong phạm vi chương trình gọi biến đó.
18	UnboundLocalError	Xảy ra khi cố gắng truy cập một biến cục bộ trong một hàm/phương thức nhưng không có giá trị nào được gán cho nó.

4. Các exceptions có sẵn trong Python

❑ Các exceptions có sẵn trong Python:

Số thứ tự	Exception	Mô tả
19	EnvironmentError	Là lớp cơ sở cho tất cả các exception về lỗi khi có bất kỳ một lỗi nào ngoài phạm vi của Python.
20	IOError	Xuất hiện khi sử dụng input/output thất bại, hoặc mở file không thành công (không tồn tại).
21	OSError	Xuất hiện khi có lỗi từ hệ điều hành.
22	SyntaxError	Xuất hiện khi chương trình có lỗi cú pháp.
23	IndentationError	Xuất hiện khi có lệnh thụt đầu dòng không đúng.
24	SystemError	Xuất hiện khi trình thông dịch phát hiện có vấn đề, nhưng lúc này trình thông dịch của Python không tự thoát (kết thúc) được.

4. Các exceptions có sẵn trong Python

❑ Các exceptions có sẵn trong Python:

Số thứ tự	Exception	Mô tả
25	SystemExit	Xuất hiện khi trong code không sử dụng hàm sys.exit() nhưng trình thông dịch của Python vẫn được thoát bằng hàm sys.exit().
26	TypeError	Xuất hiện khi thực thi toán tử hoặc hàm mà kiểu dữ liệu bị sai so với kiểu dữ liệu đã định nghĩa ban đầu
27	ValueError	Xuất hiện khi chúng ta build 1 function mà kiểu dữ liệu đúng nhưng khi chúng ta thiết lập ở tham số là khác so với khi truyền vào.
28	RuntimeError	Xuất hiện khi lỗi được sinh ra không thuộc một danh mục nào.
29	NotImplementedError	Xuất hiện khi một phương thức trừu tượng cần được thực hiện trong lớp kế thừa chứ không phải là lớp thực thi.

4. Các exceptions có sẵn trong Python

- Cú pháp: **try...except...else**

```
try:  
    # khối lệnh có khả năng xảy ra lỗi  
except loại_lỗi_1 as tên_biến_báo_lỗi_1:  
    # in thông báo lỗi  
except loại_lỗi_1 as tên_biến_báo_lỗi_1:  
    # in thông báo lỗi  
...  
else:  
    # khối lệnh khi không có exception nào xảy ra
```

- Lưu ý: có thể sử dụng lệnh *raise* để định nghĩa bổ sung cho *Exception*.

4. Các exceptions có sẵn trong Python

- ❑ Ví dụ 1: Xử lý lỗi chia cho 0 – ZeroDivisionError

```
try:  
    a = eval(input('Nhập a: '))  
    b = eval(input('Nhập b: '))  
    thuong = a / b  
except ZeroDivisionError:  
    print('Lỗi chia cho 0. Nhập giá trị cho b khác 0')  
else:  
    print('Thương = %s / %s = %s' % (a, b, thuong))
```

Nhập a: 12
Nhập b: 2
Thương = 12 / 2 = 6.0

Nhập a: 12
Nhập b: 0
Lỗi chia cho 0. Nhập giá trị cho b khác 0

4. Các exceptions có sẵn trong Python

- ❑ Ví dụ 2: Kết hợp `ZeroDivisionError` và `NameError`

```
try:  
    a = eval(input('Nhập a: '))  
    b = eval(input('Nhập b: '))  
    thuong = a / b  
except (ZeroDivisionError, NameError) as err:  
    print('Error:', err)  
else:  
    print('Thương = %s / %s = %s' % (a, b, thuong))
```

Nhập a: 12
Nhập b: 2
Thương = 12 / 2 = 6.0

Nhập a: 12
Nhập b: 0
Error: division by zero

Nhập a: 12
Nhập b: c
Error: name 'c' is not defined

4. Các exceptions có sẵn trong Python

- ❑ Ví dụ 3: Kết hợp `raise` với `ZeroDivisionError` và `NameError`

```
try:  
    a = eval(input('Nhập a: '))  
    b = eval(input('Nhập b: '))  
    if b == 0:  
        raise ZeroDivisionError('Lỗi chia cho 0. Nhập giá trị cho b khác 0')  
    thuong = a / b  
except ZeroDivisionError as err:  
    print('Error:', err)  
except NameError as err:  
    print('Error:', err)  
else:  
    print('Thương = %s / %s = %s' % (a, b, thuong))
```

Nhập a: 12
Nhập b: 0
Error: Lỗi chia cho 0. Nhập giá trị cho b khác 0

4. Các exceptions có sẵn trong Python

- ❑ Cú pháp: **try... finally**

try:

 # khối lệnh có khả năng xảy ra lỗi

finally:

 # khối lệnh luôn luôn được thực thi

- ❑ Chức năng: Khi ta muốn thực thi khối lệnh trong *finally* dù cho có lỗi xảy ra trong *try* hay không.

4. Các exceptions có sẵn trong Python

❑ Cú pháp: try... finally

```
try:  
    a = eval(input('Nhập a: '))  
    b = eval(input('Nhập b: '))  
    thuong = a / b  
except ZeroDivisionError as err:  
    print('Error:', err)  
else:  
    print('Thương = %s / %s = %s' % (a, b, thuong))  
finally:  
    print('Tổng = %s + %s = %s' % (a, b, a + b))  
    print('Hiệu = %s - %s = %s' % (a, b, a - b))  
    print('Tích = %s * %s = %s' % (a, b, a * b))
```

Nhập a: 12
Nhập b: 0
Error: division by zero
Tổng = 12 + 0 = 12
Hiệu = 12 - 0 = 12
Tích = 12 * 0 = 0

Nội dung

1. Lỗi cú pháp (Syntax Error)
2. Lỗi ngoại lệ (Exception Error)
3. Assertions
4. Các exceptions có sẵn trong Python (Standard Exceptions)
5. Exception do người dùng tự định nghĩa (User Defined Exception)

5. Exception do người dùng tự định nghĩa

- ❑ Một exception trong Python do người dùng định nghĩa luôn bắt buộc exception này phải kế thừa các lớp thuộc *Standard built-in Exception* trong Python.
- ❑ Để gọi exception do người dùng định nghĩa, cần sử dụng keyword *raise* theo cú pháp sau: `raise exception_name`

5. Exception do người dùng tự định nghĩa

❑ Các bước thực hiện:

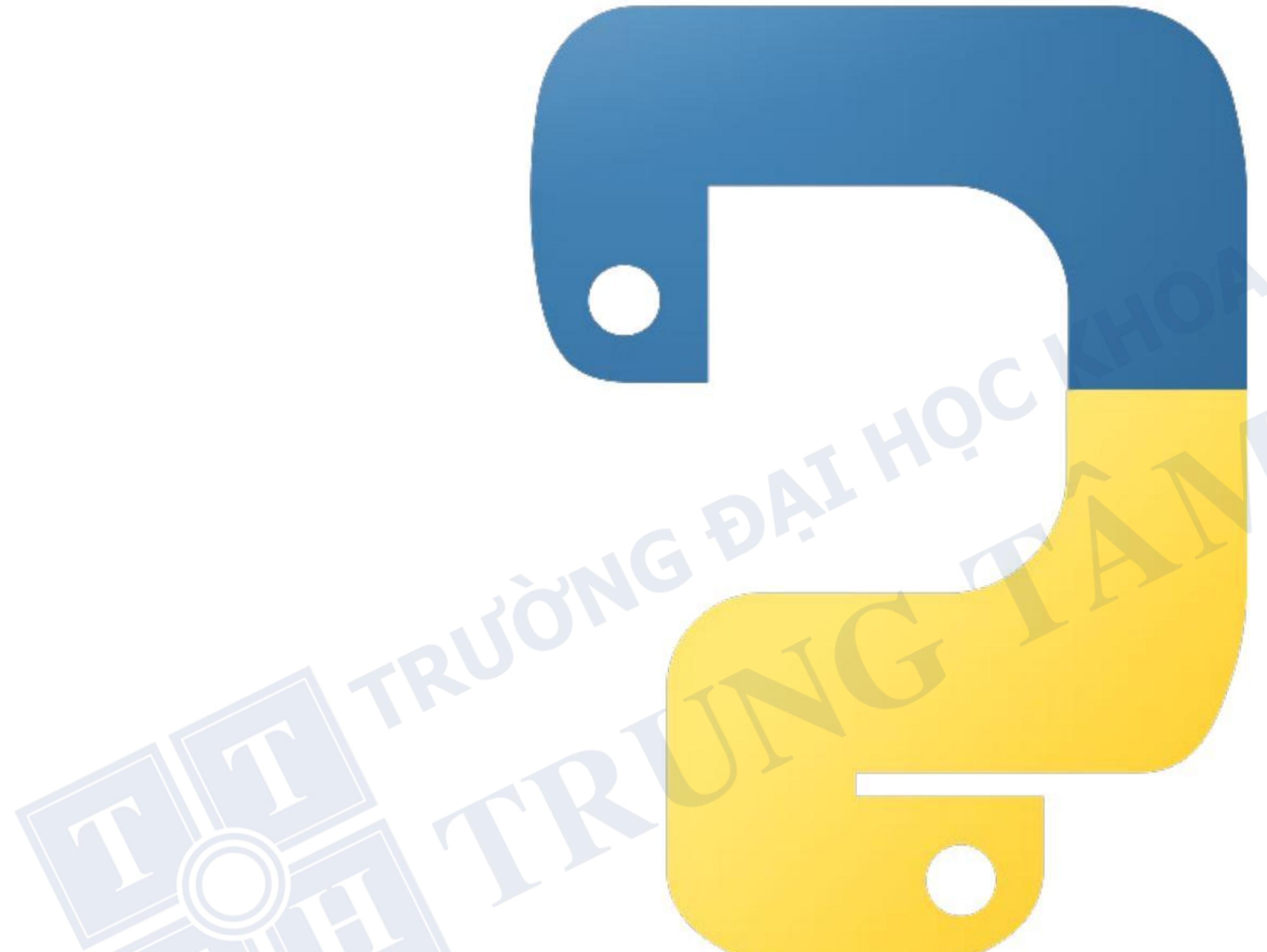
- B1: định nghĩa ra exception của người dùng (*TenDinhNghia* class).
- B2: viết chương trình (hoặc hàm) trong đó gọi *TenDinhNghia* bằng lệnh *raise*.
- B3: từ chương trình chính hoặc hàm khác, gọi hàm *X* và gởi tham số sẽ gây lỗi để biết kết quả.

5. Exception do người dùng tự định nghĩa

```
# B1
class LoichiaCh0(ZeroDivisionError):
    def __init__(self, thong_bao):
        self.thong_bao = thong_bao

# B2
def thuong(a, b):
    try:
        if b == 0:
            raise LoichiaCh0('Lỗi chia cho 0. Nhập giá trị cho b khác 0')
        thuong = a / b
    except LoichiaCh0 as err:
        print('Lỗi:', err)
    else:
        print('Thương = %s / %s = %s' % (a, b, thuong))

# B3
thuong(6, 2) # Thương = 6 / 2 = 3.0
thuong(6, 0) # Lỗi: Lỗi chia cho 0. Nhập giá trị cho b khác 0
```



TRƯỜNG ĐẠI HỌC KHOA HỌC TỰ NHIÊN
TIT TRUNG TÂM TIN HỌC
TTH