



# TIN HỌC CƠ SỞ

## Phần 2: Ngôn ngữ lập trình Python

### CHƯƠNG 14. BẮT LỖI VÀ KIỂM SOÁT LỖI

**Tin cơ sở (LT):** 010100229802 - DHKL16A1HN, - DHKL16A2HN

GV. Cao Diệp Thắng

# Mục tiêu chương

## Nắm vững:

- ☐ Cách xử lý bắt lỗi và kiểm soát lỗi trong python
- ☐ Cách sử dụng một số asent và ngoại lệ chuẩn trong Python.
- ☐ Cách tự sinh ngoại lệ.



# Nội dung

---

- 1 Khái niệm ngoại lệ
- 2 Xử lý ngoại lệ
- 3 Standard exceptions
- 4 Tự xưng ngoại lệ



## 13.1. NGOẠI LỆ

- ❑ Ngoại lệ (Exception)[5,6,7,8,9,10,11] là lỗi xảy ra trong quá trình thực thi một chương trình. Khi nó xảy ra, Python tạo ra một exception để xử lý vấn đề đó tránh cho ứng dụng hay server bị hỏng (crash) hay ngừng hoạt động không đúng cách..
- ❑ Ngoại lệ có thể là bất kỳ điều kiện bất thường nào trong chương trình phá vỡ luồng thực thi chương trình đó. Bất cứ khi nào một ngoại lệ xuất hiện, chương trình sẽ ngừng thực thi, chuyển qua quá trình gọi và in ra lỗi đến khi nó được xử lý.

Thông thường thì người ta 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.



## Quan điểm Python 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, ...
- **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

### Ví dụ 14.1. Ví dụ về syntax error

Lỗi cú pháp thiếu dấu ":"

```
>>> while True print("Hello world")
SyntaxError: invalid syntax
```

### Ví dụ 14.2. Ví dụ về lỗi ngoại lệ (exception)

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

Lỗi chia cho zero



## 14.1.2. Xử lý ngoại lệ

- ❑ Trong Python, các ngoại lệ có thể được xử lý bằng khối lệnh **try...except**. [7, 8, 9, 10] , Phần thân của **try** sẽ gồm code có thể tạo ra exception, nếu một exception được tạo ra, tất cả câu lệnh trong khối sẽ bị bỏ qua.

Phần thân của except được gọi bởi exception handler, vì nó được dùng để bắt lỗi. Khối except sẽ thực hiện khi lỗi được tạo ra, không thì sẽ được bỏ qua. Chúng ta có thể dùng exception có sẵn trong thư viện chuẩn Python.

### Ví dụ 14.3. Ví dụ về syntax error

```
1  while True:
2      try:
3          x=int(input("Nhập số X"))
4          break
5      except ValueError:
6          print("Bị lỗi, hãy nhập lại.")
7  print("X =", x)
```

Khối lệnh nhập X, có thể bị xảy ra lỗi

Xử lý lỗi nếu xảy ra!!

#### Kết quả thực hiện:

Nhap so X: a  
Bi loi, hay nhap lai.  
Nhap so X: u  
Bi loi, hay nhap lai.  
Nhap so X: 4  
X = 4



## 14.1.2. Xử lý ngoại lệ, ....

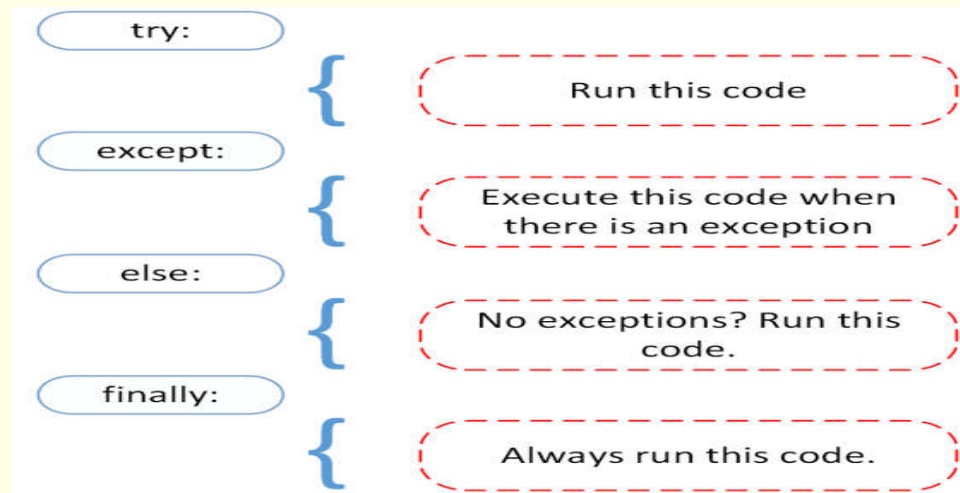
Cú pháp:

```
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

## 14.1.2. Xử lý ngoại lệ, ....



*Công việc của các khối khi xử lý ngoại lệ*

 **Lưu 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

except:                                   # xử lý mọi lỗi khác

    print("An error occurred")
    raise NameError("Ko biet")

else:                                     # tạo ra một lỗi “Ko biet”
    print("OK")                           # thực hiện nếu không có lỗi
```



### 14.1.3. Sử dụng assert

Khi gặp câu lệnh **assert**, Python sẽ đánh giá biểu thức đi kèm, với hy vọng nó là đúng (**True**). Nếu biểu thức là sai (**False**), Python sẽ đưa ra một ngoại lệ **AssertionError**. [9,10,11,12]

**assert Expression** [, Arguments]

Nếu xác nhận không thành công, Python sử dụng **ArgumentExpression** làm đối số cho **AssertionError**. Các ngoại lệ **AssertionError** có thể được bắt và xử lý giống như bất kỳ ngoại lệ nào khác bằng cách sử dụng câu lệnh **try-exception**, nhưng nếu không được xử lý, chúng sẽ chấm dứt chương trình và tạo ra một truy nguyên (traceback).



#### Ví dụ 14.4.

```
1  #Ví dụ về sử dụng asssert
2  def tinhTb(hk1,hk2):
3      [ assert(hk1 >=0 and hk2>=0), 'Cả điểm hk1 và hk2 phải >=0!']
4      tb=(hk1+hk2*2)/3
5      return tb
6  print(tinhTb(8,7.5))
7
8  print(tinhTb(-1,6))
```

Kết quả:

```
7.666666666666667
Traceback (most recent call last):
  File "c:/Users/PC/Dropbox/2022/KHUD_TCS/Chuong_14/vd_14.4.py", line 8, in <module>
    print(tinhTb(-1,6))
  File "c:/Users/PC/Dropbox/2022/KHUD_TCS/Chuong_14/vd_14.4.py", line 3, in tinhTb
    assert(hk1 >=0 and hk2>=0), 'Cả điểm hk1 và hk2 phải >=0!'
AssertionError: Cả điểm hk1 và hk2 phải >=0!
```



## 14.2. STANDARD EXCEPTIONS



### 14.2.1. Danh mục standard exceptions

Exception	Mô tả
Exception	Lớp cơ sở ( <b>base class</b> ) của tất cả các ngoại lệ
ArithmeticError	Lớp cơ sở của tất cả các lỗi xảy ra cho phép tính số học
OverflowError	Được tạo khi một phép tính vượt quá giới hạn tối đa cho một kiểu số
FloatingPointError	Được tạo khi một phép tính số thực thất bại
ZeroDivisonError	Được tạo khi thực hiện phép chia cho số 0 với tất cả kiểu số
AssertionError	Được tạo trong trường hợp lệnh <b>assert</b> thất bại
AttributeError	Được tạo trong trường hợp tham chiếu hoặc gán thuộc tính thất bại
EOFError	Được tạo khi không có input nào từ hàm <b>raw_input()</b> hoặc hàm <b>input()</b> và tới <b>EOF</b> (viết tắt của <b>end of file</b> )
RuntimeError	Được tạo khi một lỗi đã được tạo ra là không trong loại nào
IOError	Được tạo khi hoạt động <b>i/o</b> thất bại, chẳng hạn như lệnh <b>print</b> hoặc hàm <b>open()</b> khi cố gắng mở một file không tồn tại

## 14.2.2. Một số phương pháp sử dụng ngoại lệ chuẩn

### a. Sử dụng `try...except`

Cú pháp:

```
try:
    #Khởi lệnh có khả năng xảy ra lỗi
except [loại lỗi as tên_biến_báo_lỗi]:
    #in thông báo lỗi
```

Ví dụ 14.5.

```
1  x,y = 5,0
2  try:
3      |    print('x/y = ', x/y)
4  except ZeroDivisionError as er:
5      |    print('Error: ',er)
```

Kết quả:

**Error:** division by zero



## b. Sử dụng try...except



Cú pháp: **try:**

```
    #Khởi lệnh có khả năng xảy ra lỗi
except [loại lỗi 1 as tên_biến_báo_lỗi 1]:
    #in thông báo lỗi
except [loại lỗi 2 as tên_biến_báo_lỗi 2]:
    #in thông báo lỗi
...
```

**else:**

# khởi lệnh khi không có exception nào xảy ra

Ví dụ 14.6.

```
1  try:
2      x = eval(input("Nhập x\n"))
3      y = eval(input("nhập y\n"))
4      z=x/y
5  except NameError as er1:
6      print('Error : ', er1)
7  except ZeroDivisionError as er2:
8      print("Error ", er2)
9  else:
10     print('x/y', z)
```

Kết quả:

Chạy lần 1 : Nhập x  
t  
Error : name 't' is not defined

Chạy lần 2 : Nhập x  
6  
nhập y  
0  
Error division by zero

### c. Gộp nhiều lỗi vào chung một exception

Cú pháp:

```
try:
    #Khởi lệnh có khả năng xảy ra lỗi
except [loại lỗi 1 [, loại lỗi 2[,...]]:
    #in thông báo lỗi
else:
    # khởi lệnh khi không có exception nào xảy ra
```

Ví dụ 14.7.

```
1  try:
2      x = eval(input("Nhập x\n"))
3      y = eval(input("nhập y\n"))
4      z=x/y
5  except (NameError, ZeroDivisionError) as er:
6      print('Error : ', er)
7  else:
8      print('x/y', z)
```

#### d. Sử dụng try...finally

- ❑ Khối lệnh **try...finally** được sử dụng khi ta muốn thực thi khối lệnh trong **finally** cho dù lỗi có xảy ra trong **try** hay không.

Cú pháp:

**try:**

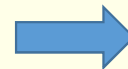
#Khối lệnh có khả năng xảy ra lỗi

**Finally:**

#Khối lệnh luôn luôn được thực thi

Ví dụ 14.8.

```
1  x,y = 5,0
2  try:
3      |   print('x/y', x/y)
4  except ZeroDivisionError as ex:
5      |   print('Error : ', ex)
6  finally:
7      |   print('x+y = ', x+y)
8      |   print('x-y = ', x-y)
9      |   print('x*y = ', x*y)
```



Kết quả:

```
Error:  division by zero
x+y =   5
x-y =   5
x*y =   0
```



## 14.3. TỰ SINH NGOẠI LỆ

*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, kỹ thuật này sử dụng hướng đối tượng là chủ đề nằm ngoài bài giảng này nên sẽ không được đề cập tới

## Ví dụ 14.9.

```
1  try:
2      a= int(input("Nhập một số nguyên dương nhỏ hơn 100: "))
3      #Sinh lỗi khi số quá bé
4      if a <=0:
5          raise ValueError("Bạn đã nhập một số quá nhỏ !")
6      #Sinh lỗi quá lớn
7      if a>=100:
8          raise ValueError("Bạn cần nhập một số nhỏ hơn 100 !")
9      # bắt lỗi:
10     #- Nhập không phải số nguyên
11     #- Nhập số quá lớn
12     #- Nhập số quá bé
13 except ValueError as ex:
14     print(ex)
```

Kết quả lần 1: Nhập một số nguyên dương nhỏ hơn 100: -2  
Bạn đã nhập một số quá nhỏ !

Kết quả lần 2: Nhập một số nguyên dương nhỏ hơn 100: 101  
Bạn cần nhập một số nhỏ hơn 100 !

Kết quả lần 3: Nhập một số nguyên dương nhỏ hơn 100: 50



## Câu hỏi thảo luận

1. Exception là gì?
2. Tại sao cần khi lập trình python?
3. Nêu cách sử dụng khối lệnh **try....expection....finally?**

## Bài tập vận dụng.

Bài tập chương 14. TLHT