Softech®
Gather to thrive together

Aptech™
COMPUTER EDUCATION
Unleash your potential

BÀI 12

Bắt lỗi và kiểm soát lỗi Trong Python

12.1 Tổng quan bắt lỗi và kiểm soát lỗi

Mục tiêu

- Nhận dạng lỗi khi viết và chạy chương trình Python
- Nắm được cách bắt lỗi và kiểm soát lỗi khi viết chương trình

Tại sao lại cần biết ?

- Để biết cách xử lý, gở lỗi khi có lỗi xảy ra
- Dự đoán được lỗi và code làm sao để ít lỗi nhất

Thành công của một lập trình viên là tạo ra một chương trình hoạt động tốt và không có lỗi !

12.2 Bắt lỗi và kiểm soát lỗi

Lỗi Syntax và lỗi logic nội tại

- Lỗi **syntax** là lỗi khi viết sai cú pháp lệnh của ngôn ngữ lập trình. Khi gặp lỗi này Python lập tức dừng chương trình và đưa ra thông báo lỗi **SyntaxError**

```
print(4 => 4)
```

Thông báo lỗi ở terminal

```
SyntaxError: invalid syntax
```

- **Cách sửa:**

  Gặp lỗi này trong VS Code đã báo đỏ phần lỗi sai chỗ nào, và khi chạy lên bạn cũng thấy nó chỉ ra chỗ gây lỗi

12.2 Bắt lỗi và kiểm soát lỗi

Lỗi Exception (ngoại lệ)

Lỗi này khó phát hiện hơn vì do không sai cú pháp mà lỗi phát sinh trong quá trình chạy chương trình. Các lỗi này xảy ra cho nội tại logic của chương trình. ➔ Lỗi Logic

- Lỗi khi nhập liệu
- Lỗi chia cho 0
- Lỗi chỉ số vượt quá giới hạn của dãy
- Lỗi thời gian chạy quá lâu
- Lỗi lời gọi hàm có tham số không đúng kiểu

Với các lỗi này Python có một cơ chế cho phép bạn bắt lỗi và tìm cách sửa lỗi hoặc tránh lỗi

12.2 Bắt lỗi và kiểm soát lỗi

Try ..except…

Ví dụ bạn có một đoạn code sau:

```
try:
  print(x)
except:
  print("An exception occurred")
```

Bắt lỗi dòng lệnh print

Nếu lỗi thì in ra lỗi

- Bạn đưa tất cả dòng lệnh cần check lỗi vào khối lệnh của **try**
- Trường hợp khối lệnh cần bắt lỗi có lỗi thì lỗi ra được in ra ở khối lệnh của **except**

Tất các các lỗi đều được Python đặt tên và gắn mã. Mỗi mà lỗi Exception đều có tên và mô tả lỗi tương ứng.

12.2 Bắt lỗi và kiểm soát lỗi

Cấu trúc bắt lỗi Exception với lệnh try đơn giản

```
try:
    <Lệnh hoặc nhóm lệnh cần bắt lỗi>
except <Mã lỗi Exception 1>
    <Các lệnh xử lý lỗi 1>
....
except <Mã lỗi Exception n>
    <Các lệnh xử lý lỗi n>
```

Cách bắt lỗi này vẫn chưa đầy đủ. Chương tình có thể dừng đột ngột

Trình tự xử lý

- Thực hiện nhóm lệnh cần bắt lỗi
- Nếu không lỗi thì chuyển sang lệnh tiếp theo say try..except
- Nếu có lỗi và lỗi này khớp với các mã lỗi có trong except thì sẽ thực hiện các lệnh xử lý tương ứng với except đó
- Nếu có lỗi và lỗi này không khớp với các mã lỗi except thì dừng chương trình và báo lỗi

12.2 Bắt lỗi và kiểm soát lỗi

Ví dụ

```
import random
try:
  r = random.randint(1,3) # lấy giá trị ngẫu nhiên cho r
  if r == 1:
     print(int("Fred")) # chuyển thành số nguyên sai
  elif r == 3:
     [][2] = 5 # giá giá trị cho index không tồn tại
     print(3/0)
except ValueError:
     print("Không thể chuyển thành số nguyên")
except IndexError:
     print("Chỉ số không tồn tại")
except ZeroDivisionError:
     print("Không thể chia cho 0")
```

12.2 Bắt lỗi và kiểm soát lỗi

Cấu trúc bắt lỗi Exception với lệnh try đầy đủ

```
try:
    <Lệnh hoặc nhóm lệnh cần bắt lỗi>
except <Mã lỗi Exception 1>
   <Các lệnh xử lý lỗi 1>
....
except <Mã lỗi Exception n>
   <Các lệnh xử lý lỗi n>
except Exception:
   <Các lệnh xử lý lỗi khác>
```

Trình tự xử lý

- Thực hiện nhóm lệnh cần bắt lỗi
- Nếu không lỗi thì chuyển sang lệnh tiếp theo say try..except
- Nếu có lỗi và lỗi này khớp với các mã lỗi có trong except thì sẽ thực hiện các lệnh xử lý tương ứng với except đó
- Nếu có lỗi và lỗi này không khớp với các mã lỗi except nằm trên thì rơi vào Xử lý các lỗi khác nằm cuối cùng

12.2 Bắt lỗi và kiểm soát lỗi

Ví dụ trên được sửa lại như sau

```
import random
try:
  r = random.randint(1,3) # lấy giá trị ngẫu nhiên cho r
  if r == 1:
     print(int("Fred")) # chuyển thành số nguyên sai
  elif r == 3:
     [][2] = 5 # giá giá trị cho index không tồn tại
     print(3/0)
except ValueError:
     print("Không thể chuyển thành số nguyên")
except IndexError:
     print("Chỉ số không tồn tại")
except ZeroDivisionError:
     print("Không thể chia cho 0")
except Exception:
     print("Lỗi khác")
```

16.2 Bắt lỗi và kiểm soát lỗi

Nhóm lệnh else và finally

```
try:
    <Lệnh hoặc nhóm lệnh cần bắt lỗi>
except <Mã lỗi Exception 1>
    <Các lệnh xử lý lỗi 1>
except <Mã lỗi Exception n>
    <Các lệnh xử lý lỗi n>
except Exception:
    <Các lệnh xử lý lỗi khác>
else:
    <Các lệnh xử lý khi không có lỗi>
finally:
<Các lệnh luôn thực hiện>
```

Trình tự xử lý

- Thực hiện nhóm lệnh cần bắt lỗi
- Nếu không lỗi thì chuyển sang lệnh tiếp theo say try..except
- Nếu có lỗi và lỗi này khớp với các mã lỗi có trong except thì sẽ thực hiện các lệnh xử lý tương ứng với except đó
- Nếu có lỗi và lỗi này không khớp với các mã lỗi except nằm trên thì rơi vào Xử lý các lỗi khác nằm cuối cùng
- Nếu không có lỗi sẽ rơi vào khối lệnh else
- Nhóm lệnh trong finally luôn được thực hiện dù lỗi hay không lỗi

12.2 Bắt lỗi và kiểm soát lỗi

Ví dụ

```
def divide(x, y)
try:
   result = x / y;
except ZeroDivisionError:
     print(“Không thể chia cho 0”)
else:
     print(“Kết quả là: ”, result)
finally:
     print(“Đã thực hiện xong lệnh”)
```

Tổng kết lại bài

1. Nắm được tổng quan lỗi xảy ra trong Python
2. Nắm được cách bắt lỗi và xử lý lỗi