BÀI 7

Exception Handling & File IO

Tóm Tắt Nội Dung

Trong bài học này chúng ta sẽ đi tìm hiểu lần lượt các nội dung

1. Tổng quan về xử lí ngoại lệ trong Python
2. Tổng quan về xử lí file tập tin
3. Phương thức đọc file
4. Phương thức ghi và tạo mới file
5. Phương thức xóa file, folder

7.1 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 !

7.2 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

7.2 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

7.2 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.

7.2 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

7.2 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")
```

7.2 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

7.2 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")
```

7.2 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

7.2 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”)
```

7.2 Bắt lỗi và kiểm soát lỗi

Nguồn tham chiếu

- Xử lý ngoại lệ trong Python

  https://www.w3schools.com/python/python_try_except.asp

7.3 Tổng quan về xử lí file tập tin

Khái niệm file handling (xử lý tệp tin)

- Trong Python là quá trình làm việc với các tệp tin trên hệ thống máy tính. Điều này bao gồm đọc tệp, ghi vào tệp và thực hiện các thao tác khác liên quan đến tệp. Python cung cấp một số phương thức và module hỗ trợ làm việc với tệp rất dễ dàng.

- Python có thể làm việc với nhiều loại tệp khác nhau, bao gồm:

1. **Tệp văn bản** (Text files): Đây là loại tệp thông thường chứa dữ liệu dưới dạng văn bản

2. **Tệp nhị phân** (Binary files): Đây là các tệp không chứa dữ liệu văn bản, mà thường là dữ liệu nhị phân, hình ảnh, âm thanh, hoặc các loại dữ liệu khác không thuộc kiểu văn bản

3. **Tệp CSV** (CSV files): CSV (Comma Separated Values) là một loại tệp văn bản đặc biệt, được sử dụng để lưu trữ dữ liệu dưới dạng bảng, trong đó các giá trị cách nhau bởi dấu phẩy

7.3 Tổng quan về xử lí file tập tin

Khái niệm file handling (xử lý tệp tin)

4. **Tệp JSON** (JSON files): JSON (JavaScript Object Notation) là một định dạng dữ liệu phổ biến dựa trên văn bản, thường được sử dụng để truyền và lưu trữ dữ liệu dạng cấu trúc

5. **Tệp XML** (XML files): XML (eXtensible Markup Language) là một định dạng dữ liệu được sử dụng để lưu trữ thông tin dưới dạng cây cấu trúc

6. **Tệp SQLite** (SQLite files): SQLite là một hệ quản lý cơ sở dữ liệu nhẹ và phổ biến được tích hợp sẵn trong Python

7. **Các loại tệp khác**: Ngoài các loại tệp nêu trên, Python cũng có thể làm việc với các loại tệp tùy chỉnh hoặc định dạng tệp đặc biệt khác thông qua việc sử dụng các thư viện phù hợp hoặc tự viết mã xử lý tùy chỉnh

7.3 Tổng quan về xử lí file tập tin

Khái niệm file handling (xử lý tệp tin)

- Trong phạm vi bài học này, chúng ta chỉ tìm hiểu kiểu tập tin văn bản thuần.
- Tệp văn bản thường có phần mở rộng **.txt** hay **.dat**

7.3 Tổng quan về xử lí file tập tin

Phương thức open() - Mở tệp tin

- Việc đọc hay ghi dữ liệu từ tệp tin sẽ thông qua một đầu đọc (file head). Khi đọc hoặc ghi dữ liệu sẽ đọc hay ghi dữ liệu trực tiếp từ vị trí của đầu đọc
- Khi đọc xong hay ghi xong đầu đọc sẽ dịch chuyển về cuối tệp tin đúng một khoảng bằng lượng thông tin (tính theo byte) đã đọc (ghi) được.

- Lệnh **open("tên_file.txt", "r")** sẽ mở tệp sẵn sàng đọc. Đầu đọc sẽ chuyển về vị trí gốc là 0.
- Lệnh **open("tên_file.txt", "w")** sẽ mở tệp, xóa tất cả nội dung đang có và đưa đầu đọc về vị trí đầu tệp. Nếu tệp chưa tồn tại thì tạo mới

7.3 Phương thức đọc tập tin

Phương thức open() - Mở tệp tin

Để làm việc với tệp, trước tiên chúng ta cần mở tệp. Sử dụng hàm **open()** để mở một tệp trong Python. Hàm này trả về một đối tượng tệp tin, cho phép bạn thực hiện các thao tác đọc/ghi trên tệp.

```
file = open("ten_file.txt", ["mode“])
```

- **ten_file.txt:** là tên tệp tin muốn mở
- **mode:** tùy chọn, là chế độ mở tệp với các tùy chọn sau:
  - **r:** là Read (đọc, mở), giá trị mặc định
  - **w:** là Write (ghi đè), nếu tệp chưa tồn tại thì tạo mới
  - **a:** là Append (ghi nối vào cuối tệp), nếu tệp chưa tồn tại thì tạo mới
  - **x:** là Create (tạo mới tệp), trả về lỗi nếu tệp đã tồn tại
  - **t:** là text mode, giá trị mặc định

7.3 Phương thức đọc tập tin

Phương thức open() - Mở tệp tin

Ví dụ: để mở file demo_file.txt bạn chỉ cần code như sau:

```
file = open(“demo_file.txt")
```

Dòng code trên sẽ tương đương với, vì **rt** là mặc định

```
file = open(“demo_file.txt“, “rt”)
```

7.2 Phương thức đọc tập tin

Phương thức read() – Đọc toàn bộ nội dung tệp tin

Ví dụ bạn có một file demo_file.txt với nội dung:

```
Hello! Welcome to demo_file.txt
This file is for testing purposes.
Good Luck!
```

Để lấy được nội dùng này và hiển thị ra cho người xem bạn code nhu sau

```
file = open("demo_file.txt", “r")
content = f. f.read() #Lấy tất cả nội dung
print(content)
print(f.read(5)) #Chỉ lấy 5 kí tự đầu tiên
```

Đầu tiên mở file ra, sau đó dùng phương thức **read()** để đọc nội dung tập tin

7.3 Phương thức đọc tập tin

Phương thức read() – Đọc toàn bộ nội dung tệp tin

Chúng ta cùng tìm hiểu cách đọc dữ liệu trong Python qua 2 lần gọi read()

Trong lần gọi read() đầu tiên, toàn bộ nội dung được lưu vào biến nhớ content (content là một str) , khi đó đầu đọc nằm ở vị trí cuối tệp

Nếu gọi read() lần thứ 2: content = f.read() thì content bây giờ là rỗng.
Lí do là lệnh read() luôn đọc dữ liệu từ vị trí đầu đọc trở đi

7.3 Phương thức đọc tập tin

Làm sao biết đầu đọc đang ở vị trí nào ?

Hàm tell() sẽ cho chúng ta biết vị trí của đầu đọc tính từ vị trí đầu tệp tin.

```
f = open("demo_file.txt", “r")
print(f.tell())  #Mở tệp đầu đọc ở vị trí số 0 (đầu tệp)

content = f.read()
print(f.tell())  #Đọc xong đầu đọc ở vị trí số 110 (cuối tệp)

f.close() #đóng tệp, kết thúc phiên làm việc với file hiện tại
```

Phương thức **close()** được gọi sau khi bạn mở file và ghi xong rất quan trọng, nó giúp: Giải phóng tài nguyên, Lưu trữ dữ liệu, Tránh lỗi khi làm việc sau khi tệp đã được đóng.

7.3 Phương thức đọc tập tin

Phương thức readline()

Phương thức này được sử dụng để đọc một dòng văn bản từ tệp. Mỗi lần gọi phương thức readline(), nó sẽ đọc một dòng ký tự đầu tiên của tệp văn bản và di chuyển con trỏ tệp đến dòng kế tiếp. Khi không còn dòng nào để đọc, readline() trả về một chuỗi rỗng

```
f = open("demo_file.txt", “r")
line1 = f.readline()
print(line1)  # In ra dòng đầu tiên của tệp

line2 = f.readline()
print(line2)  # In ra dòng thứ 2 của tệp
```

7.3 Phương thức đọc tập tin

Phương thức readlines()

Phương thức này được sử dụng để đọc tất cả các dòng văn bản từ tệp và trả về một danh sách (kiểu list) các chuỗi, trong đó mỗi phần tử trong danh sách là một dòng văn bản từ tệp

```
f = open("demo_file.txt", “r")
lines = f.readlines()
print(type(lines)) #in ra được: list
print(lines)  # In ra toàn bộ nội dung của tệp ở dạng list
```

Khi bạn sử dụng readlines(), toàn bộ nội dung của tệp sẽ được đọc và lưu trữ trong bộ nhớ. Nếu tệp quá lớn, điều này có thể ảnh hưởng đến hiệu năng và tiêu tốn nhiều tài nguyên.

7.3 Phương thức ghi và tạo mới file

Phương thức write() - Ghi tệp tin đã tồn tại

Để ghi nội dung vào một tệp tin đã tồn tại trước hết bạn phải mở nó với phương thức open() và chọn 1 trong 2 chế độ (mode):

a: là Append (ghi nối vào cuối tệp), nếu tệp chưa tồn tại thì tạo mới

w: là Write (ghi đè), nếu tệp chưa tồn tại thì tạo mới

Ví dụ mở file demo_file.txt và ghi tiếp nội dung vào

```
f = open("demo_file.txt", "a")
f.write("Now the file has more content!")
f.close()
```

7.3 Phương thức ghi và tạo mới file

Phương thức write() - Ghi tệp tin đã tồn tại

Ví dụ mở file demo_file.txt và **ghi đè** nội dung đã có

```
file = open("demo_file.txt", “w")
f.write("Now the file has more content!")
f.close()
```

Bạn kiểm tra nội dung đã được thay đổi chưa với cách code như sau:

```
file = open("demo_file.txt", “r")
print(f.read())
```

Đầu tiên mở file ra, sau đó dùng phương thức read() để đọc nội dung tập tin

7.3 Phương thức ghi và tạo mới file

Phương thức write() - Ghi tệp tin đã tồn tại

Ghi nội xuống dòng

```
f = open("demo_file.txt", “w")
msg1 = ‘Đây là dòng 1 \n’
msg2 = ‘Đây là dòng 2 \n’
msg3 = ‘Đây là dòng 3 \n’
f.write(msg1)
f.write(msg2)
print(msg3, file = fileout) #Ghi nội dung với lệnh print()
f.close()
```

Để ghi nhiều dòng và muốn xuống dòng bạn có thể sử dụng kí tự \n vào cuối mỗi dòng như ví dụ trên

Bạn cũng có thể sử dụng lệnh print() với tham số thứ 2 (file) như trên để ghi nội dung

7.3 Phương thức ghi và tạo mới file

Phương thức writelines() - Ghi tệp tin đã tồn tại

Đây là một tùy chọn khác cho việc ghi file, khi bạn có nội dung cần ghi ở dạng list các chuỗi văn bản

```
f = open("demo_file.txt", “w")
mylist = ["apple", "banana", "cherry"]
f.writelines(mylist)
f.close()
```

Với cách này bạn không thể sử dụng kí tự \n để xuống dòng

7.3 Phương thức ghi và tạo mới file

Tạo mới tệp tin

Để tạo mởi một tệp tin trong Python, bạn sử dụng phương thức open() với một trong 3 tùy chọn:

w: là Write (ghi đè), nếu tệp chưa tồn tại thì tạo mới

a: là Append (ghi nối vào cuối tệp), nếu tệp chưa tồn tại thì tạo mới

x: là Create (tạo mới tệp tin), trả về lỗi nếu tệp đã tồn tại

Tạo mới file myfile.txt (tạo file với nội dung trống)

```
file = open("myfile.txt", "x")
```

Tạo mới file myfile.txt nếu nó chưa tồn tại, có thể thêm nội dung sau khi tạo file

```
file = open("myfile.txt", "w")
```

7.3 Phương thức xóa tệp tin, folder

Xóa tệp tin

Để xóa được một tệp tin (file) bạn cần đến OS module, sau đó dùng phương thức os.remove() như sau:

```
import os
os.remove("demofile.txt")
```

Để tránh lỗi xay ra khi xóa file bạn nên kiểm tra sự tồn tại của file trước

```
import os
if os.path.exists("demofile.txt"): #kiểm tra xem tồn tại không
   os.remove("demofile.txt") #xóa nếu tồn tại
else:
   print("File không tồn tại")
```

7.3 Phương thức xóa tệp tin, folder

Tạo folder

Tạo một folder rỗng trong Python với đường dẫn chỉ định

```
import os
os.mkdir(“path/to/new_directory")
```

Xóa folder

Để xóa được một folder bạn cần đến OS module, sau đó dùng phương thức os.rmdir() như sau:

```
import os
os.rmdir(“myfolder")
```

Và lưu ý là nó chỉ xóa được khi trong folder đó rỗng

7.3 Phương thức xóa tệp tin, folder

Đổi tên file

Đổi tên một file trong Python với đường dẫn chỉ định

```
import os
os.rename("path/to/old_filename.txt", "path/to/new_filename.txt")
```

Đổi tên folder

Dùng chung hàm với đổi file

```
import os
os.rename("path/to/old_folder_name", "path/to/new_folder_name")
```

7.3 Phương thức xóa tệp tin, folder

Nguồn tham chiếu

- Xử lý file trong Python – w3School

  https://www.w3schools.com/python/python_file_handling.asp

- Các phương thức xử lý trong module os của Python

  https://www.w3schools.com/python/module_os.asp

Tổng kết lại bài

1. Nắm được tổng quan về xử lí file tập tin
2. Nắm được phương thức đọc file
3. Nắm được phương thức ghi và tạo mới file
4. Nắm được phương thức xóa file, folder
5. Nắm được cách thức xử lý lỗi trong Python