Lập trình Python

Bài 9: Làm việc với tập tin trong Python

Tóm tắt nội dung bài trước

- Python đòi hỏi lập trình viên triệt để xử lý các vấn đề phát sinh khi thực thi chương trình bằng cơ chế ngoại lệ
- Python cung cấp cú pháp try-except-else-finally để xử lý ngoại lệ
  - Khối try chứa đoạn mã có thể phát sinh lỗi
  - Khối except để xử lý ngoại lệ phát sinh từ khối try
  - Khối else thực thi trong trường hợp khối try không sinh lỗi
  - Khối finally luôn được thực thi trong mọi tình huống, sử dụng để thực thi những đoạn mã "dọn dẹp" các vấn đề còn tồn đọng
- Lập trình viên có thể tự sinh ngoại lệ bằng cách sử dụng lệnh raise và có thể tự tạo hệ thống ngoại lệ của riêng mình nếu cần

Nội dung

1. Quan điểm xử lý tập tin của Python
2. Đóng/Mở tập tin
3. Đọc/Ghi dữ liệu của tập tin
4. Con trỏ tập tin
5. Làm việc với hệ thống thư mục
6. Bài tập

Phần 1

Quan điểm xử lý tập tin của Python

Quan điểm xử lý tập tin của Python

- Tập tin (file) và thư mục (folder) là thành phần cơ bản của hệ thống lưu trữ dữ liệu bền vững
  - Tuy có một vài hệ thống không sử dụng những khái niệm này
- Python chia các tác vụ tập tin làm hai nhóm:
  1. Tác vụ quản lý: không ảnh hưởng đến nội dung tập tin (đổi tên, di chuyển, xóa, sao chép, phân quyền,...)
     - Python cung cấp nhiều hàm thuộc thư viện os (import os) để xử lý chỉ với 1-2 dòng lệnh
  2. Tác vụ nội dung: có tương tác với nội dung tập tin (đọc, ghi,...)

Quan điểm xử lý tập tin của Python

- Loại tác vụ nội dung, Python thực hiện quy trình 3 bước mở tập tin – xử lý – đóng tập tin tương tự như các ngôn ngữ lập trình khác
- Python cũng chia tập tin làm hai loại:
  - Tập tin văn bản: chứa nội dung chủ yếu là text và các dấu trình bày (tab, xuống dòng, căn lề,...)
    - Python tự động xử lý việc lưu trữ dấu xuống dòng khác nhau giữa các hệ điều hành Linux/Unix và Windows
    - Python hỗ trợ việc tự động chuyển đổi mã hóa (encode) giữa các loại văn bản khác nhau
  - Tập tin nhị phân: Python xem như dãy các byte dữ liệu và thường thao tác theo các khối dữ liệu để tăng tốc độ xử lý
    - Ví du: file âm thanh, hình ảnh, .exe

Phần 2

Đóng/Mở tập tin

Làm việc với nội dung tập tin

- Gồm 3 bước:
  1. Mở tập tin: Đây là bước yêu cầu hệ thống chuẩn bị các điều kiện cần thiết để đọc/ghi nội dung tập tin bao gồm định vị dữ liệu trên vùng lưu trữ và khởi tạo các vùng đệm
  2. Làm việc với tập tin: Bước chính của quá trình, trong bước này chương trình thực hiện các thao tác liên quan đến nội dung tập tin
  3. Đóng tập tin: Đảm bảo nội dung mới được cập nhật lên vùng lưu trữ và giải phóng các tài nguyên đã được cấp phát để làm việc với tập tin
- Các bước này đều có thể phát sinh ngoại lệ IOError

Mở file

- Thao tác mở file khá chậm, vì thực hiện những bước sau:
  - Kiểm tra người dùng có mở quá nhiều file không
  - Kiểm tra file có tồn tại trên hệ thống hay không
  - Kiểm tra chương trình có quyền truy cập nội dung hay không
  - Kiểm tra có thể thao tác file vào thời điểm hiện tại hay không
    - File có thể bị khóa bởi chương trình khác
    - File có thể chỉ đọc vì được ghi trên thiết bị cấm ghi
    - File có thể chỉ ghi vì nó là loại thiết bị cấm đọc
  - Định vị vùng dữ liệu file trên thiết bị lưu trữ
  - Chuẩn bị vùng đệm cho việc đọc/ghi dữ liệu
- Vì vậy chỉ mở file khi cần thiết
- Chọn cách mở file phù hợp với mục đích xử lý

Mở file: `f = open(filename, mode)`

Giá trị	Ý nghĩa
t	Mở file văn bản (chế độ mặc định).
r	Mở tập tin văn bản chỉ để đọc (chế độ mặc định). Nếu file không tồn tại sẽ xảy ra lỗi
w	Mở tập tin văn bản để ghi, nếu tập tin không tồn tại thì sẽ tạo mới
x	Tạo tập tin văn bản mới để ghi, sinh lỗi nếu tập tin đã tồn tại
a	Mở tập tin văn bản để ghi tiếp vào cuối nếu tập tin đã tồn tại, nếu tập tin không tồn tại thì sẽ tạo mới
b	Mở file nhị phân
+	Mở file cho cả đọc và ghi (cập nhật thêm chế độ, ví dụ: r chỉ đọc thì r+ cả đọc và ghi; w chỉ ghi thì w+ là cả ghi và đọc)

Ví dụ về mở file

```
# mở file mode 'r' hoặc 'rt' để đọc (chế độ mặc định)
f1 = open("test.txt")
# mở file mode 'w' để ghi
f2 = open("test.txt", 'w')
# mở file mode 'r+b' để đọc và ghi dạng nhị phân
f3 = open("img.bmp", 'r+b')
# mở file văn bản để đọc
# chỉ định rõ nội dung được mã hóa dạng utf-8
f4 = open("test.txt", mode = 'r', encoding = 'utf-8')
```

Đóng file: `f.close()`

- Thao tác đóng file rất quan trọng:
  - Đẩy mọi dữ liệu trên vùng đệm xuống thiết bị lưu trữ
  - Cập nhật thông tin trên hệ thống file (filesize, last update,…)
  - Giải phóng vùng dữ liệu dùng cho làm việc với file
- Vì vậy khi không sử dụng đến file nữa, nên đóng file ngay
- Quên đóng file có gây lỗi không? Không có lỗi, hệ thống tự đóng tất cả các file đang mở khi kết thúc chương trình
- Sử dụng phát biểu with giúp tự động đóng file:

```
with open("test.txt", encoding = 'utf-8') as f:
    # thực hiện các thao tác với tệp
    ...
# biến f bị hủy, tập tin được tự động đóng lại
```

Đóng file: f.close()

- Cách làm đúng nhất là đóng file trong khối finally:

```
try:
    # mở tập tin
    f = open("test.txt", encoding = 'utf-8')
    # thực hiện các thao tác với tệp
    ...
finally:
    # đóng tập tin
    f.close()
```

- Chú ý: ngay cả lệnh đóng file cũng có thể sinh ngoại lệ, trong trường hợp này ngoại lệ sẽ được trả về khối try bên ngoài

Phần 3

Đọc/Ghi dữ liệu của tập tin

Các hàm đọc file

- read(N): đọc N byte/ký tự tiếp theo
  - Nếu không viết N thì đọc đến cuối file
  - Nếu dữ liệu trong file không đủ N byte/ký tự thì đọc đến cuối file
  - Nếu tập tin mở ở chế độ văn bản thì trả về str
  - Nếu tập tin mở ở chế độ nhị phân thì trả về dãy byte
- readline(N): đọc một dòng từ file, tối đa là N byte/ký tự, nếu không viết N thì trả về str là dữ liệu được đọc tới khi nào gặp kí tự hết dòng hoặc hết file
  - Dữ liệu trả về bao gồm cả kí tự xuống dòng \n, trừ tình huống đọc dòng cuối cùng của file

Các hàm đọc file

- readlines(N): sử dụng readline đọc các dòng cho đến hết file và trả về một danh sách các string, nếu viết N thì sẽ xử lý tối đa là N byte hoặc số dòng tương ứng với N ký tự (tổng số ký tự của các dòng đọc được >= N)
- Các lập trình viên Python có cách đọc mọi dòng của file đơn giản hơn rất nhiều:

```
for line in open("test.txt", encoding = 'utf-8'):
    # thực hiện các thao tác với từng dòng
    ...
```

- Như vậy file văn bản trong Python là một kiểu tuần tự
- Đoạn mã trên có thể ghép cùng với phát biểu with:

```
with open('workfile') as f:
    for line in f:
        print(line, end='')
```

Các hàm ghi dữ liệu ra file

- write(data): ghi data ra file, trả về số byte/ký tự ghi được
  - Phương thức làm việc với cả file văn bản và file nhị phân
  - Nếu file văn bản thì data phải là kiểu str
  - Nếu file nhị phân thì data phải là khối byte (kiểu bytearray hoặc kiểu bytes)
- writelines(data): ghi toàn bộ nội dung data vào file theo từng dòng
  - Chỉ làm việc với kiểu file văn bản
  - Dữ liệu data phải là danh sách các str
  - Nếu cố dùng kiểu dữ liệu khác sẽ phát sinh lỗi TypeError

Phần 4

Con trỏ tập tin

Con trỏ tập tin

- Con trỏ tập tin là vị trí hiện thời sẽ đọc/ghi dữ liệu, tương tự như khi ta ghi dữ liệu lên màn hình
- Một tập tin chỉ có một con trỏ tập tin
- Khi mở tập tin ở chế độ “thêm cuối” (a – append), con trỏ tập tin tự động đặt ở cuối tập tin
- Các chế độ mở tập tin khác luôn đặt con trỏ tập tin ở đầu
- Python cung cấp một số lệnh cho phép lấy vị trí và di chuyển con trỏ tập tin
  - Chỉ nên sử dụng với tập tin nhị phân
  - Không phải loại tập tin nào cũng lấy được vị trí hoặc dịch chuyển được con trỏ

Phương thức làm việc với con trỏ tập tin

Phương thức	Chức năng
seekable()	Trả về **True** nếu tập tin là dạng truy cập ngẫu nhiên (dùng được phương thức **seek**)
tell()	Trả về vị trí con trỏ tập tin hiện tại (tính từ đầu tập tin)
seek(offset, from)	Chuyển con trỏ tập tin đến vị trí mới **offset**, tham số thứ hai **from** quyết định cách tính vị trí: - **SEEK_SET = 0**: tính từ đầu tập tin (mặc định) - **SEEK_CUR = 1**: tính từ vị trí hiện thời - **SEEK_END = 2**: tính từ cuối tập tin

Các phương thức khác của tập tin

Phương thức	Chức năng
fileno()	Trả về một số nguyên là mã định danh của tập tin
flush()	Đẩy dữ liệu khỏi vùng đệm, ghi xuống thiết bị lưu trữ
readable()	Trả về **True** nếu tập tin có thể đọc được
truncate(size)	Cắt tập tin hiện tại lấy đúng thành **size** byte đầu tiên
writable()	Trả về **True** nếu tập tin có thể ghi được.

Phần 5

Làm việc với hệ thống thư mục

Làm việc với hệ thống thư mục

- Các phương thức quản lý tập tin và thư mục đều thuộc thư viện os (import os)
- Các phương thức loại này đều có thể sinh lỗi OSError
- Để tương thích giữa các hệ điều hành, Python xem các dấu gạch chéo đều là dấu phân tách đường dẫn
  - Nghĩa là "c:\\test", r'c:\test' và "c:/test" được coi là như nhau
- Lấy thư mục làm việc hiện tại: `os.getcwd()`
- Thay đổi thư mục làm việc: `os.chdir(path)` với `path` là đường dẫn đến thư mục mới
- Tạo thư mục mới: os.mkdir(path)

Làm việc với hệ thống thư mục

- Lấy danh sách các thư mục và tập tin nằm trong thư mục path: `os.listdir(path)`
  - Nếu bỏ tham số `path` thì lấy danh sách từ thư mục hiện tại
- Đổi tên thư mục hoặc tập tin: `os.rename(old, new)`
  - Phương thức làm việc với cả tập tin và thư mục
- Xóa bỏ tập tin: `os.remove(filename)`
- Xóa bỏ thư mục: `os.rmdir(path)`
  - Phương thức chỉ xóa được thư mục trống
  - Trường hợp muốn xóa thư mục bất kể nó trống hay không, có thể sử dụng phương thức `shutil.rmtree(path)` (cần `import` thư viện `shutil`)

Phân giải đường dẫn theo thư mục làm việc

```
# Chú ý: Python phân giải đường dẫn theo thư mục hiện tại
# Vì vậy cần rất cẩn thận khi viết đường dẫn trong các lệnh

import os

# tạo thư mục abc trong thư mục hiện tại
os.mkdir('abc')
# tạo thư mục xyz trong thư mục abc thuộc thư mục hiện tại
os.mkdir('abc/xyz')
# tạo thư mục abc trong trong thư mục temp thuộc ổ đĩa C
os.mkdir('C:/temp/abc')
```

Phần 6

Bài tập

Bài tập

1. Đọc 1 file và in ra màn hình 5 dòng đầu tiên, nếu file không đủ 5 dòng thì in toàn bộ nội dung file.
2. Đọc 1 file và in ra màn hình 5 dòng cuối cùng, nếu file không đủ 5 dòng thì in toàn bộ nội dung file.
3. Đọc 1 file, tìm và in ra nội dung của dòng dài nhất trong file đó
4. Đọc 1 file, tìm và in ra từ dài nhất trong file
5. Đọc 1 file, thống kê và in ra tất cả các chữ cái có trong file và số lần xuất hiện của các chữ đó
6. Đọc 1 file, thống kê và tần xuất xuất hiện của tất cả các từ trong file, in theo thứ tự giảm dần của số lần xuất hiện

Bài tập

7. Đọc tập tin văn bản abc.txt sau đó chuyển nội dung đọc được sang chữ in hoa và ghi vào tập tin xyz.txt
8. Đọc tập tin văn bản abc.txt, lọc lấy tất cả các số trong dữ liệu đọc được và ghi chúng lần lượt vào tập tin number.txt, mỗi số ghi trên một dòng
9. Nhập tên của hai tập tin văn bản, đọc và so sánh xem nội dung của hai tập tin có giống nhau hay không
10. Lấy danh sách các tập tin trong thư mục hiện tại, tiến hành so sánh nội dung của các tập tin trong thư mục, và chỉ ra các tập tin có nội dung giống nhau. Chương trình in ra các nhóm tập tin có cùng nội dung, đánh số các nhóm tăng dần bắt đầu từ 1

Bài tập

11. Ảnh jpg là một tập tin nhị phân có đặc điểm là 4 byte dữ liệu từ vị trí thứ 6 đến thứ 9 tạo thành chữ JFIF (vị trí bắt đầu tính từ số 0). Hãy nhập tên tập tin, kiểm tra xem đó có thể là tập tin ảnh jpg không?

12. Ảnh bitmap (BMP) là một tập tin nhị phân:
    - 2 byte đầu tạo nên chữ BM
    - 4 byte từ vị trí 18-21 tạo nên số nguyên 4 byte là số dòng (chiều cao) của ảnh
    - 4 byte từ vị trí 22-25 tạo nên số nguyên 4 byte là số cột (chiều ngang) của ảnh

    Nhập tên tập tin, cho biết đó có phải là ảnh bitmap không, nếu đúng thì in ra cỡ của ảnh (dòng x cột)

note: Sử dung int.from_bytes(bytes, byteorder) để convert từ kiểu bytes sang int. byteorder nhận 'little' hoặc 'big', trong đó 'little' thể hiện thứ tự bit ý nghĩa nhất (LSB) →…→ bít nhiều ý nghĩa nhất (MSB).