# Tính bền bỉ

Cho đến nay, chúng ta đã học cách viết chương trình và truyền đạt ý định của mình tới Bộ xử lý trung tâm bằng cách sử dụng thực thi có điều kiện, các hàm và các phép lặp. Chúng ta đã học cách tạo và sử dụng cấu trúc dữ liệu trong Bộ nhớ chính. CPU và bộ nhớ là nơi phần mềm của chúng ta hoạt động và chạy. Đó là nơi tất cả "suy nghĩ" xảy ra.

Nhưng nếu bạn nhớ lại từ các cuộc thảo luận về kiến trúc phần cứng, khi tắt nguồn, mọi thứ được lưu trữ trong CPU hoặc bộ nhớ chính sẽ bị xóa. Vì vậy, cho đến nay, các chương trình của chúng ta chỉ là những bài tập thú vị nhất thời để học Python.

# Bộ nhớ thứ cấp

Trong chương này, chúng ta bắt đầu làm việc với Bộ nhớ thứ cấp (hoặc các tệp). Bộ nhớ thứ cấp không bị xóa khi tắt nguồn. Hoặc trong trường hợp ổ đĩa flash USB, dữ liệu chúng ta ghi từ các chương trình có thể bị xóa khỏi hệ thống và chuyển sang hệ thống khác.

Chúng ta sẽ chủ yếu tập trung vào việc đọc và ghi các tệp văn bản chẳng hạn như những tệp chúng ta tạo trong trình soạn thảo văn bản. Phần sau chúng ta sẽ xem cách làm việc với các tệp cơ sở dữ liệu là tệp nhị phân, được thiết kế đặc biệt để đọc và ghi thông qua phần mềm cơ sở dữ liệu.

# Mở các tệp tin

Khi chúng ta muốn đọc hoặc ghi một tệp (giả sử trên ổ cứng của bạn), trước tiên chúng ta phải mở tệp đó. Việc mở tệp sẽ giao tiếp với hệ điều hành của bạn, hệ điều hành này sẽ biết dữ liệu của mỗi tệp được lưu trữ ở đâu. Khi bạn mở một tệp, bạn đang yêu cầu hệ điều hành tìm tệp theo tên và đảm bảo rằng tệp tồn tại. Trong ví dụ này, chúng ta mở tệp mbox.txt, tệp này sẽ được lưu trữ trong cùng một thư mục mà bạn đang ở khi khởi động Python. Bạn có thể tải xuống tệp này từ trang  [www.py4e.com/code3/mbox.txt](https://www.py4e.com/code3/mbox.txt)

>>> fhand = open('mbox.txt')

>>> print(fhand)

<\_io.TextIOWrapper name='mbox.txt' mode='r' encoding='cp1252'>

Nếu được  open (mở) thành công, hệ điều hành trả về cho chúng ta một tệp tin xử lý (file handle). File handle không phải là dữ liệu thực có trong tệp, nhưng thay vào đó, nó là một "xử lý" mà chúng ta có thể sử dụng để đọc dữ liệu. Bạn được xử lý nếu tệp được yêu cầu tồn tại và bạn có quyền thích hợp để đọc tệp.

# Tệp tin xử lý (File Handle)

Nếu một tệp không tồn tại, việc open (mở) sẽ không thành công với một lần truy xuất và bạn sẽ không có quyền truy cập vào nội dung của tệp:

>>> fhand = open('stuff.txt')

Traceback (most recent call last):

File "<stdin>", line 1, in <module>

FileNotFoundError: [Errno 2] No such file or directory: 'stuff.txt'

Sau đó chúng ta sẽ sử dụng try và  except để giải quyết các tình huống khi chúng ta cố gắng mở các tệp tin không tồn tại dễ dàng hơn.

# Tệp văn bản và dòng

Một tệp văn bản có thể được coi là một chuỗi các dòng, giống như một xâu ký tự Python có thể được coi là một chuỗi các ký tự. Ví dụ: đây là một mẫu tệp văn bản ghi lại hoạt động thư từ các cá nhân khác nhau trong nhóm phát triển dự án nguồn mở:

From stephen.marquard@uct.ac.za Sat Jan 5 09:14:16 2008

Return-Path: <postmaster@collab.sakaiproject.org>

Date: Sat, 5 Jan 2008 09:12:18 -0500

To: source@collab.sakaiproject.org

From: stephen.marquard@uct.ac.za

Subject: [sakai] svn commit: r39772 - content/branches/

Details: http://source.sakaiproject.org/viewsvn/?view=rev&rev=39772

...

Toàn bộ tệp tương tác mail có sẵn ở

[www.py4e.com/code3/mbox.txt](https://www.py4e.com/code3/mbox.txt)

và một bản rút gọn của tệp có sẵn ở

[www.py4e.com/code3/mbox-short.txt](https://www.py4e.com/code3/mbox-short.txt)

Các tệp này ở định dạng tiêu chuẩn cho một tệp chứa nhiều thư. Các dòng bắt đầu bằng "From" phân tách các thư và các dòng bắt đầu bằng "From:" là một phần của thư. Để biết thêm thông tin về định dạng mbox, hãy xem <https://en.wikipedia.org/wiki/Mbox>.

Để ngắt tệp thành các dòng, có một ký tự đặc biệt biểu diễn cho "cuối dòng" được gọi là ký tự *newline.*

Trong Python, chúng ta biểu diễn ký tự *newline* dưới dạng \n trong hằng số xâu ký tự. Mặc dù trông giống như hai ký tự, nhưng nó thực sự là một ký tự duy nhất. Khi chúng ta xem biến bằng cách nhập "stuff" vào trình thông dịch, nó hiển thị cho chúng ta \n trong xâu ký tự, nhưng khi chúng ta sử dụng print để hiển thị xâu ký tự, chúng ta thấy xâu ký tự được chia thành hai dòng bởi ký tự newline .

>>> stuff = 'Hello\nWorld!'

>>> stuff

'Hello\nWorld!'

>>> print(stuff)

Hello

World!

>>> stuff = 'X\nY'

>>> print(stuff)

X

Y

>>> len(stuff)

3

Bạn cũng có thể thấy rằng độ dài của xâu X\nY  là ba ký tự vì ký tự newline là một ký tự.

Vì vậy, khi chúng ta nhìn vào các dòng trong một tệp, chúng ta cần *tưởng tượng* có một ký tự vô hình đặc biệt được gọi là newline ở cuối mỗi dòng đánh dấu kết thúc của một dòng.

Vì vậy, ký tự newline phân tách các ký tự trong tệp thành các dòng.

# Đọc tệp tin

Mặc dù *file handle* không chứa dữ liệu cho tệp, nhưng khá dễ dàng để tạo một vòng lặp for để đọc qua và đếm từng dòng trong tệp:

fhand = open('mbox-short.txt')

count = 0

for line in fhand:

count = count + 1

print('Line Count:', count)

# Code: http://www.py4e.com/code3/open.py

Chúng ta có thể sử dụng file handle làm chuỗi trong vòng lặp for của chúng ta. Vòng lặp for  chỉ đơn giản là đếm số dòng trong tệp và in chúng ra. Bản dịch sơ của vòng lặp for  sang tiếng Anh là, "dành cho mỗi dòng trong tệp được biểu diễn bởi file handle, hãy thêm 1 vào biến count."

Lý do mà hàm open không đọc toàn bộ tệp là tệp có thể khá lớn với nhiều gigabyte dữ liệu. Câu lệnh open vẫn mất một khoảng khoảng thời gian đó, cho cho dù dù kích thước của tệp là bao nhiêu.Vòng lặp for có thể đọc dữ liệu từ tệp.

Khi tệp được đọc bằng vòng lặp for theo cách này, Python sẽ xử lý việc tách dữ liệu trong tệp thành các dòng riêng biệt bằng cách sử dụng ký tự newline . Python đọc từng dòng qua newline và đặt newline làm ký tự cuối cùng trong biến line cho mỗi lần lặp của vòng lặp for.

Bởi vì vòng lặp for đọc dữ liệu từng dòng một, nó có thể đọc và đếm các dòng trong các tệp rất lớn một cách hiệu quả mà không dùng hết bộ nhớ chính để lưu trữ dữ liệu. Chương trình trên có thể đếm các dòng trong bất kỳ tệp kích thước nào sử dụng rất ít bộ nhớ vì mỗi dòng được đọc, đếm và sau đó bị loại bỏ.

Nếu bạn biết tệp tương đối nhỏ so với kích thước của bộ nhớ chính, bạn có thể đọc toàn bộ tệp thành một xâu bằng cách sử dụng phương thức  read  trên file handle.

>>> fhand = open('mbox-short.txt')

>>> inp = fhand.read()

>>> print(len(inp))

94626

>>> print(inp[:20])

From stephen.marquar

Trong ví dụ này, toàn bộ nội dung (tất cả 94.626 ký tự) của tệp mbox-short.txt được đọc trực tiếp vào biến inp. Chúng ta sử dụng phương pháp tách xâu (string slicing) để in ra 20 ký tự đầu tiên của dữ liệu xâu được lưu trữ trong inp.

Khi tệp được đọc theo cách này, tất cả các ký tự bao gồm tất cả các dòng và ký tự newline là một xâu lớn trong biến inp. Bạn nên lưu trữ output của read dưới dạng một biến vì mỗi lần gọi read cũng sẽ tốn tài nguyên:

>>> fhand = open('mbox-short.txt')

>>> print(len(fhand.read()))

94626

>>> print(len(fhand.read()))

0

Hãy nhớ rằng dạng hàm open này chỉ nên được sử dụng nếu dữ liệu tệp khớp với bộ nhớ chính của máy tính của bạn. Nếu tệp quá lớn,không vừa với bộ nhớ chính, bạn nên viết chương trình của mình để đọc tệp theo từng phần bằng cách sử dụng vòng lặp for hoặc while.

# Tìm kiếm trong tệp

Khi bạn đang tìm kiếm thông qua dữ liệu trong một tệp, việc đọc qua tệp rất phổ biến, bỏ qua hầu hết các dòng và chỉ xử lý các dòng đáp ứng một điều kiện cụ thể. Chúng ta có thể kết hợp mẫu để đọc tệp với các phương thức xâu để xây dựng cơ chế tìm kiếm đơn giản.

Ví dụ: nếu chúng ta muốn đọc một tệp và chỉ in ra những dòng bắt đầu bằng tiền tố “From:”, chúng ta có thể sử dụng phương thức xâu *startwith* để chỉ chọn những dòng có tiền tố mong muốn:

fhand = open('mbox-short.txt')

count = 0

for line in fhand:

if line.startswith('From:'):

print(line)

# Code: http://www.py4e.com/code3/search1.py

Khi chương trình này chạy, chúng ta có thể thể có các output sau:

From: stephen.marquard@uct.ac.za

From: louis@media.berkeley.edu

From: zqian@umich.edu

From: rjlowe@iupui.edu

...

Output trông tuyệt vời vì những dòng duy nhất chúng ta đang thấy là những dòng bắt đầu bằng “From:”, nhưng tại sao chúng ta lại thấy những dòng trống thừa? Điều này là do ký tự *newline* không thể nhìn thấy được đó. Mỗi dòng kết thúc bằng một newline, vì vậy câu lệnh print in xâu ký tự trong biến *line* bao gồm một newline và sau đó print thêm một newline khác, dẫn đến hiệu ứng giãn cách kép mà chúng ta thấy.

Chúng ta có thể sử dụng phương pháp tách (slicing) để in tất cả trừ ký tự cuối cùng, nhưng một cách tiếp cận đơn giản hơn là sử dụng phương thức *rstrip* để loại bỏ khoảng trắng từ phía bên phải của một xâu như sau:

fhand = open('mbox-short.txt')

for line in fhand:

line = line.rstrip()

if line.startswith('From:'):

print(line)

# Code: http://www.py4e.com/code3/search2.py

Khi chương trình này chạy, chúng ta có thể thể có các output sau:

From: stephen.marquard@uct.ac.za

From: louis@media.berkeley.edu

From: zqian@umich.edu

From: rjlowe@iupui.edu

From: zqian@umich.edu

From: rjlowe@iupui.edu

From: cwen@iupui.edu

...

Khi các chương trình xử lý tệp của bạn trở nên phức tạp hơn, bạn có thể muốn cấu trúc các vòng lặp tìm kiếm của mình bằng cách sử dụng continue. Ý tưởng cơ bản của vòng lặp tìm kiếm là bạn đang tìm kiếm các dòng “interesting” và bỏ qua các dòng “uninteresting” theo cách hiệu quả. Và sau đó khi chúng ta tìm thấy một dòng “interesting” , chúng ta sẽ làm làm gì đó với dòng đó.

Chúng ta có thể cấu trúc vòng lặp theo mẫu bỏ qua các dòng uninteresting như sau:

fhand = open('mbox-short.txt')

for line in fhand:

line = line.rstrip()

# Skip 'uninteresting lines'

if not line.startswith('From:'):

continue

# Process our 'interesting' line

print(line)

# Code: http://www.py4e.com/code3/search3.py

Output của chương trình là như nhau. Trong tiếng Anh, những dòng uninteresting là những dòng không bắt đầu bằng “From:”, chúng ta bỏ qua chúng khi sử dụng continue.. Đối với các dòng “interesting” (tức là những dòng bắt đầu bằng “From:”), chúng ta sẽ xử lý các dòng đó.

Chúng ta có thể sử dụng phương thức xâu find  để mô phỏng tìm kiếm trong trình soạn thảo văn bản để tìm các dòng mà xâu tìm kiếm nằm ở bất kỳ vị trí nào trong dòng. Vì find tìm kiếm sự xuất hiện của một xâu trong một xâu khác và trả về vị trí của xâu hoặc -1 nếu không tìm thấy xâu, chúng ta có thể viết vòng lặp sau để hiển thị các dòng chứa xâu ký tự “@ uct.ac.za ”(Tức là họ đến từ Đại học Cape Town ở Nam Phi):

fhand = open('mbox-short.txt')

for line in fhand:

line = line.rstrip()

if line.find('@uct.ac.za') == -1: continue

print(line)

# Code: http://www.py4e.com/code3/search4.py

và cho ra output sau:

From stephen.marquard@uct.ac.za Sat Jan 5 09:14:16 2008

X-Authentication-Warning: set sender to stephen.marquard@uct.ac.za using -f

From: stephen.marquard@uct.ac.za

Author: stephen.marquard@uct.ac.za

From david.horwitz@uct.ac.za Fri Jan 4 07:02:32 2008

X-Authentication-Warning: set sender to david.horwitz@uct.ac.za using -f

From: david.horwitz@uct.ac.za

Author: david.horwitz@uct.ac.za

Ở đây chúng ta cũng sử dụng dạng rút gọn của câu lệnh  if trong đó chúng ta đặt continue trên cùng dòng với if. Dạng rút gọn này của if hoạt động giống như if continue ở dòng tiếp theo và được thụt lề.

# Cho phép người dùng chọn tên tệp

Chúng ta thực sự không muốn phải chỉnh sửa code Python của mình mỗi khi xử lý một tệp khác. Sẽ hữu ích hơn nếu yêu cầu người dùng nhập xâu tên tệp mỗi khi chương trình chạy để họ có thể sử dụng chương trình trên các tệp khác nhau mà không cần thay đổi code Python.

Điều này thực hiện khá đơn giản, đọc tên tệp từ người dùng bằng cách sử dụng input  như sau:

fname = input('Enter the file name: ')

fhand = open(fname)

count = 0

for line in fhand:

if line.startswith('Subject:'):

count = count + 1

print('There were', count, 'subject lines in', fname)

# Code: http://www.py4e.com/code3/search6.py

Chúng ta đọc tên tệp từ người dùng và đặt nó vào một biến có tên fname và mở tệp đó. Bây giờ chúng ta có thể chạy chương trình nhiều lần trên các tệp khác nhau.

python search6.py

Enter the file name: mbox.txt

There were 1797 subject lines in mbox.txt

python search6.py

Enter the file name: mbox-short.txt

There were 27 subject lines in mbox-short.txt

Trước khi xem phần tiếp theo, hãy xem chương trình trên và tự hỏi bản thân, "Ở đây có gì sai không?" hoặc "Người dùng thân thiện của chúng ta có thể làm gì để khiến chương trình nhỏ xinh của chúng tôi thoát ra ngoài một cách vô duyên với tính năng truy xuất, khiến chúng trông không thú vị trong mắt người dùng?"

# Sử dụng try, except, và open

Tôi đã nói với bạn là không được xem trước. Đây là cơ hội cuối cùng của bạn.

Điều gì sẽ xảy ra nếu người dùng của chúng ta nhập thứ gì đó không phải là tên tệp?

python search6.py

Enter the file name: missing.txt

Traceback (most recent call last):

File "search6.py", line 2, in <module>

fhand = open(fname)

FileNotFoundError: [Errno 2] No such file or directory: 'missing.txt'

python search6.py

Enter the file name: na na boo boo

Traceback (most recent call last):

File "search6.py", line 2, in <module>

fhand = open(fname)

FileNotFoundError: [Errno 2] No such file or directory: 'na na boo boo'

Đừng cười. Người dùng cuối cùng sẽ làm mọi điều để có thể phá vỡ chương trình của bạn, dù có chủ đích hoặc với mục đích xấu. Trên thực tế, một phần quan trọng của bất kỳ nhóm phát triển phần mềm nào là một người hoặc một nhóm được gọi là Đảm bảo chất lượng (hay viết tắt là QA) có nhiệm vụ chính là làm những điều điên rồ nhất có thể để cố gắng phá vỡ phần mềm mà lập trình viên đã tạo.

Nhóm QA chịu trách nhiệm tìm ra các sai sót trong các chương trình trước khi chúng ta chuyển giao chương trình cho người dùng cuối, những người có thể đang mua phần mềm hoặc trả lương cho chúng ta để viết phần mềm. Vì vậy, nhóm QA là người bạn tốt nhất của lập trình viên.

Vì vậy, bây giờ chúng ta thấy lỗ hổng trong chương trình, chúng ta có thể sửa nó một cách dễ dàng bằng cách sử dụng cấu trúc try/except. Chúng ta cần giả định rằng việc gọi open có thể không thành công và thêm code khôi phục khi open không thành công như sau:

fname = input('Enter the file name: ')

try:

fhand = open(fname)

except:

print('File cannot be opened:', fname)

exit()

count = 0

for line in fhand:

if line.startswith('Subject:'):

count = count + 1

print('There were', count, 'subject lines in', fname)

# Code: http://www.py4e.com/code3/search7.py

Hàm exit kết thúc chương trình. Nó là một hàm mà chúng ta gọi mà không bao giờ trả về. Giờ đây, khi người dùng của chúng ta (hoặc nhóm QA) nhập tên tệp không hợp lệ hoặc tên tệp xấu, chúng ta sẽ "bắt" chúng và khôi phục theo cách tốt hơn:

python search7.py

Enter the file name: mbox.txt

There were 1797 subject lines in mbox.txt

python search7.py

Enter the file name: na na boo boo

File cannot be opened: na na boo boo

Việc bảo vệ lời gọi hàm open là một ví dụ điển hình về việc sử dụng try và except trong một chương trình Python. Chúng tôi sử dụng thuật ngữ "Pythonic" khi đang làm điều gì đó theo "cách của Python". Chúng ta có thể nói rằng ví dụ trên là cách Pythonic mở một tệp.

Một khi bạn trở nên thành thạo hơn với Python, bạn có thể tham gia vào cuộc đối đáp với các lập trình viên Python khác để quyết định xem giải pháp nào trong hai giải pháp tương đương cho một vấn đề là “ Pythonic hơn”. Mục tiêu để "Pythonic hơn" thể hiện quan điểm rằng lập trình vừa kỹ thuật, vừa nghệ thuật. Chúng ta không phải lúc nào cũng quan tâm đến việc làm cho một thứ gì đó hoạt động, chúng ta cũng muốn giải pháp của mình phải trang nhã và được đồng nghiệp đánh giá là trang nhã.

# Ghi tệp

Để ghi một tệp, bạn phải mở tệp đó với chế độ “w” làm tham số thứ hai

>>> fout = open('output.txt', 'w')

>>> print(fout)

<\_io.TextIOWrapper name='output.txt' mode='w' encoding='cp1252'>

Nếu tệp đã tồn tại, việc mở tệp ở chế độ ghi (write mode) sẽ xóa dữ liệu cũ và bắt đầu làm mới, vì vậy hãy cẩn thận! Nếu tệp không tồn tại, một tệp mới sẽ được tạo.

Phương thức  write của đối tượng file handle đưa dữ liệu vào tệp, trả về số ký tự được ghi. Chế độ ghi mặc định là văn bản để ghi (và đọc) các xâu ký tự.

>>> line1 = "This here's the wattle,\n"

>>> fout.write(line1)

24

Một lần nữa, đối tượng tệp sẽ theo dõi vị trí của nó, vì vậy nếu bạn gọi write một lần nữa, nó sẽ thêm dữ liệu mới vào cuối.

Chúng ta phải đảm bảo quản lý các đầu dòng khi chúng ta ghi vào tệp bằng cách chèn ký tự newline một cách rõ ràng khi chúng ta muốn kết thúc một dòng. Câu lệnh print tự động thêm newline, nhưng phương thức write không tự động thêm newline.

>>> line2 = 'the emblem of our land.\n'

>>> fout.write(line2)

24

Khi bạn ghi xong, bạn phải đóng tệp để đảm bảo rằng bit dữ liệu cuối cùng được ghi vật lý vào đĩa để nó không bị mất nếu mất điện.

>>> fout.close()

Chúng ta cũng có thể đóng các tệp mà chúng ta mở để đọc, nhưng sẽ hơi cẩu thả nếu chúng ta chỉ mở một vài tệp vì Python đảm bảo rằng tất cả các tệp đang mở đều được đóng khi chương trình kết thúc. Khi chúng ta đang ghi tệp, chúng ta muốn đóng tệp một cách rõ ràng để không để bất cứ vấn đề gì xảy ra.

# Gỡ lỗi

Khi bạn đang đọc và ghi tệp, bạn có thể gặp sự cố với khoảng trắng. Những lỗi này có thể khó gỡ lỗi vì khoảng trắng, tab và newline thường ẩn:

>>> s = '1 2\t 3\n 4'

>>> print(s)

1 2 3

4

Hàm repr tích hợp có thể giúp ích cho bạn. Nó nhận bất kỳ đối tượng nào làm đối số và trả về một biểu diễn xâu của đối tượng. Đối với xâu ký tự, nó biểu diễn các ký tự khoảng trắng với các chuỗi dấu gạch chéo ngược:

>>> print(repr(s))

'1 2\t 3\n 4'

Điều này có thể hữu ích cho việc gỡ lỗi.

Một vấn đề khác mà bạn có thể gặp phải là các hệ thống khác nhau sử dụng các ký tự khác nhau để biểu thị phần cuối của một dòng. Một số hệ thống sử dụng một newline, được biểu thị bằng \n. Những loại khác sử dụng ký tự trả về, được biểu diễn bằng \r. Một số sử dụng cả hai. Nếu bạn di chuyển tệp giữa các hệ thống khác nhau, những mâu thuẫn này có thể gây ra sự cố.

Đối với hầu hết các hệ thống, có các ứng dụng để chuyển đổi từ định dạng này sang định dạng khác. Bạn có thể tìm thấy chúng (và đọc thêm về vấn đề này) tại [https://www.wikipedia.org/wiki/Newline](https://wikipedia.org/wiki/Newline). Hoặc, tất nhiên, bạn có thể tự viết một cái.