File I/O trong Python

## Hệ thống nhập xuất chuẩn

Trong Python có 3 luồng nhập xuất là luồng input, luồng output và luồng error. Luồng input là luồng nhập, tất cả các dữ liệu đi vào chương trình đều đi qua đường này, luồng này mặc định được kết nối giữa chương trình với bàn phím. Luồng output là luồng xuất, làm nhiệm vụ đưa dữ liệu ra ngoài, mặc định gắn với màn hình console (trên Windows là cmd – Command Prompt, trên Linux là Terminal). Luồng error là luồng báo lỗi, luồng này đưa các thông báo lỗi ra ngoài, mặc định cũng gắn với màn hình console.

## Luồng input

Đây là dòng dữ liệu đi vào chương trình.

|  |  |
| --- | --- |
| name.py | |
| 1  2  3  4  5  6  7  8  9  10  11  12  13  14 | import sys      print ('Enter your name: ',)    name = ''    while True:     c = sys.stdin.read(1)     if c == '\n':        break     name = name + c    print ('Your name is:', name) |

Để làm việc với dòng nhập xuất thì chúng ta phải import module sys. Trong module này có đối tượng stdin với phương thức read(), phương thức này đọc các kí tự trong luồng nhập. Trong ví dụ trên, chúng ta in ra dòng chữ “Enter your name:”, sau đó cho một vòng lặp chạy, mỗi lần lặp chúng ta gọi phương thức read() trên để đọc từng kí tự vào từ luồng nhập và kiểm tra nếu kí tự vào là kí tự Enter thì thoát vòng lặp. Kí tự enter có kí hiệu là \n.

|  |  |
| --- | --- |
| Output | |
| 1  2 | Enter your name: Jan   Your name is: Jan |

Ngoài phương thức read() ra còn có các hàm làm nhiệm vụ tương tự là input() và raw\_input() .

Hàm input() in ra dòng thông báo, đọc dữ liệu vào và đánh giá  dữ liệu.

|  |  |
| --- | --- |
| input.py | |
| 1  2  3 | data = input('Enter expression: ')    print ('You have entered:', data) |
| Output | |
| 1  2 | Enter expression: 3\*3  You have entered: 9 |

Hàm raw\_input() cũng tương tự như hàm input().

## Luồng output

Luồng output là luồng dữ liệu đi ra khỏi chương trình.

|  |  |
| --- | --- |
| stdout.py | |
| 1  2  3  4 | import sys    sys.stdout.write('Honore de Balzac, Father Goriot\n')  sys.stdout.write('Honore de Balzac, Lost Illusions\n') |

Trong ví dụ trên, chúng ta in các đoạn text ra màn hình bằng phương thức write().

|  |  |
| --- | --- |
| Output | |
| 1  2 | Honore de Balzac, Father Goriot  Honore de Balzac, Lost Illusions |

Hàm print() mà chúng ta dùng lâu nay thực chất là nó gọi tới phương thức write() của đối tượng stdout.

|  |  |
| --- | --- |
| printkeyw.py | |
| 1  2 | print ('Honore de Balzac, The Splendors and Miseries of Courtesans')  print ('Honore de Balzac, Gobseck') |

## Chuyển hướng luồng nhập xuất vào file

Như đã nói ở đầu bài, luồng nhập xuất mặc định được gắn với màn hình và bàn phím. Tuy nhiên chúng ta có thể chuyển hướng chúng để làm việc với file.

|  |  |
| --- | --- |
| redirect.py | |
| 1  2  3  4  5  6  7  8  9  10  11  12  13  14 | import sys      f = open('output.txt', 'w')  sys.stdout = f    print ('Lucien')  sys.stdout.write('Rastignac\n')  sys.stdout.writelines(['Camusot\n', 'Collin\n'])    sys.stdout = sys.\_\_stdout\_\_    print ('Bianchon')  sys.stdout.write('Lambert\n') |

Trong đoạn code trên, chúng ta chuyển hướng stdout từ màn hình sang file.

|  |  |
| --- | --- |
| Output | |
| 1  2 | Bianchon  Lambert |
| output.txt | |
| 1  2  3  4 | Lucien  Rastignac  Camusot  Collin |

## **In kết quả ra màn hình trong Python**

Đến đây, chắc bạn đã quá quen thuộc về cách sử dụng của lệnh print. Lệnh này được sử dụng để in kết quả trên màn hình. Hàm này chuyển đổi biểu thức mà bạn đã truyền cho nó thành dạng chuỗi và ghi kết quả trên đầu ra chuẩn Standard Output. Cú pháp của lệnh print là:

print "Hoc Python la kha don gian,", "ban co thay vay khong?"

Kết quả là:

Hoc Python la kha don gian, ban co thay vay khong?

## **Đọc input từ bàn phím trong Python**

Python cung cấp hai hàm đã được xây dựng sẵn để nhận input từ người dùng. Hai hàm đó là:

* Hàm input()
* Hàm raw\_input()

## **Hàm input() trong Python**

Hàm này được sử dụng để nhận input từ người dùng. Hàm này giống hàm raw\_input(), nhưng với hàm input() này thì bất cứ biểu thức nào được nhập từ người dùng thì nó ước lượng và sau đó trả về kết quả. Ví dụ:

str = input("Nhap dau vao cua ban: ");

print "Dau vao da nhan la : ", str

Code trên sẽ cho kết quả sau tùy thuộc vào input bạn đã nhập:

Nhap dau vao cua ban: [x\*5 for x in range(2,10,2)]

Dau vao da nhan la : [10, 20, 30, 40]

## **Hàm raw\_input() trong Python**

Hàm raw\_input() được sử dụng để nhận đầu vào từ người dùng. Nó nhận đầu vào từ Standard Input dưới dạng một chuỗi và đọc dữ liệu từ từng dòng một. Ví dụ:

str = raw\_input("Nhap dau vao cua ban: ");

print "Dau vao da nhan la : ", str

Kết quả khi mình nhập "Hello Python!" là:

Nhap dau vao cua ban: Hello Python

Dau vao da nhan la : Hello Python

**Ghi chú**: Hàm raw\_input() trả về một chuỗi. Vì thế trong trường hợp một biểu thức cần được ước lượng, thì nó phải ép kiểu sang kiểu dữ liệu sau của nó. Bạn theo dõi một số ví dụ dưới đây.

Ví dụ 1:

prn=int(raw\_input("Trang VietJack"))

r=int(raw\_input("Thu Tu"))

t=int(raw\_input("Vi Tri"))

si=(prn\*r\*t)/100

print "VietJack Chao Ban ",si

Kết quả là:

>>>

Trang VietJack1000

Thu Tu10

Vi Tri2

VietJack Chao Ban 200

>>>

Ví dụ 2:

name=raw\_input("Nhap ten ban ")

math=float(raw\_input("Nhap diem mon Toan"))

physics=float(raw\_input("Nhap diem mon Vat Ly"))

chemistry=float(raw\_input("Nhap diem mon Hoa Hoc"))

rollno=int(raw\_input("Nhap mssv"))

print "Welcome ",name

print "MSSV cua ban la ",rollno

print "Diem mon Toan la ",math

print "Diem mon Vat Ly la ",physics

print "Diem mon Hoa Hoc la ",chemistry

print "Diem trung binh la ",(math+physics+chemistry)/3

Kết quả là:

>>>

Nhap ten ban Hoang

Nhap diem mon Toan7.68

Nhap diem mon Vat Ly7.14

Nhap diem mon Hoa Hoc8.84

Nhap mssv0987645672

Welcome Hoang

MSSV cua ban la 987645672

Diem mon Toan la 7.68

Diem mon Vat Ly la 7.14

Diem mon Hoa Hoc la 8.84

Diem trung binh la 7.8866666667

>>>

## **Làm việc với File trong Python**

Python cung cấp nhiều cách tiện lợi để bạn làm việc với file. Ở trên, bạn đã đọc dữ liệu từ Standard Input và ghi dữ liệu tới Standard Output. Bây giờ chúng ta tìm hiểu cách sử dụng các file dữ liệu thực sự. Một file là một nơi lưu trữ ngoại vi trên hard disk, tại đó dữ liệu có thể được lưu trữ và thu nhận. Dưới đây là các hoạt động trên File:

## **Mở file trong Python**

Trước khi làm việc với bất cứ File nào, bạn phải mở File đó. Để mở một File, Python cung cấp hàm open(). Nó trả về một đối tượng File mà được sử dụng với các hàm khác. Với File đã mở, bạn có thể thực hiện các hoạt động đọc, ghi, … trên File đó. Cú pháp của hàm open() là:

doi\_tuong\_file = open(ten\_file [, access\_mode][, buffer])

Ở đây,

* **ten\_file** là tên File bạn muốn truy cập.
* **access\_mode** xác định chế độ của File đã được mở. Có nhiều mode sẽ được trình bày trong phần dưới. Bạn nên xác định mode này phụ thuộc vào các hoạt động mà bạn muốn thực hiện trên File đó. Chế độ truy cập mặc định là read.
* **buffer** Nếu buffer được thiết lập là 0, nghĩa là sẽ không có trình đệm nào diễn ra. Nếu xác định là 1, thì trình đệm dòng được thực hiện trong khi truy cập một File. Nếu là số nguyên lớn hơn 1, thì hoạt động đệm được thực hiện với kích cỡ bộ đệm đã cho. Nếu là số âm, thì kích cỡ bộ đệm sẽ là mặc định (hành vi mặc định).

## **Đóng một File trong Python**

Khi bạn đã thực hiện xong các hoạt động trên File thì cuối cùng bạn cần đóng File đó. Python tự động đóng một File khi đối tượng tham chiếu của một File đã được tái gán cho một file khác. Tuy nhiên, sử dụng phương thức close() để đóng một file là một sự thực hành tốt cho bạn. Phương thức close() có cú pháp như sau:

fileObject.close();

## **Đọc một File trong Python**

Để đọc một File, bạn sử dụng phương thức read() trong Python. Cú pháp là:

doi\_tuong\_file.read(giatri);

string = fileopen.read()

print(string)

Ở đây, value là số byte để được đọc từ file đã mở. Phương thức này bắt đầu đọc từ phần đầu file và nếu bạn không cung cấp tham số value thì phương thức này cố gắng đọc nhiều dữ liệu nhất có thể, có thể tới cuối File.

Hoặc có thể dùng vòng lặp để in giá trị

fileopen = open("File Sample.txt","r+")

for index in fileopen:

print(index)

## **Phương thức readline**

### **Cú pháp:**

<File>.**readline(size=-1)**

**Công dụng:**Với parameter size thì hoàn toàn tương tự như phương thức **read**.

* Khác biệt ở chỗ, phương thức readline chỉ đọc một dòng có nghĩa là đọc tới khi nào gặp newline hoặc hết file thì ngừng.
* Con trỏ file cũng sẽ đi từ dòng này qua dòng khác.
* Kết quả đọc được trả về dưới dạng một chuỗi.
* Nếu không đọc được gì, phương thức sẽ trả về một chuỗi có độ dài bằng

**Ví dụ:**

>>> fobj = open('kteam.txt')

>>> fobj.readline()

'How Kteam\n'

>>> fobj.readline(10)

'Free Educa'

>>> fobj.readline()

'tion\n'

>>> fobj.readline()

'\n'

>>> fobj.readline()

'Share to better\n'

>>> fobj.close()

## **Phương thức readlines**

### **Cú pháp:**

<File>.**readlines(hint=-1)**

Ở mức độ cơ bản, ta không phải quan tâm đến parameter hint.

**Công dụng:** Phương thức này sẽ đọc toàn bộ file, sau đó cho chúng vào một list. Với các phần tử trong list là mỗi dòng của file.

* Con trỏ file sẽ được đưa  tới cuối file. Khi đó, nếu bạn tiếp tục dùng readlines. Bạn sẽ nhận được một list rỗng.

**Ví dụ:**

>>> fobj = open('kteam.txt')

>>> list\_content = fobj.readlines()

>>> list\_content

['How Kteam\n', 'Free Education\n', '\n', 'Share to better\n', '\n', "print('hello world!')\n"]

>>> list\_content[2]

'\n'

>>> list\_content[-1]

"print('hello world!')\n"

>>> fobj.close()

0

1

2

3

4

5

6

7

8

9

## **Đọc file bằng constructor nhận iterable**

Như đã nói, file object nhận được từ hàm open cũng là một **iterable**.

Thế nên, ta có thể sử dụng constructor list

>>> fobj = open('kteam.txt')

>>> list\_content = list(fobj)

>>> list\_content

['How Kteam\n', 'Free Education\n', '\n', 'Share to better\n', '\n', "print('hello world!')\n"]

>>> fobj.close()

0

1

2

3

4

5

Và cũng có thể là Tuple.

>>> fobj = open('kteam.txt')

>>> tup\_content = tuple(fobj)

>>> tup\_content

('How Kteam\n', 'Free Education\n', '\n', 'Share to better\n', '\n', "print('hello world!')\n")

>>> fobj.close()

0

1

2

3

4

5

Các constructor này cũng sẽ đưa con trỏ file xuống cuối file.

## **Ghi tới một File trong Python**

Phương thức write() được sử dụng để ghi bất kỳ chuỗi nào tới một File đã mở. Bạn chú ý là phương thức write này không thêm một ký tự newline (dòng mới) ('\n') vào cuối chuỗi. Cú pháp của write() là:

doi\_tuong\_file.write(string);

Dưới đây là chương trình ví dụ để đọc và ghi dữ liệu từ một File trong Python:

obj=open("abcd.txt","w")

obj.write("Python xin chao cac ban")

obj.close()

obj1=open("abcd.txt","r")

s=obj1.read()

print s

obj1.close()

obj2=open("abcd.txt","r")

s1=obj2.read(20)

print s1

obj2.close()

Kết quả là:

>>>

Python xin chao cac ban

Chao mung ban den voi the gioi Python

>>>

## **Các thuộc tính của File trong Python**

Đối tượng File có các thuộc tính sau:

|  |  |
| --- | --- |
| **Thuộc tính** | **Miêu tả** |
| file.closed | Trả về true nếu file đã được đóng, nếu không là false |
| file.mode | Trả về chế độ truy cập nào mà file đã mở với |
| file.name | Trả về tên file |
| file.softspace | Trả về false nếu space được yêu cầu tường minh với print, nếu không là true |

### **Ví dụ**

Chúng ta tạo *foo.txt* có nội dung sau:

# Mo mot file

fo = open("foo.txt", "wb")

fo.write( "Python la mot ngon ngu lap trinh tuyet voi.\nMinh cung nghi nhu the!!\n");

# Dong file da mo

fo.close()

Giờ chúng ta kiểm tra các thuộc tính của nó:

# Mo mot file

fo = open("foo.txt", "wb")

print "Ten cua file la: ", fo.name

print "File da duoc dong hay chua : ", fo.closed

print "Che do mode la : ", fo.mode

print "Softspace la : ", fo.softspace

Kết quả là:

Ten cua file la: foo.txt

File da duoc dong hay chua : False

Che do mode la : wb

Softspace la : 0

## **Các chế độ truy cập (mode) của File trong Python**

File có thể được mở với các chế độ truy cập khác nhau. File có thể được mở trong Text Mode hoặc Binary Mode. Bảng dưới liệt kê và giới thiệu các chế độ này:

|  |  |
| --- | --- |
| **Mode** | **Miêu tả** |
| r | Mở file trong chế độ đọc, đây là chế độ mặc định. Con trỏ tại phần bắt đầu của File |
| rb | Mở file trong chế độ đọc cho định dạng nhị phân, đây là chế độ mặc định. Con trỏ tại phần bắt đầu của File |
| r+ | Mở file để đọc và ghi. Con trỏ tại phần bắt đầu của File |
| rb+ | Mở file để đọc và ghi trong định dạng nhị phân. Con trỏ tại phần bắt đầu của File |
| w | Mở File trong chế độ ghi. Nếu file đã tồn tại, thì ghi đè nội dung của file đó, nếu không thì tạo một file mới |
| wb | Mở File trong chế độ ghi trong định dạng nhị phân. Nếu file đã tồn tại, thì ghi đè nội dung của file đó, nếu không thì tạo một file mới |
| w+ | Mở file để đọc và ghi. Nếu file tồn tại thì ghi đè nội dung của nó, nếu file không tồn tại thì tạo một file mới để đọc và ghi |
| wb+ | Mở file để đọc và ghi trong định dạng nhị phân. Nếu file tồn tại thì ghi đè nội dung của nó, nếu file không tồn tại thì tạo một file mới để đọc và ghi |
| a | Mở file trong chế độ append. Con trỏ là ở cuối file nếu file này đã tồn tại. Nếu file không tồn tại, thì tạo một file mới để ghi |
| ab | Mở file trong chế độ append trong chế độ nhị phân. Con trỏ là ở cuối file nếu file này đã tồn tại. Nếu file không tồn tại, thì tạo một file mới để ghi |
| a+ | Mở file trong để đọc và append. Con trỏ file tại cuối nếu file đã tồn tại. Nếu không tồn tại thì tạo một file mới để đọc và ghi |
| ab+ | Mở file trong để đọc và append trong định dạng nhị phân. Con trỏ file tại cuối nếu file đã tồn tại. Nếu không tồn tại thì tạo một file mới để đọc và ghi |

## **Thay tên file trong Python**

Phương thức *rename()* trong os Module được sử dụng để thay tên file. Phương thức này nhận hai tham số là tên file cũ và tên file mới.

### **Cú pháp**

os.rename(ten\_file\_hien\_tai, ten\_file\_moi)

Ví dụ sau thay tên *test1.txt* thành *test2.txt*:

import os

# Thay ten tu test1.txt thanh test2.txt

os.rename( "test1.txt", "test2.txt" )

## **Xóa file trong Python**

Bạn có thể sử dụng phương thức *remove()* của os Module để xóa các file với tham số là tên file bạn cần xóa.

### **Cú pháp**

os.remove(ten\_file)

Ví dụ sau sẽ xóa *test2.txt*:

import os

# Xoa test2.txt

os.remove("text2.txt")

## **Vị trí File trong Python**

Phương thức**tell()**nói cho bạn biết vị trí hiện tại bên trong file. Nói cách khác, việc đọc và ghi tiếp theo sẽ diễn ra trên các byte đó.

Phương thức **seek(offset[, from])** thay đổi vị trí hiện tại bên trong file. Tham số*offset* chỉ số byte để được di chuyển. Tham số **from** xác định vị trí tham chiếu mà từ đó byte được di chuyển.

Nếu *from* được thiết lập là 0 nghĩa là sử dụng phần đầu file như là vị trí tham chiếu và 1 nghĩa là sử dụng vị trí hiện tại như là vị trí tham chiếu và nếu là 2 thì sử dụng phần cuối file như là vị trí tham chiếu.

**Với Python 3.X. Một text file sẽ chỉ được sử dụng whence = 0 còn whence = 1 hoặc whence = 2 chỉ sử dụng với binary file**

### **Ví dụ**

Sử dụng foo.txt đã tạo ở trên để minh họa các hàm tell và seek:

# Mo mot file

fo = open("foo.txt", "r+")

str = fo.read(10);

print "Chuoi da doc la : ", str

# Kiem tra con tro hien tai

position = fo.tell();

print "Con tro file hien tai : ", position

# Dat lai vi tri con tro tai vi tri ban dau mot lan nua

position = fo.seek(0, 0);

str = fo.read(10);

print "Chuoi da doc la : ", str

# Dong file da mo

fo.close()

Kết quả là:

Chuoi da doc la : Python is

Con tro file hien tai : 10

Chuoi da doc la : Python la

## **Thư mục trong Python**

Tất cả file được chứa trong các thư mục đa dạng và Python cũng cung cấp rất nhiều phương thức để xử lý các hoạt động đa dạng liên quan tới thư mục. **os** Module có một số phương thức giúp bạn tạo, xóa, và thay đổi các thư mục.

## **Phương thức mkdir() trong Python**

Bạn có thể sử dụng phương thức mkdir() của os Module để tạo các thư mục trong thư mục hiện tại. Bạn cần cung cấp một tham số là tên thư mục cho phương thức này.

### **Cú pháp**

os.mkdir("thu\_muc\_moi")

Ví dụ sau tạo một thư mục *test* trong thư mục hiện tại.

import os

# Tao mot thu muc la "test"

os.mkdir("test")

## **Phương thức chdir() trong Python**

Bạn có thể sử dụng phương thức chdir() để thay đổi thư mục hiện tại. Phương thức chdir() nhận một tham số là tên của thư mục bạn muốn tới từ thư mục hiện tại.

### **Cú pháp**

os.chdir("thu\_muc\_moi")

Ví dụ sau tới thư mục /home/newdir.

import os

# Thay doi mot thu muc toi "/home/newdir"

os.chdir("/home/newdir")

## **Phương thức getcwd() trong Python**

Phương thức getcwd() hiển thị thư mục đang làm việc hiện tại.

### **Cú pháp**

os.getcwd()

Ví dụ sau hiển thị thư mục đang làm việc hiện tại.

import os

# Lenh nay se cung cap vi tri thu muc hien tai

os.getcwd()

## **Phương thức rmdir() trong Python**

Phương thức rmdir() xóa thư mục mà có tên được truyền như là một tham cố cho phương thức này.

Trước khi xóa thư mục, tất cả nội dung trong nó nên được xóa.

### **Cú pháp**

os.rmdir('ten\_thu\_muc')

Ví dụ sau sẽ xóa thư mục /tmp/test. Bạn phải cung cấp tên đầy đủ của thư mục, nếu không phương thức này sẽ không tìm thấy thư mục đó và sẽ không có hoạt động xóa diễn ra.

import os

# Xoa thu muc "/tmp/test" .

os.rmdir( "/tmp/test" )

## **Các phương thức xử lý File và thư mục trong Python**

Đối tượng File và OS cung cấp rất nhiều phương thức tiện ích để xử lý và thao tác với File và thư mục trên hệ điều hành Windows và Unix. Bạn truy cập đường link sau để tìm hiểu các phương thức này.

* [Đối tượng File](https://vietjack.com/python/phuong_thuc_xu_ly_file_io_trong_python.jsp): cung cấp các phương thức để thao tác File.
* [os Module](https://vietjack.com/python/phuong_thuc_thao_tac_file_io_trong_python.jsp): cung cấp rất nhiều phương thức để thao tác File và thư mục.

# Câu lệnh with

Cấu trúc cơ bản của câu lệnh with là

**with** expression [**as** variable]:

**with**-block

* Nhớ rằng with-block nằm thụt vào so với dòng with expression (theo chuẩn PEP8 là 4 space và là dùng space không dùng tab)
* Câu lệnh này liên quan đến phương thức **\_\_enter\_\_**và **\_\_exit\_\_**của đối tượng. Do đó, ở đây Kteam sẽ nói cơ bản khi sử dụng file.
* Đặc điểm của câu lệnh with khi sử dụng với file là. Khi kết thúc with-block. File sẽ được đóng.

>>> **with** open('kteam.txt') **as** fobj:

... data = fobj.read()

...

>>> data

"How Kteam\nFree Education\n\nShare to better\n\nprint('hello world!')\n"

>>> fobj.read() # không thể đọc file, vì file đã đóng

Traceback (most recent call last):

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

* ValueError: I/O operation on closed file.

## **Module pickle**

Bây giờ chúng ta sẽ đọc và ghi file theo kiểu dữ liệu nhị phân, ví dụ về các dạng dữ liệu nhị phân là hình ảnh, nhạc, phim… ngay cả các đoạn text thông thường cũng có cách để ghi theo kiểu nhị phân. Trong ví dụ dưới đây, chúng ta sẽ ghi dữ liệu của một đối tượng vào file. Để ghi dữ liệu dạng nhị phân thì chúng ta dùng tới module pickle. Khi ghi lên file, pickle sẽ chuyển các đoạn text thành các dạng kí hiệu, khi đọc, nó sẽ chuyển ngược lại về text cho chúng ta.

|  |  |
| --- | --- |
| pickle.py | |
| 1  2  3  4  5  6  7  8  9  10  11  12  13  14  15  16  17  18  19  20  21  22  23  24  25  26  27  28 | import pickle    class Person:     def \_\_init\_\_(self, name, age):        self.name = name        self.age = age       def getName(self):        return self.name       def getAge(self):        return self.age      person = Person('Monica', 15)  print (person.getName())  print (person.getAge())    f = open('monica', 'wb')  pickle.dump(person, f)  f.close()    f = open('monica', 'r')  monica = pickle.load(f)  f.close()    print (monica.getName())  print (monica.getAge()) |

Trong đoạn code trên, chúng ta định nghĩa lớp Person và tạo một đối tượng với lớp này. Sau đó chúng ta ghi dữ liệu của đối tượng này vao file bằng phương thức dump(). Để đọc dữ liệu nhị phân thì chúng ta dùng phương thức load().