TRƯỜNG ĐẠI HỌC HỌC VĂN LANG

**KHOA CÔNG NGHỆ THÔNG TIN**



**TÀI LIỆU GIẢNG DẠY**

**KỸ THUẬT LẬP TRÌNH**

2020

|  |  |  |
| --- | --- | --- |
|  |  | Lập trình cơ bản với Python |
|  |  |  |

**Chương 6: TẬP TIN**

**Mục tiêu:**

* *Lưu trữ dữ liệu lên tập tin*
* *Truy xuất dữ liệu trên tập tin*
* *Sử dụng dữ liệu sau khi đọc*

**Nội dung chính:**

* + *Đóng mở tập tin*
  + *Ghi dữ liệu lên tập tin*
  + *Đọc dữ liệu từ tập tin*
  + *Thao tác dữ liệu*

1. **Giới thiệu tập tin**

Tập tin được sử dụng để lưu trữ dữ liệu lên các thiết bị như ổ cứng, mạng… Trong máy tính, tập tin được chia làm hai loại: tập tin văn bản và tập tin nhị phân.

Nói một cách vắn tắt, tập tin văn bản chứa nội dung là các kí tự mà con người có thể đọc được. Tập tin nhị phân chứa nội dung là các byte thô mà chỉ có chương trình đặc thù mới hiểu và thể hiện nội dung tương ứng.

Đối với tập tin văn bản, Python hỗ trợ cả chuẩn ASCII và Unicode. Cách truy xuất ở mỗi dạng cũng có phần khác nhau. Để đơn giản và phù hợp với ở mức độ môn học, chúng ta tạm thời gác Unicode qua một bên và tập trung chủ yếu vào ACII.

Đối với tập tin nhị phân, chúng ta chỉ lướt nhẹ qua về cách mở và cũng không đi sâu vào.

**1. Trình tự thao tác với tập tin**

Khi thao tác với tập tin, chúng ta thường trải qua các giai đoạn:



Lập trình cơ bản với Python

* Mở tập tin với một đường dẫn xác định đến nơi đang chứa tập tin và kèm với đó là các mode mà chúng ta muốn thao tác.
* Kế tiếp, ta thực hiện các thao tác mà mình muốn làm với tập tin như đọc, ghi, di chuyển trong tập tin.
* Cuối cùng là đóng tập tin lại để giải phóng tài nguyên cũng như hoàn tất quá trình ghi lên ổ cứng.

Trong 3 thao tác trên, thường lập trình viên quên viết thao tác cuối cùng (đóng tập tin). Python có chức năng phát hiện và tự thực hiện các công việc cần thiết để đóng tập tin khi lập trình viên quên. Điều này giúp cho chương trình tránh các lỗi xảy ra. Tuy nhiên, thói quen lập trình tốt vẫn là nhớ đóng tập tin lại khi xài xong.

1. **Mở tập tin**

Python hỗ trợ hàm open() để mở tập tin:



1. = open(**"demofile.txt"**)

Ví dụ trên sẽ mở một file có tên là demofile.txt tại nơi đang chứa code python. Khi mở thành công, Python sẽ trả về một đối tượng file và chúng ta đặt tên cho nó là f. Ta có thể chỉ định đường dẫn tuyệt đối trong chuỗi tập tin nhưng cách này không khuyến khích.

Với cách mở trên, file được mở mặc định với chế độ mở là văn bản và việc mở để dành cho đọc. Trong trường hợp muốn thực hiện các mục đích khác như mở để ghi, mở tập tin nhị phân, … ta cần chỉ định thêm đối số thứ hai trong hàm trên. Đối số ở dạng chuỗi.



1. = open(**"demofile.txt"**, **"rt"**)

Trong ví dụ, chúng ta chỉ định rõ mở file để đọc và tập tin ở dạng văn bản.

Các cấu hình:



Lập trình cơ bản với Python

* “r”: đọc (mặc định)
* “a”: mở ghi tiếp (tạo mới nếu file chưa có)
* “w”: viết (tạo mới nếu file chưa có)
* “x”: tạo mới (nếu có file thì báo lỗi)

Có thể bổ sung thêm loại tập tin:

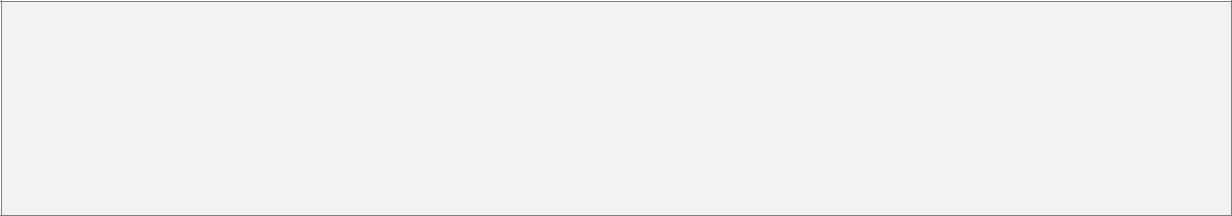
* + “t”: loại tập tin văn bản (mặc định)
  + “b”: loại tập tin nhị phân

1. **Đọc tập tin**

Các hàm mà Python hỗ trợ cho việc đọc:

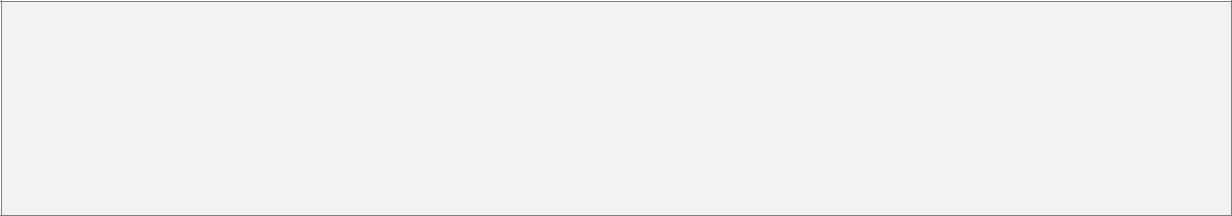
* + read(): đọc toàn bộ nội dung của file vào một biến chuỗi
  + read(N): đọc N kí tự (hoặc byte) vào biến chuỗi
  + readline(): đọc 1 dòng (bao gồm cả dấu xuống dòng \n) vào biến chuỗi.
  + readlines(): đọc toàn bộ file vào một danh sách các dòng.

Ví dụ:



1. = open(**"demofile.txt"**, **"r"**) print(f.read()) print(f.read(5)) print(f.readline()) print(f.readlines())

Để có thể đọc lần lượt từng dòng cho đến khi hết file, ta có thể làm như sau:



1. = open(**'script2.py'**) **while True**:

line = f.readline() **if not** line: **break**

print(line.upper(), end=**''**)

Trong ví dụ, ta kiểm tra biến line để biết dấu hiệu kết thúc file. Khi đạt đến vị trí cuối, biến line sẽ rỗng. Ngoài cách trên, ta có thể dùng cú pháp đơn giản hơn:

Lập trình cơ bản với Python



* 1. = open(**"demofile.txt"**, **"rt"**) **for** line **in** f:

print(line)

1. **Ghi tập tin**

Các hàm hỗ trợ cho việc viết nội dung lên tập tin:

- write(aString): viết một chuỗi lên tập tin

- writelines(aList): viết tất cả các dòng trong danh sách lên tập tin

Ví dụ:

* 1. = open(**"demofile.txt"**, **"w"**)



f.write(**"Hello World"**)

Ta có thể viết tiếp một tập tin đang sẵn có:



1. = open(**"demofile.txt"**, **"a"**)

f.write(**"Now the file has one more line!"**

Lưu ý: để viết một đối tượng không phải là chuỗi lên file, ta phải chuyển đổi (convert) đối tượng này thành chuỗi trước khi ghi. Phương thức write() không tự động làm cho chúng ta các công việc này.



X, Y, Z = 43, 44, 45

S = **'Spam'**

D = {**'a'**: 1, **'b'**: 2}

L=[1,2,3]

1. = open(**'datafile.txt'**, **'w'**) F.write(S + **'\n'**) F.write(**'%s,%s,%s\n'** % (X, Y, Z)) F.write(str(L) + **'$'** + str(D) + **'\n'**) F.close()

Một trường hợp ngoại lệ là khi dùng hàm writelines() cho phép ghi một danh sách lên file. Tuy nhiên, mỗi phần tử trong danh sách cũng phải ở dạng chuỗi.

Lập trình cơ bản với Python

1. **Đóng tập tin**

Một thói quen lập trình tốt là nên đóng file lại khi không còn sử dụng nữa.

myfile = open(**'myfile.txt'**, **'w'**) myfile.write(**'hello text file\n'**) myfile.close()

1. **Xóa tập tin**

Để xóa tập tin ta phải sử dụng một import một module của hệ thống và chạy thực thi hàm remove():

**import** os

os.remove(**"demofile.txt"**)

Lưu ý, khi file không tồn tại, lệnh này sẽ phát sinh ra lỗi. Để khắc phục nguy cơ này ta nên kiểm ta file có tồn tại không trước khi xóa.

**import** os

**if** os.path.exists(**"demofile.txt"**):os.remove(**"demofile.txt"**)

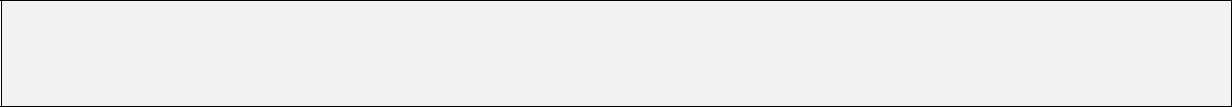
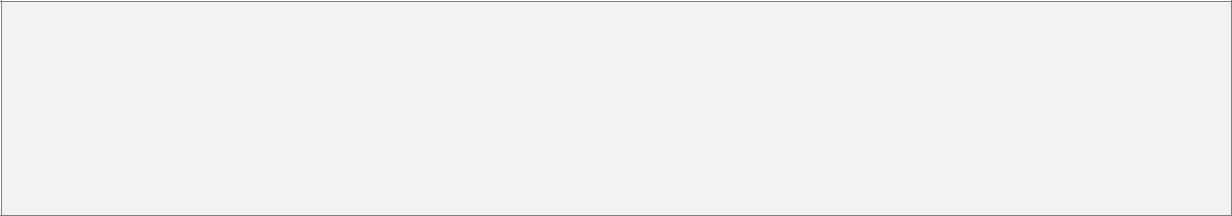
**else**:

print(**"The file does not exist"**)

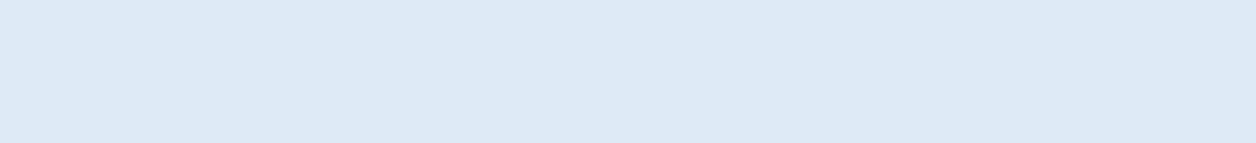
Python còn cung cấp cho chúng ta thêm hàm xóa thư mục:

**import** os

os.rmdir(**"myfolder"**)



Lập trình cơ bản với Python



**BÀI TẬP**

****

*Bài 10.1 Viết chương trình tạo một tập tin văn bản với tên và nội dung do người dùng chỉ định.*

Ví dụ:

Tên tập tin: abc.txt

Nội dung: “Hello World”



*Bài 10.2 Viết chương trình đọc nội dung của một tập tin văn bản cho trước. Tên tập tin do người dùng chỉ định. Trong tập tin có nhiều dòng chữ.*

Ví dụ:

Tên tập tin: abc.txt

Nội dung:

Hello World

I Love You Very Much

I am here. Where are you?



*Bài 10.3 Viết chương trình đọc một danh sách các số được ghi trong một tập tin văn bản, với mỗi số cách nhau bằng dấu khoảng trắng. Hiển thị danh sách ra màn hình và tính tổng các số đó.*

Ví dụ:

Tên tập tin: abc.txt

Nội dung: 1 2 3 4 5.5 6.5

Tổng là: 22



Lập trình cơ bản với Python



*Bài 10.4 Viết chương trình ghi thêm nội dung vào một tập tin văn bản đang có trên hệ thống.*

Ví dụ:

Tên tập tin: abc.txt

Nội dung đang có: “Hello”

Nội dung thêm: “my love”



*Bài 10.5 Viết chương trình xóa một tập tin đang có trên hệ thống.*

Ví dụ:

Tên tập tin: abc.txt

Đã xóa tập tin

Tên tập tin: xyz.txt

Tập tin không tìm thấy



*Bài 10.6 Viết chương trình xóa toàn bộ tập tin trong một thư mục có sẵn. Lưu ý, không xóa thư mực*

Ví dụ:

Thư mục: myFolder

Đã xóa tất cả tập tin trong thư mục



*Bài 10.6 Viết chương trình nối 2 tập tin văn bản lại thành một tập tin duy nhất.*

Ví dụ:

Tập tin: abc.txt và def.txt

Tập tin đã nối là abcdef.txt

Lập trình cơ bản với Python



*Bài 10.6 Viết chương trình tách tập tin văn bản thành các tập*

*tin sao cho mỗi tập tin không quá x kí tự và số lượng tập tin*

*được tạo là ít nhất. Với x là số kí tự do người dùng chỉ định.*

Ví dụ:

Tập tin: abc.txt

Tập tin đã nối là a.txt, b.txt, c.txt, d.txt



*Bài 10.7 Viết chương trình ghi một danh sách sinh viên lên tập tin sau đó đọc lại vào một danh sách khác. Kiểm tra hai danh sách có giống nhau không.*

*Thông tin sinh viên bao gồm:*

* *Mã SV*
* *Họ tên*
* *Năm sinh*
* *Giới tính*
* *Điểm Toán*
* *Điểm Lý*
* *Điểm Hóa*

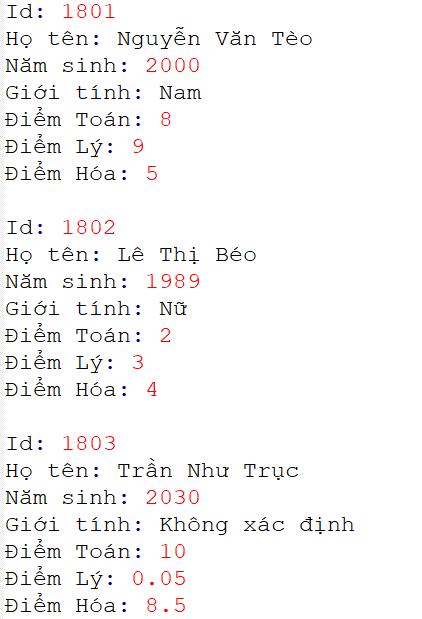
*Mỗi sinh viên sẽ cách nhau một dòng trống. Danh sách phải có ít nhất 5 sinh viên.*

Lập trình cơ bản với Python

Ví dụ:

Tên tập tin: abc.txt

Nội dung:



**Chương 7 : TỪ ĐIỂN**

**Mục tiêu:**

* *Tổ chức dữ liệu với kiểu dữ liệu Từ Điển*
* *Thực hiện các thao tác trên Từ Điển*
* *Vận dụng các hàm xây dựng sẵn của Từ Điển*

**Nội dung chính:**

* *Khai báo và sử dụng kiểu dữ liệu Từ Điển*
* *Chuyển đổi kiểu dữ liệu khác thành Từ Điển*
* *Duyệt trong Từ Điển*
* *Các lệnh thường dùng*

1. **Khai báo dữ liệu**

Từ điển là một tập hợp dữ liệu tương tự như kiểu danh sách nhưng chúng không có thứ tự, được đánh chỉ mục và có thể thay đổi dữ liệu bên trong nó. Điều này đồng nghĩa với việc ta có thể chứa và truy xuất dữ liệu bởi các khóa thay vì vị trí như trong danh sách (list).

Mỗi phần tử trong từ điển là một cặp (khóa, giá trị). Khóa được sử dụng để có thể truy xuất giá trị dữ liệu sau này.

Để tạo từ điển, ta đặt các phần tử trong dấu ngoặc nhọn, mỗi phần tử cách nhau bằng dấu phẩy. Ví dụ:

mydict = { **"brand"**: **"Ford"**, **"model"**: **"Mustang"**, **"year"**:1964

}

Trong ví dụ trên ta đã tạo ra một từ điển có tên là mydict chứa 3 phần tử với mỗi phần tử là một cặp khóa:giá trị. Khóa và giá trị trong từ điển không nhất thiết chỉ là các chuỗi. Khóa có thể là bất kì kiểu nào như số nguyên, số thực,



Lập trình cơ bản với Python

* Tương tự, giá trị có thể là một đối tượng bất kì trong Python. Điều này làm cho từ điển trở thành một cấu trúc khá linh hoạt.

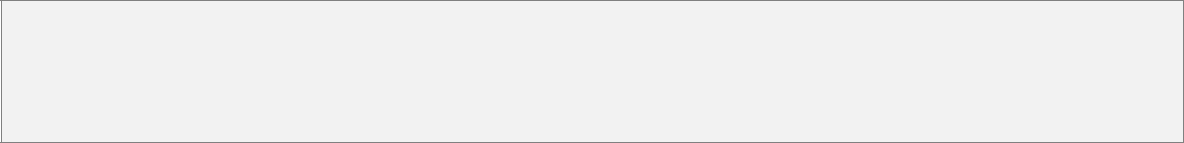


table = {1975: **'Holy Grail'**,

1979: **'Life of Brian'**, *# Keys are integers, not strings*

1983: **'The Meaning of Life'**}

1. **Kiểu dữ liệu động**

Từ điển là một kiểu dữ liệu động. Nó có thể mở rộng thêm các phần tử sau khi đã tạo cũng như xóa bớt phần tử khi cần thiết mà không tạo ra một từ điển mới.

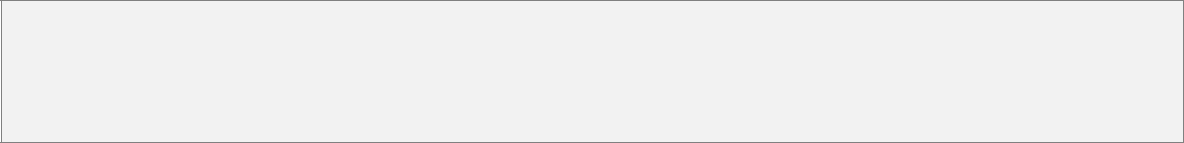
D = {} *# Assign by keys dynamically* D[**'name'**] = **'Bob'**

D[**'age'**] = 40

Trong ví dụ trên, đầu tiên ta tạo ra một từ điển rỗng. Sau đó, để thêm phần từ vào ta thực hiện phép gán như thể phần tử đã tồn tại. Khi thực hiện lệnh gán, bên dưới đó, từ điển sẽ tự động thêm khóa mới nếu như phần tử đó chưa tồn tại trong từ điển.

1. **Tạo từ điển từ kiểu dữ liệu khác**

Ta có thể tạo từ điển từ các dữ liệu khác đã tồn tại. Khi nhận một dữ liệu từ kiểu khác, từ điển sẽ tự động phân tích để tìm ra cặp khóa và giá trị, sau đó tiến hành thêm vào từ điển. Để thực hiện việc này, từ điển sử dụng hàm dict và nhận vào đối số là các cặp khóa, giá trị hoặc từ một danh sách các bộ khác.



dict(name=**'Bob'**, age=40) *# dict keyword argument form*

dict([(**'name'**, **'Bob'**), (**'age'**, 40)]) *# dict key/value tuples form*

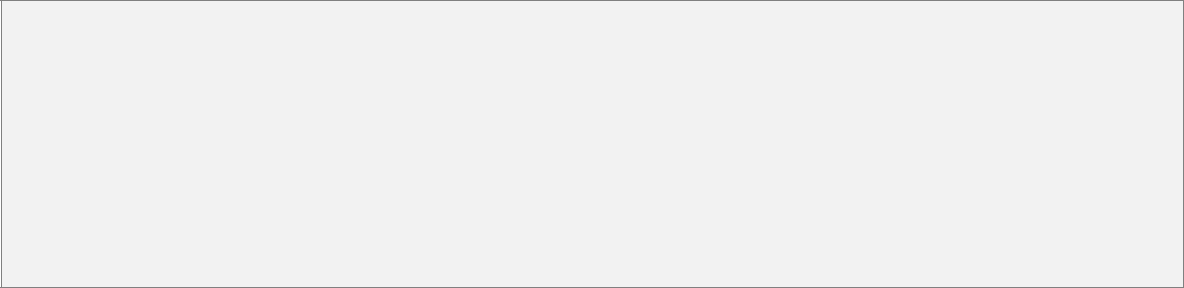
Lập trình cơ bản với Python

1. **Truy xuất phần tử**

Các phần tử được truy xuất thông qua khóa của chúng. Cú pháp truy xuất giống như truy xuất trong danh sách (list), chỉ khác là thay vì vị trí trong danh sách thì lúc này sẽ là các khóa.

mydict = { **"brand"**: **"Ford"**, **"model"**: **"Mustang"**, **"year"**:1964

}



x = mydict[**"model"**] *#access item with key "model"*

print(x) *#Ford*

Ngoài ra, Python còn cho phép chúng ta lấy phần tử thông qua các hàm của từ điển.

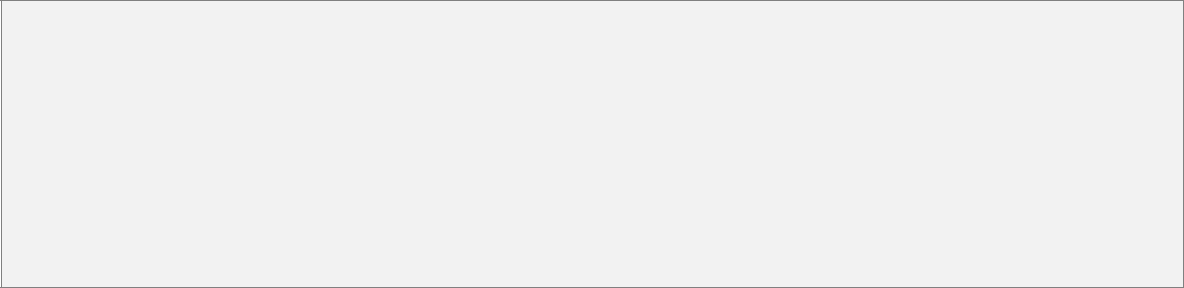


mydict.get(**'brand'**)

Một điểm lợi khi dùng hàm là khi khóa không có trong từ điển, nếu theo cách truy xuất thông thường sẽ gây cho chương trình bị lỗi. Khi dùng hàm get, ta sẽ nhận giá trị trả về là None, chương trình vẫn thực hiện bình thường mà không bị vấn đề.

1. **Thay đổi giá trị**

Tương tự như cách truy xuất phần tử trong từ điển, để thay đổi giá trị bằng cách thực hiện phép gán.



mydict = {

**"brand"**: **"Ford"**,

**"model"**: **"Mustang"**,

**"year"**:1964

}

mydict[**"year"**] = 2018

Trong ví dụ, ta đã thay đổi phần tử có khóa “year” với giá trị mới là 2018.

Lưu ý: khi khóa không tồn tại, từ điển sẽ tạo khóa này và bổ sung giá trị vào.



Lập trình cơ bản với Python

1. **Duyệt trong từ điển**

Các cú pháp lặp đều có thể sử dụng để duyệt qua các phần tử trong từ điển.

**for** x **in** mydict:print(mydict[x])

Câu lệnh trên thực hiện duyệt từng khóa x trong từ điển và in giá trị tương ứng với khóa đó. Ngoài cách duyệt qua từng khóa, ta có thể duyệt qua từng giá trị trong từ điển bằng cách:

**for** x **in** mydict.values():print(x)

Như vậy, lúc này x chính là từng giá trị trong từ điển. Trong trường hợp muốn có cả khóa và giá trị, ta có thể sử dụng cú pháp sau:

**for** x, y **in** mydict.items():print(x, y)

1. **Thêm, xóa phần tử**

Thêm phần tử vào mảng ta có thể thực hiện tương tự như phép gán giá trị vào một khóa mới. Từ điển sẽ tự làm các công việc còn lại.

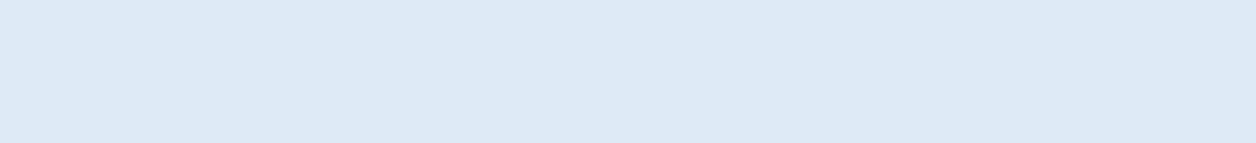
Để xóa phần tử khỏi từ điển, ta có thể sử dụng từ khóa del, hàm pop(), hàm clear()...



|  |  |  |
| --- | --- | --- |
| mydict = | { |  |
| **"brand"**: **"Ford"**, | |  |
| **"model"**: **"Mustang"**, | |  |
|  | |  |
| **"year"**:1964 | |  |
| } |  |  |
| **del** mydict[**"model"**] | | *#sử dụng từ khóa del* |
| mydict.pop(**"model"**) | | *#sử dụng hàm pop* |
| mydict.clear() | | *#đưa từ điển về trạng thái rỗng {}* |
| **del** mydict |  | *#xóa toàn bộ từ điển (mydict sẽ* |
| *không tồn tại nữa)* | |  |
|  |  |  |



Lập trình cơ bản với Python



**BÀI TẬP**

****

Bài 8.1 Viết một đoạn lệnh cho phép người dùng thêm một khóa vào từ điển. Trong đó khóa và giá trị do người dùng chỉ định:

Ví dụ:

Từ điển trước khi thêm: {0: 10, 1: 20}

Phần tử cần thêm: key = 2, value = 30

Từ điển sau khi thêm (không cần đúng thứ tự như trong ví dụ):

{0: 10, 1: 20, 2: 30}



*Bài 8.2* Viết đoạn lệnh để kiểm tra xem một khóa có nằm trong từđiển không. Nếu có, xuất ra “Tồn tại giá trị trong từ điển với giá trị là v” với v là giá trị có trong từ điển. Nếu không, xuất ra là “Không có khóa k trong từ điển” với k là khóa đã nhập.

Ví dụ: {“a”: 1, “b”: 2}

Tìm “a”

Tồn tại giá trị trong từ điển với giá trị là 1

*Bài 8.3 Viết chương trình xóa một phần tử với khóa cho trước khỏi từ điển. Xuất lại từ điển sau khi xóa.*

Ví dụ: {“a”: 1, “b”: 2, “c”: 3.5, “d”: “hello”}

Xóa phần tử với khóa: “c”

Từ điển sau khi xóa (không cần đúng thứ tự như ví dụ):

{“a”: 1, “b”: 2, “d”: “hello”}

*Bài 8.4 Viết chương trình kiểm tra xem từ điển có rỗng không. Nếu có, in ra là “Từ điển rỗng”. Nếu không, in ra là “Từ điển không rỗng”.*

Ví dụ: {“a”: 1, “b”: 2, “c”: 3.5, “d”: “hello”}



Lập trình cơ bản với Python

Từ điển không rỗng

*Bài 8.5 Viết chương trình tính tổng các giá trị là số nguyên có trong từ điển.*

Ví dụ: {“a”: 1, “b”: 2, “c”: 3.5, “d”: “hello”}

Tổng là 3

*Bài 8.6 Viết chương trình gộp 2 từ điển lại làm một. Nếu khóa trùng thì giá trị của khóa này trong từ điển mới là tổng của 2 giá trị trong từ điển cũ. In ra từ điển gộp.*

Ví dụ:

Từ điển 1: {“a”: 1, “b”: 2, “c”: 3.5, “d”: “hello”}

Từ điển 2: {“e”: 1, “b”: 2, “f”: 3.5, “d”: “world”}

Từ điển sau khi gộp (không cần đúng thứ tự như ví dụ):

{“a”: 1, “b”: 4, “c”: 3.5, “d”: “helloworld”, “e”:1, “f”:3.5}

*Bài 8.6 Viết chương trình xóa tất cả các phần tử có giá trị trùng nhau trong từ điển. In ra từ điển sau khi xóa.*

Ví dụ: {“a”: 1, “b”: 2, “c”: 1, “d”: “hello”}

Từ điển sau khi xóa: {“b”: 2, “d”: “hello”}

*Bài 8.7 Viết chương trình xuất ra 3 phần tử có giá trị lớn nhất trong từ điển. Thứ tự in từ giá trị lớn đến nhỏ, nếu bằng nhau thì thứ tự nào cũng được.*

Từ điển: {“a”: 1, “b”: 8, “c”: 9, “d”: 2.5, “e”: 8}

3 phần tử có giá trị lớn nhất là:

1. 9
2. 8
3. 8

Lập trình cơ bản với Python

*Bài 8.8 Viết chương trình chuyển đổi một chuỗi do người dùng nhập thành từ điển với quy ước như sau:*

*- Mỗi kí tự trong chuỗi là khóa trong từ điển. Có phân biệt hoa thường.*

*- Giá trị của nó là số lần xuất hiện của kí tự đó trong chuỗi* Ví dụ: “ILoveYouVeryMuch”

Từ điển có được là (không cần đúng thứ tự như trong ví dụ):

{“I”:1, “L”:1, “o”:2, “v”:1, “e”:2, “Y”:1, ”u”:2, ”V”:1, ”r”:1, ”y”:1, ”M”:1, ”c”:1, ”h”:1}

*Bài 8.9 Viết chương trình dùng từ điển để quản lý một danh sách các sinh viên trong lớp với mỗi sinh viên có các thông tin:*

* *Id: mã sinh viên*
* *Họ tên*
* *Năm sinh*
* *Giới tính*
* *Địa chỉ*
* *Điểm Toán, Lý, Hóa*

*Mỗi sinh viên được quản lý với khóa là mã id của sinh viên và không trùng nhau.*

*Chương trình cho phép:*

* *Thêm một sinh viên mới*
* *Xóa một sinh viên dựa trên mã sinh viên*
* *Tìm sinh viên với họ tên*
* *Tính điểm TB cho từng sinh viên*

**Chương 7 : Danh sách**

**Mục tiêu:**

* *Nắm được khái niệm danh sách trong Python*
* *Biết cách khai báo một danh sách*
* *Các thao tác cơ bản trên dang sách*

**Nội dung chính:**

* *Khai báo danh sách*
* *Cách thức xử lý danh sách*
* *Các phương thức của danh sá**ch*

**1. Tổng quan về danh sách trong Python**

Danh sách (List) là một tập hợp các phần tử. Danh sách trong Python có thể chứa các phần tử thuộc các kiểu dữ liệu giống nhau hay khác nhau, nhưng thông thường chúng thuộc cùng một kiểu dữ liệu.

***Khai báo danh sách:***

Để tạo danh sách, có thể dùng dấu ngoặc vuông

List1 = [10, 20, 30, 40]

List2 = ['crunchy frog', 'ram bladder', 'lark vomit'] List3 = ['spam', 2.0, 5, [10, 20]]

List1 là danh sách gồm 4 số nguyên.

List2 danh sách gồm 3 chuỗi ký tự

List3 là một sanh sách gồm những kiểu dữ liệu khách nhau:

‘spam’: một chuỗi

2.0: số thực

5: số nguyên

[10,20]: một danh sách.

Một danh sách có thể chứa các phần tử là một danh sách (List3 trong ví dụ trên). Cơ chế này gọi là nested.

Danh sách chứa ***0*** phần tử gọi là danh sách rỗng. Khai báo danh sách rỗng bằng cặp dấu ngoặc vuông []

* cheeses = ['Cheddar', 'Edam', 'Gouda']
* numbers = [17, 123]
* empty = []
* print cheeses, numbers, empty ['Cheddar', 'Edam', 'Gouda'] [17, 123] []
  1. **Một số thao tác cơ bản trên danh sách:**

1. Truy xuất đến các phần tử và danh sách con trong một danh sách cho trước

Chỉ số của danh sách hoạt động theo cơ chế sau:

- Chỉ số được biểu diễn bằng một con số nguyên



Lập trình cơ bản với Python

* Nếu đọc hoặc ghi một phần tử không tồn tại, sẽ thu về kết quả là IndexError
* Nếu chỉ số là số âm, nó sẽ đếm ngược từ cuối danh sách
* squares[0] # trả về phần tử có chỉ số 0

1

* squares[-1] #trả về phần tử cuối cùng

25

* squares[-3:] # trả về list con, bắt đầu tại vị trí 3 từ cuối list, kết thúc tại cuối list

[9, 16, 25]

* squares[:] # trả về tất cả phần tử của list

[1, 4, 9, 16, 25]

* 1. Kiểm tra một phần tử có trong danh sách hay không: Dùng toán tử ***in***
* cheeses = ['Cheddar', 'Edam', 'Gouda']
* 'Edam' in cheeses

True

* 'Brie' in cheeses False
  1. Cộng (nối) hai list.

a = [1, 2, 3]

b = [4, 5, 6]

c = a + b

print c

[1, 2, 3, 4, 5, 6]

Tương tự, phép nhân ***\****, sẽ lặp lại danh sách một số lần.

* [0]\*4 [0, 0, 0, 0]
* [1,2,3]\*3

[1, 2, 3, 1, 2, 3, 1, 2, 3]

1. Phép gán trong danh sách.

Thay đổi giá trị các phần tử của danh sách

* cubes = [1, 8, 27, 65, 125]
* cubes[3] = 64 # thay giá trị
* cubes

Lập trình cơ bản với Python

[1, 8, 27, 64, 125]

Có thể thực hiện phép gán cho một list con, điều này có thể

thay đổi kích thước của list

* letters = ['a', 'b', 'c', 'd', 'e', 'f', 'g']
* letters

['a', 'b', 'c', 'd', 'e', 'f', 'g']

* # thay đổi một số giá trị
* letters[2:5] = ['C', 'D', 'E']
* letters

['a', 'b', 'C', 'D', 'E', 'f', 'g']

* # xóa chúng
* letters[2:5] = []
* letters

['a', 'b', 'f', 'g']

* # xóa list bằng cách thay tất cả phần tự bằng list rỗng
* letters[:] = []
* letters

[]

* 1. Chiều dài của list có thể được lấy bằng hàm len()
* letters = ['a', 'b', 'c', 'd']
* len(letters)

4

Lưu ý: dù list có thể chứa list, nested list chỉ tính là một phần tử. List sau có bốn phần tử:

['spam', 1, ['Brie', 'Roquefort', 'Pol le Veq'], [1, 2, 3]]

1. Duyệt list

Để duyệt qua một list, có thể dùng vòng lặp

cheeses = ['Cheddar', 'Edam', 'Gouda'] for cheese in cheeses:

print cheese

Để cập nhật nội dung trong list, dùng vòng lặp và chỉ số

numbers = [17, 123]

for i in range(len(numbers)):

numbers[i] = numbers[i] \* 2

1. Xóa danh sách

Có nhiều cách để xóa.



Lập trình cơ bản với Python

Xóa một phần tử tại vị trí cho trước:

* t = ['a', 'b', 'c']
* x = t.pop(1)
* print t

['a', 'c']

* print x

B

Xóa và trả vê một giá trị tại vị trí cho trước. Nếu hàm pop không truyền vào tham số, phần tử cuối sẽ bị xóa và trả về.

Nếu chỉ muốn xóa mà không cần lấy giá trị trả về:

* t = ['a', 'b', 'c']
* del t[1]
* print t

['a', 'c']

Nếu biết phần tử muốn xóa mà không biết vị trí:

* t = ['a', 'b', 'c']
* t.remove('b')
* print t

['a', 'c']

Xóa nhiều phần tử:

* t = ['a', 'b', 'c', 'd', 'e', 'f']
* del t[1:5]
* print t

['a', 'f']

1. **Các phương thức của List**

|  |  |  |  |
| --- | --- | --- | --- |
| Phương thức | Diễn giải |  |  |
|  |  | | |
| list.append(*x*) | Thêm một phần tử vào cuối list | | |
|  |  | | |
| list.extend(*iterable*) | Thêm các phần tử vào cuối list | | |
|  | Ví dụ: |  |  |
|  | x = [1, 2, 3] | |  |
|  | x.extend([4, 5]) | | |
|  | print (x) |  |  |
|  | [1, 2, 3, | 4, | 5] |
|  |  | | |
| list.insert(*i*, *x*) | Thêm phần tử ***x*** vào vị trí ***i*** | | |
|  |  | | |
| list.remove(*x*) | Xóa phần tử đầu tiên trong list có giá | | |
|  | trị bằng x. Trả về lỗi nếu không tìm thấy | | |
|  |  | | |
| list.pop([*i*]) | Trả về phần tử tại vị trí ***i*** trong list và | | |
|  | xóa phần tử đó ra khỏi list | | |
|  |  | | |
| list.clear() | Xóa tất cả phần tử trong list | | |
|  |  |  |  |



Lập trình cơ bản với Python

|  |  |
| --- | --- |
| list.index(*x*, *start*, | Trả về vị trí của phần tử đầu tiên trong |
| *end*) | list có giá trị bằng x, trả về lỗi nếu |
|  | không tìm thấy. |
|  | Có thể giới hạn phạm vi tìm kiếm bằng |
|  | cách xác định vị trí bắt đầu start và vị |
|  | trí kết thúc end |
|  |  |
| list.count(*x*) | Trả về số lần xuất hiện của x trong danh |
|  | sách |
|  |  |
| list.sort(*key=None*, | Sắp xếp các phần tử trong list. Mặc định |
| *reverse=False*) | là sắp tăng dần. Có thể sắp xếp giảm dần |
|  | bằng cách reverse=True |
|  |  |
| list.reverse() | Đảo ngược các phần tự trong list |
|  |  |
| list.copy() | Tạo nên bản copy của list |
|  |  |

Ví dụ: minh họa cách sử dụng một số phương thực của list

* fruits = ['orange', 'apple', 'pear', 'banana', 'kiwi', 'apple', 'banana']
* fruits.count('apple')

2

* fruits.count('tangerine')

0

* fruits.index('banana')

3

* fruits.index('banana', 4) # Tìm banana tiếp theo bắt đầu từ vị trí 4

6

* fruits.reverse()
* fruits

['banana', 'apple', 'kiwi', 'banana', 'pear', 'apple', 'orange']

* fruits.append('grape')
* fruits

['banana', 'apple', 'kiwi', 'banana', 'pear', 'apple', 'orange', 'grape']

* fruits.sort()
* fruits

['apple', 'apple', 'banana', 'banana', 'grape', 'kiwi', 'orange', 'pear']

* fruits.pop()

'pear'

Lập trình cơ bản với Python

Bài tập List

Bài 5.1

Viết chương trình tính tổng các phần tử trong một danh sách gồm các số thực

Bài 5.2

Viết chương trình tính tổng các số chẵn trong một danh sách gồm các số nguyên

Bài 5.3

Viết chương trình tìm phần tử lớn nhất trong danh sách gồm cách số thực

Bài 5.4

Viết chương trình kiểm tra xem 2 hai danh sách có phần tử trùng nhau hay không.

Bài 5.5

Viết chương trình xóa các phần tử trùng trong danh sách

Bài 5.6

Viết chương trình xóa các số chẵn, trong một danh sách các số nguyên

Bài 5.7

Viết chương trình xuất ra các phần tử chung của hai danh sách

Bài 5.8

Viết chương trình để tách một danh sách gồm các số nguyên ra thành hai danh sách: danh sách 1 chứa các số chẫn, và danh sách 2 chứa các số lẽ.

Bài 5.9

Viết chương trình tìm vị trí của một phần tử trong một danh sách cho trước. Nếu phần tử này không có trong danh sách, xuất ra -1

Ví dụ: a = [1, 3, 5, 7, 9]

Tìm 3. Xuất ra 1

Tìm 9. Xuất ra 4



Lập trình cơ bản với Python

Tìm 2. Xuất ra -1

Bài 5.10

Viết chương trình đảo ngược 1 danh sách và xuất ra màn hình. Ví dụ: a = [1, 3, 5, 7]. Output: [7, 5, 3, 1]

