Giới thiệu về dictionary và set trong python

1 dictionary

Với kiểu dữ liệu danh sách chúng ta có thể lưu trữ đối tượng không hạn chế và truy xuất phần tử thông qua chỉ số. Tuy nhiên với dữ liệu có cấu trúc như thông tin của một sinh viên gồm nhiều trường thông tin: mã sinh viên, họ tên, tuổi,… thì kiểu dữ liệu danh sách không thể lưu một cách tường minh được.

Vì thế mà python cung cấp cho chúng ta thêm kiểu dữ liệu từ điển (dictionary) để lưu trữ tập dữ liệu cấu trúc.(<khóa>:<giá trị>).

Đặc tính của từ điển:

* Được giới hạn bởi cặp ngoặc nhọn {}, tất cả những gì nằm trong đó là những phần tử của từ điển.
* Các phần tử của từ điển được phân cách nhau ra bởi dấu phẩy dưới ",".
* Mỗi phần tử của từ điển là một cặp <khóa>: <giá trị>
* Ngăn cách giữa thành phần <khóa> và thành phần <giá trị> bởi dấu hai chấm ":"
* Các <khóa> trong một từ điển buộc phải là một đối tượng băm (duy nhất)
  1. Cách tạo từ điển.

Cách 1: sử dụng câu lệnh gán để tạo từ điển

<Tên biến> = {<khóa\_1: giá trị\_1>, <khóa\_2: giá trị\_2>,...,<khóa\_n: giá trị\_n>}

Trong đó:

• <tên biến>: Là tên của một biến do người dùng đặt theo quy tắc đặt tên

• < khóa\_1>, < khóa\_2>, ... ,< khóa\_n>: Là danh mục các khóa được dùng như chỉ số của các phần tử trong từ điển và các khóa này là không trùng nhau.

• <giá\_trị\_1>, <giá\_trị\_2>, …, <giá\_trị\_n>: Là dãy các giá trị được lưu trữ trong từ điển đi theo khóa và được quản lý bởi <tên biến>.

Nếu ta không đưa cặp khóa, giá trị nào vào danh sách thì Python sẽ khởi tạo một từ điển rỗng.

Ví dụ:

#Khởi tạo từ điển lưu thông tin về sinh viên

sinhvien= {"MaSV": "20dh03057", "Ten": "Trần Xuân thành", "Tuổi": 21}

#Hiển thị từ điển sinh viên ra màn hình

print("Thông tin môn học: ",sinhvien)

Cách 2: sử dụng hàm dict() để tạo danh sách:

Cú pháp: <Tên biến> = dict()

Ví dụ:

sinhvien = dict(

'Tên': 'Lê Hữa Lộc',

'Quê quán': 'Hạ Long',

'Giới tinh': 'Nữ',

'Tình trạng hôn nhân': 'Mập mờ với nhiều em'

)

* 1. truy cập đến các phần tử trong từ điển

1.2.1 Lấy giá trị gắn với khóa của phần tử trong từ điển

Cú pháp:

hoặc

|  |  |  |
| --- | --- | --- |
| #Truy xuất tự nhiên  <Tên biến>[<khóa>] | Hoặc | #Sử dụng phương thức get()  <Tên biến>.get(<khóa>) |

Ví dụ:

#khởi tạo từ điển sinh viên

sinhvien = {

'Tên': 'Lê Hữa Lộc',

'Quê quán': 'Hạ Long',

'Tình trạng hôn nhân': 'Mập mờ với nhiều em'

}

#Hiển thị tên của sinh viên đó ra màn hình

print("Tên=",sinhvien['Tên'])

#Hiển thị tuổi của người đó ra màn hình

print("Tình trạng hôn nhân =",sinhvien.get('Tình trạng hôn nhân'))

1.2.2 Lấy tất cả các khóa trong từ điển

Cú pháp: <Tên biến>.keys() #Truy xuất đến thuộc tính keys

Ví dụ: Lấy tất cả các khóa có trong biến từ điển sinhvien trên.

sinhvien = {

"MaSV": "20dh03057",

"Ten": "Trần Xuân thành",

"Tuoi": 21

}

#Hiển thị tất cả khóa trong sinhvien ra màn hình

print("Các khóa =",sinhvien.keys())

#kết quả: Các khóa = dict\_keys(['MaSV', 'Ten', 'Tuoi'])

1.2.3 lấy tất cả các khóa trong từ điển

Cú pháp: <Tên biến>.values() #Truy xuất đến thuộc tính values

Ví dụ: Lấy tất cả các giá trị có trong biến từ điển manhinh

manhinh = {

"DoPhanGiai": "2k",

"DoLon": "27inch",

"HangSanXuat": "HkC",

"CongNghe": "HDR10"

}

print("Giá trị =",manhinh.values())

#kết quả: Giá trị = dict\_values(['2k', '27inch', 'HkC', 'HDR10'])

1.2.4 Duyệt qua tất cả các phần tử trong từ điển: Để duyệt qua tất cả các phần tử

trong từ điển, chúng ta có thể sử dụng cấu trúc for hoặc while như sau:

Ví dụ: Duyêt qua tất cả các phần tử có trong biến laptop

laptop={

"Ten": "Vivobook\_15\_j533",

"HangSanXuat": "Asus",

"CongKetNoi": "Hdmi,usb-type-C 3.1, 2x usb-typeA 3.0..."

}

# Duyệt qua tất cả các phần tử trong biến laptop

for i in laptop.keys():

print( i, ": ", laptop[i])

#kết quả:

Ten : Vivobook\_15\_j533

HangSanXuat : Asus

CongKetNoi : Hdmi,usb-type-C 3.1, 2x usb-typeA 3.0...

* 1. Thêm sửa một phần tử trong từ điển

Cú pháp: <Tên biến>.[<khóa>]= <giá trị>

Trong đó:

• <tên biến>: Là tên một biến từ điển đã được khai báo trước đó

• < khóa>: Chỉ định khóa được lưu trữ trong từ điển, nếu khóa này đã có thì giá trị cũ sẽ mất đi và giá trị mới sẽ thay thế; nếu khóa chưa có trong từ điển thì một phần tử mới <khoá>: <giá trị> sẽ được thêm vào từ điển.

• <giá trị>: Là giá trị được lưu trữ trong từ điển đi theo khóa và được quản lý bởi <tên biến>.

Ví dụ: Thêm phần tử "BanPhim": "FullSize" và Sửa "DoPhanGiai": "2k" vào biến từ điển laptop

# biến khởi tạo ban đầu

laptop={

"Ten": "Vivobook\_15\_j533",

"HangSanXuat": "Asus",

"CongKetNoi": "Hdmi,usb-type-C 3.1, 2x usb-typeA 3.0...",

"DoPhanGiai": "FullHD"

}

#lệnh thêm phần tử "BanPhim": "FullSize"

laptop["BanPhim "] = " FullSize "

# Lệnh sửa phần tử "DoPhanGiai"

laptop["DoPhanGiai"] = "2k"

# Duyệt qua tất cả các phần tử trong biến laptop

for i in laptop.keys():

print( i, ": ", laptop[i])

#kết quả:

Ten : Vivobook\_15\_j533

HangSanXuat : Asus

CongKetNoi : Hdmi,usb-type-C 3.1, 2x usb-typeA 3.0...

DoPhanGiai : 2k

BanPhim : FullSize

* 1. Xóa một phần tử trong từ điển

Cú pháp: # Xóa phần tử có khóa ra khỏi từ điển

<Tên biến>.\_\_delitem\_\_(<khóa>);

#Xoá tât cả các phần tử có trong từ điển

< Tên biến >.clear();

Ví dụ: Xóa phần tử có khóa bằng Camera trong từ điển điện thoại

dienthoai ={

"Ten": "LG\_V50\_Thinq",

"Camera": "3 x 16px",

"DungLuongPin": "4000MAH",

"DoPhanGiai": "2k"

}

# lệnh xóa phần tử

dienthoai.\_\_delitem\_\_('Camera')

# Duyệt qua tất cả các phần tử trong biến dienthoai

for i in dienthoai.keys():

print( i, ": ", dienthoai[i])

#kết quả

Ten : LG\_V50\_Thinq

DungLuongPin : 4000MAH

DoPhanGiai : 2k

1. kiểu dữ liệu set

Các đặc điển quan trọng trong set:

* Các phần tử duy nhất: Mỗi phần tử trong set chỉ xuất hiện một lần duy nhất. Nếu bạn thêm cùng một phần tử một lần nữa, nó sẽ không tạo ra sự thay đổi trong tập hợp.
* Không có thứ tự: Các phần tử trong set không được sắp xếp theo thứ tự cụ thể. Điều này có nghĩa là bạn không thể truy cập phần tử trong set bằng cách sử dụng chỉ số như trong danh sách.
* Các phép toán tập hợp: set hỗ trợ nhiều phép toán tập hợp như hợp, giao, hiệu, và phép đối xứng.
  1. cách tạo set
* cách 1: sử dụng câu lênh gán

<Tên biến> = {< giá trị\_1>, < giá trị\_2>,...,< giá trị\_n>}

Trong đó:

• <tên biến>: Là tên của một biến do người dùng đặt theo quy tắc đặt tên

• <giá\_trị\_1>, <giá\_trị\_2>, …, <giá\_trị\_n>: Là dãy các giá trị được lưu trữ trong set theo tên biến.

Ví dụ:

my\_set = {1, 2, 3,3 }# lưu ý chỉ có một số 3 trong set này

print(my\_set) # kết quả {1,2,3}

* cách 2 : sử dụng hàm set

<Tên biến> = set(< giá trị\_1>, < giá trị\_2>,...,< giá trị\_n>)

Ví dụ: my\_set = set(“Đang học”, “tự dưng”, “mất điện”)

* 1. truy cập đến các phần tử trong set

Do set không có thứ tự, ta không thể truy cập phần tử trong set bằng cách sử dụng chỉ số như trong danh sách. Tuy nhiên, ta có thể sử dụng vòng lặp để duyệt qua tất cả các phần tử trong set.

Ví dụ:

my\_set = {1, 2, 3}

for item in my\_set:

print(item)

#Kết quả:

1

2

3

2.3. Thêm, sửa và xóa phần tử trong set:

Để thêm phần tử vào set ta có thể sử dụng phương thức add()

Để xóa phần tử vào set ta có thể sử dụng phương thức remove()

Để sửa phần tử trong set ta chỉ có thể xóa phần tử cũ và thêm phần tử mới

Ví dụ

my\_set = {“python”, “c\#”, “java”}

my\_set.remove(“java”) # lệnh xóa phần tử java khỏi my\_set

my\_set.add(“javascript”) # lệnh thêm phần tử dart vào my\_set

print(my\_set) # kết quả: {'c#', 'javascript', 'python'}

Khi bạn sử dụng phương thức add() để thêm phần tử vào set, ta không cần quan tâm đến vị trí cụ thể. Mọi phần tử trong set đều được coi là duy nhất và không có sự phân biệt về vị trí. Vì vậy khi sử dụng phương thức add() vị trí củ các phần tử sẽ ở vị trí ngẫu nhiên sau mỗi lần chạy.

Ngoài việc sử dụng phương thức add() thì python còn hỗ trợ phương thức update() hoặc ký tự “|” (phép hợp ) để thêm nhiều phần tử vào set.

Ví vụ:

my\_set = {"python","c#","java"}

my\_set.update({"javscript","html","css"})

print(my\_set) #kết quả: {'css', 'c#', 'html', 'javscript', 'python', 'java'}

new\_set = {"ruby","dart","php"} # khai báo một set mới

my\_set |= new\_set # sử dụng phép toán hợp

print(my\_set)

#kết quả: {'php', 'css', 'java', 'ruby', 'html', 'dart', 'python', 'c#', 'javscript'}

* ngoài phép toán hợp python còn hỗ trợ một vài phép toán tập hợp khác cho set như phép giao, hiệu và phép đối xứng

ví dụ:

set\_1 = {"python","c#","java"}

set\_2 = {"dart","javascript","java"}

phep\_giao = set\_1 & set\_2

print(phep\_giao) #kết quả: {'java'}

phep\_hieu = set\_1 - set\_2

print(phep\_hieu) #kết quả: {'c#', 'python'}

phep\_doi\_xung = set\_1 ^ set\_2

print(phep\_doi\_xung) #kết quả: {'c#', 'python', 'dart', 'javascript'}