Dictionary

Mục tiêu

- Trình bày được khái niệm Dictionary
- Trình bày được khái niệm Iterator

Dictionary trong Python

PANTECH
The foundation of tomorrow

Giới thiệu

Dictionary (bộ từ điển) là một kiểu dữ liệu, tập hợp các giá trị dữ liệu có thứ tự, là một danh sách lưu trữ dữ liệu các phần tử (element), mà mỗi phần tử là một cặp từ khóa và giá trị (key & value)

Nội dung bài đọc sẽ có các phần chính sau:

- Cấu trúc, cách tạo Dictionary
- Key & value trong từng phần tử của Dictionary
- Các phương thức sử dụng trong Dictionary

Dictionary trong Python

PANTECH
The foundation of tomorrow

Tạo Dictionary

- Trong Python, một Dictionary có thể được tạo ra bằng cách đặt một chuỗi các phần tử trong dấu ngoặc nhọn {}, và được phân tách bằng dấu phẩy ‘ , ’.
- Mỗi phần tử là một cặp khóa và giá trị (key & value) ngăn cách bởi dấu hai chấm ‘ : ‘.

Dictionary trong Python

Ví dụ:

```
#Tạo Dictionary với key là số nguyên
Dict = {1: 'Geeks', 2: 'For', 3: 'Geeks'}
print("\nDictionary with the use of Integer Keys: ")
print(Dict)

# Tạo Dictionary với nhiều kiểu key
Dict = {'Name': 'Geeks', 1: [1, 2, 3, 4]}
print("\nDictionary with the use of Mixed Keys: ")
print(Dict)
```

Dictionary trong Python

Kết quả:

Dictionary with the use of Integer Keys:

{1: 'Geeks', 2: 'For', 3: 'Geeks'}

Dictionary with the use of Mixed Keys:

{'Name': 'Geeks', 1: [1, 2, 3, 4]}

Dictionary trong Python

- Dictionary cũng có thể tạo từ phương thức khởi tạo (constructor) của lớp dict.
- Một Dictionary trống có thể tạo từ bằng cách đặt dấu ngoặc nhọn {}

Ví dụ :

```
# Tạo Dictionary trống
Dict = {}
print("Empty Dictionary: ")
print(Dict)

# Tạo Dictionary với dict
Dict = dict({1: 'Geeks', 2: 'For', 3:'Geeks'})
print("\nDictionary with the use of dict(): ")
print(Dict)
```

Kết quả :

```
Empty Dictionary:
{}
Dictionary with the use of dict():
{1: 'Geeks', 2: 'For', 3: 'Geeks'}
```

Dictionary trong Python

PANTECH
The foundation of tomorrow

Khóa và giá trị (key & value) trong Dictionary

- Các phần tử (element) trong Dictionary được sắp xếp theo thứ tự, có thể thay đổi, nhưng không cho phép trùng lặp
- Các phần tử cấu trúc từ cặp key & value, và có thể tham chiếu bằng cách sử dụng key
- Các giá trị (value) trong Dictionary có thể thuộc bất kì kiểu dữ liệu nào (string, số ...) và có thể trùng lặp
- Còn các khóa (key) không thể trùng lặp và phải bất biến (string, tuple ...).

Dictionary trong Python

Phương thức	Mô tả
copy()	Phương thức trả về một bản sao của Dictionary
clear()	Phương thức xóa các phần tử của Dictionary
pop()	Xóa phần tử với từ khóa chỉ định
fromkeys()	Trả về một Dictionary với key và value chỉ định
get()	Trả về value với key chỉ định
items()	Trả về một danh sách là một tuple chứa key, value tương ứng
keys()	Trả về một danh sách chứa các key của Dictionary
popitem()	Xóa phần tử cuối cùng của Dictionary
setdefault()	Trả về value với key chỉ định, nếu không có key thì thì sẽ chèn thêm key và value chỉ định. Giá trị mặc định của value là None
update()	Cập nhật thêm key và value chỉ định
values()	Trả về một danh sách chứa value của Dictionary

Dictionary trong Python

Ngoài ra khi xóa phần tử có thể dùng keyword **del**

Ví dụ:

```
this_dict = {
  "brand": "Ford",
  "model": "Mustang",
  "year": 1964
}
del this_dict["model"]
print(this_dict)
```

Dictionary trong Python

Lưu ý: Keyword del có thể xóa hoàn toàn Dictionary

Ví dụ:

```
this_dict = {
  "brand": "Ford",
  "model": "Mustang",
  "year": 1964
}
del this_dict
print(this_dict)
```

Kết quả: Sẽ xuất hiện lỗi vì this_dict không còn tồn tại

Dictionary trong Python

Dictionary lồng nhau (Nested Dictionary)

Là đặt một dictionary bên trong một dictionary khác. Đó là tập hợp các dictionary thành một dictionary duy nhất

Cú pháp:

```
nested_dict = {'dictA': {'key_1': 'value_1'},
               'dictB': {'key_2': 'value_2'}}
```

Ở đây, nested_dict là một dictionary lồng nhau với dictionary dictA và dictB. Chúng là hai dictionary, mỗi dictionary có key và value riêng.

Truy cập vào phần tử của Dictionary lồng nhau, ta sử dụng index [] trong Python

Dictionary trong Python

Ví dụ:

```
people = {1: {'name': 'John', 'age': '27', 'sex': 'Male'},
          2: {'name': 'Marie', 'age': '22', 'sex': 'Female'}}

print(people[1]['name'])
print(people[1]['age'])
print(people[1]['sex'])
```

Kết quả:

John
27
Male

Dictionary trong Python

Thêm phần tử vào Dictionary lồng nhau

Ví dụ :

```
people = {1: {'name': 'John', 'age': '27', 'sex': 'Male'},
          2: {'name': 'Marie', 'age': '22', 'sex': 'Female'}}
people[3] = {}

people[3]['name'] = 'Luna'
people[3]['age'] = '24'
people[3]['sex'] = 'Female'
people[3]['married'] = 'No'
print(people[3])
```

Kết quả:

{'name': 'Luna', 'age': '24', 'sex': 'Female', 'married': 'No'}

Dictionary trong Python

Xóa phần tử trong Dictionary lồng nhau

Ví dụ:

```
people = {1: {'name': 'John', 'age': '27', 'sex': 'Male'},
          2: {'name': 'Marie', 'age': '22', 'sex': 'Female'},
          3: {'name': 'Luna', 'age': '24', 'sex': 'Female'},
       4: {'name': 'Peter', 'age': '29', 'sex': 'Male'}}


del people[3], people[4]
print(people)
```

Kết quả:

```
{1: {'name': 'John', 'age': '27', 'sex': 'Male'},
 2: {'name': 'Marie', 'age': '22', 'sex': 'Female'}}
```

Dictionary trong Python

Lưu ý :

- Dictionary lồng nhau là danh sách dictionary không có thứ tự
- Có thể thu gọn, hoặc mở rộng Dictionary khi cần
- Dictionary lồng nhau cũng có cặp key & value, và truy cập bằng cách sử dụng key

Dictionary trong Python

PANTECH
The foundation of tomorrow

Lặp trong Dictionary

Có nhiều cách để lặp qua 1 Dictionary trong Python

- Lặp qua các khóa (key)
- Lặp qua các giá trị (value)
- Lặp qua các cặp khóa và giá trị

Dictionary trong Python

Ví dụ: Lặp qua các khóa (key)

```
fruitscolor = {"Banana" : "Yellow",
"Mango" : "Green",
"Apple" : "Red",
"Grapefruit" : "Pink",
"Blackberry" : "Purple",
"Sapodilla" : "Brown"}

# For loop to Iterating over key
for key in fruitscolor:
 print (key)
```

Kết quả:

```
Banana
Mango
Apple
Grapefruit
Blackberry
Sapodilla
```

Dictionary trong Python

Ví dụ: Lặp qua các giá trị (value)

```
fruitscolor = {"Banana" : "Yellow",
"Mango" : "Green",
"Apple" : "Red",
"Grapefruit" : "Pink",
"Blackberry" : "Purple",
"Sapodilla" : "Brown"}

# Using for loop Iterating over item and
get value of item through value() function
for color in fruitscolor.values():
    print(color)
```

Kết quả :

```
Yellow
Green
Red
Pink
Purple
Brown
```

Dictionary trong Python

Ví dụ: Lặp qua các cặp khóa và giá trị

```
fruitscolor = {"Banana" : "Yellow",
               "Mango" : "Green",
               "Apple" : "Red",
               "Grapefruit" : "Pink",
               "Blackberry" : "Purple",
               "Sapodilla" : "Brown"}

# For loop to Iterating over key & value

for fruit, color in fruitscolor.items():
 print(fruit, ":", color)
```

Kết quả :

Banana : Yellow
Mango : Green
Apple : Red
Grapefruit : Pink
Blackberry : Purple
Sapodilla : Brown

Tổng kết

Qua bài viết này chúng ta đã tìm hiểu:

- Qua bài viết chúng ta đã tìm hiểu :
- Khái niệm Dictionary, cặp key & value của từng phần tử trong Dictionary
- Cách tạo Dictionary, các phương thức sử dụng trong Dictionary
- Dictionary lồng nhau
- Lặp trong Dictionary

Iterator

Giới thiệu

Iterator là một đối tượng được sử dụng để lặp qua các đối tượng chứa nhiều giá trị như **list, tuple, dict và set**

Nội dung bài đọc sẽ có các phần chính sau:

- Iterator và Iterable
- Phương thức Iterator

Iterator

Iterable và Iterator

Iterable là một đối tượng bất kì trong Python, có phương thức __iter__ trả về một iterator, hoặc một phương thức xác định __getitem__ có thể nhận các phần tử có vị trí bắt đầu từ 0.

Vì vậy iterable là một object mà bạn có thể lấy ra một iterator.

Itearable bao gồm **list, tuple, dict và set**, cho phép truy cập qua các phần tử bằng chỉ mục hay còn gọi là index.

Còn **Iterator** là đối tượng với một công việc duy nhất là trả về phần tử tiếp theo trong các đối tượng iterable có phương thức __next__

Iterable cho phép nó được lặp lại trong vòng lặp for, và có độ dài vô hạn

Iterator

Ví dụ:

```
mytuple = ("apple", "banana", "cherry")
for x in mytuple:
    print(x)
```

Kết quả:

```
apple
banana
cherry
```

Iterator

Khi chúng ta đã biết iterable và iterator là gì và cách sử dụng chúng, thì chúng ta có thể xác định một hàm lặp qua một đối tượng iterable mà không cần sử dụng vòng lặp for

- Tạo một danh sách có thể lặp qua
- Dùng phương thức **iter** để lấy ra iterator
- Vòng lặp sẽ bắt đầu khi ta dùng phương thức **next**
- Vòng lặp sẽ lặp qua tất cả các iterator, và sẽ dừng lại khi không còn phần tử nào để lặp qua, sẽ trả về **StopIteration**

Iterator

Ví dụ:

```
my_list = [4, 7, 0, 3]
my_iter = iter(my_list)
print(my_iter.__next__())
print(my_iter.__next__())
print(my_iter.__next__())
print(my_iter.__next__())
next(my_iter)
```

Kết quả:

```
4
7
0
3
StopIteration   Traceback(most recent
call last)
<ipython-input-3-42971c095224> in
<module>
      5 print(my_iter.__next__())
      6 print(my_iter.__next__())
----> 7 next(my_iter)
StopIteration:
```

Iterator

PANTECH
The foundation of tomorrow

Các phương thức của đối tượng Iterator

Trong Python, Iterator là một đối tượng thực hiện giao thức vòng lặp, sử dụng các phương thức **__iter__()** và **__next__()**

Trong đó

- Phương thức **__iter__()** dùng để khởi tạo , và luôn phải trả về chính đối tượng của iterator
- Phương thức **__next__()** dùng để thực hiện thao tác lặp lại, và phải trả về phần tử kế tiếp trong danh sách

Iterator

Lưu ý: Đối tượng Iterator

- Có thể được truyền cho phương thức next() để trả về phần tử tiếp theo trong nhóm hoặc ngoại lệ StopIteration nếu không còn phần tử nào
- Trả về chính nó khi được truyền cho phương thức iter()

Iterator

Ví dụ:

```
class MyNumbers:
    def __iter__(self):
        self.a = 1
        return self

    def __next__(self):
        x = self.a
        self.a += 1
        return x
myclass = MyNumbers()
myiter = iter(myclass)

print(next(myiter))
print(next(myiter))
print(next(myiter))
print(next(myiter))
```

Kết quả:

1
2
3
4

Iterator

PANTECH
The foundation of tomorrow

StopIteration

Để ngăn việc lặp đi lặp lại của vòng lặp, chúng ta thêm câu lệnh **StopIteration** để thoát khỏi chương trình lặp

Iterator

Ví dụ:

```
class MyNumbers:
    def __iter__(self):
        self.a = 2
        return self

    def __next__(self):
        x = self.a
        self.a += 2
        if (self.a < 11):
            return x
        else:
            StopIteration

mynumber = MyNumbers()
inumber = iter(mynumber)

print(next(inumber))
print(next(inumber))
print(next(inumber))
print(next(inumber))
print(next(inumber))
```

Kết quả: Lệnh print cuối cùng nó trả về None vì kết quả đã vượt khỏi số 10.

```
2
4
6
8
None
```

Tổng kết

Qua bài viết này chúng ta đã tìm hiểu:

- Khái niệm Iterator, Iterable
- Phương thức tạo Iterator, câu lệnh StopIterator

[Thực hành] Tìm dữ liệu bằng Id

- **Mục tiêu**

  Luyện tập:

  - Cách thao tác với đối tượng dictionary ,
  - Viết và sử dụng các hàm trong lập trình,
  - Sử dụng hàm if...elif...else
  - Sử dụng vòng lặp for...in

- **Mô tả**

  Viết chương trình cho phép người dùng tìm dữ liệu của nhân viên bằng cách nhập id của nhân viên đó (Database sẽ được cho sẵn).

[Thực hành] Tìm sản phẩm

- **Mục tiêu**

  Luyện tập:

  - Cách thao tác với đối tượng dictionary,
  - Viết và sử dụng các hàm trong lập trình,
  - Sử dụng hàm if...elif...else
  - Sử dụng vòng lặp for...in

- **Mô tả bài toán**

  Viết chương trình hiển thị các sản phẩm có giá bé hơn hoặc bằng so với giá tiền mà người dùng nhập (database đã được cho sẵn).

[Bài tập] Ứng dụng từ điển đơn giản

- **Mục tiêu**

  Luyện tập:

  - Cách thao tác với đối tượng Dictionary,
  - Viết và sử dụng các hàm trong lập trình,
  - Sử dụng hàm if...elif...else
  - Sử dụng vòng lặp for...in

- **Mô tả**

  Viết chương trình cho phép người dùng tra cứu các từ tiếng Anh sang tiếng Việt bằng cách nhập từ cần tra cứu. Danh sách các từ được lưu trữ sẵn trong một Dictionary.

[Bài tập] Đếm số lần xuất hiện của một từ

- **Mục tiêu**

  Luyện tập:

  - Cách thao tác với chuỗi.
  - Viết và sử dụng các hàm trong lập trình,
  - Sử dụng vòng lặp for...in

- **Mô tả**

  Viết chương trình đếm số lần xuất hiện của một từ trong một văn bản nhất định (đã cho sẵn)

[Bài tập] Ứng dụng quản lý sản phẩm

- **Mục tiêu**

  Luyện tập:
  - Cách thao tác với đối tượng Dictionary,
  - Viết và sử dụng các hàm trong lập trình,
  - Sử dụng hàm if...elif...else
  - Sử dụng vòng lặp while
- **Mô tả**

  Viết chương trình cho phép người dùng quản lý sản phẩm, gồm các chức năng sau:
  - Hiển thị danh sách sản phẩm
  - Thêm sản phẩm mới vào danh sách
  - Sửa tên của sản phẩm trong danh sách
  - Xoá một sản phẩm khỏi danh sách