10. Kiểu dữ liệu Dictionary

Chúng ta đã làm quen với các kiểu dữ liệu tuần tự trong chương trước. Trong chương này của khóa học chúng tôi sẽ trình bày một kiểu dữ liệu quan trọng khác, đó là kiểu dictionary (hay còn gọi kiểu từ điển) và các phương thức cũng như toán tử trong dictionary. Các chương trình hoặc tập lệnh Python không có “list” và “dictionary” thì gần như không thể tưởng tượng được. “List” và “dictionary” có thể dễ dàng thay đổi, chẳng hạn có thể bị thu hẹp và mở rộng trong thời gian chạy. Quá trình co giãn đó được thực hiện mà không cần phải sao chép từ đó nó đạt được một lợi thế vô cùng lớn về khía cạnh hiệu năng. Một dữ liệu kiểu “dictionary” hoàn toàn có thể là một phần tử trong một dữ liệu kiểu “list” khác và ngược lại. Vậy sự khác biệt giữa danh sách và từ điển là gì? List là tập hợp các đối tượng tuần tự được, trong khi dictionary là các tập dữ liệu không có thứ tự. Nhưng sự khác biệt chính là các mục trong từ điển được truy cập qua “key” mà không phải thông qua vị trí của chúng (ta gọi đó là index). Một “dictionary” là một mảng kết hợp (còn gọi là băm). Mỗi “key” của “dictionary” được liên kết (hoặc ánh xạ) đến một giá trị. Các giá trị của một “dictionary” có thể là bất kỳ kiểu dữ liệu trong Python. Vì vậy, các “dictionary” là các cặp “key – value” không có thứ tự.

“dictionary” không hỗ trợ hoạt động tuần tự như string, tuple, list. “dictionary” thuộc loại được xây dựng với cơ chế mapping!

Vào cuối chương này, chúng tôi sẽ chỉ ra cách chuyển một “dictionary” thành một danh sách chứa các (key, value)-tuples hoặc tách thành hai list (một list với các Key và một list với values). Sự chuyển đổi này cũng có thể được thực hiện ngược lại.

Khai báo một dictionary: mỗi phần tử luôn là một cặp key-value. ‘i’ là một key, và value sẽ là “iPhone” trong ví dụ phía dưới.

Để tiện lợi cho hướng dẫn trong phần này, tôi sẽ trình bầy các đoạn mã ngay trên CMD, không qua IDE.

```
>>> phones = {'i':"iPhone",'s':"Samsung",'n':"Nokia"}

>>> phones

{'i': 'iPhone', 's': 'Samsung', 'n': 'Nokia'}
```

Các bạn có thể thấy ứng với mỗi key ta luôn có một value, vậy cách đơn giản đọc và thay đổi value của một key sẽ như ví dụ dưới đây.

```
>>> phones['i']

'iPhone'

>>> phones['n'] = "NOKIA"

>>> phones

{'i': 'iPhone', 's': 'Samsung', 'n': 'NOKIA'}

>>>
```

Key phải là một đối tượng không đổi (immutable object). Thử hai ví dụ sau, dùng tuple làm key sẽ cho kết quả tốt, nhưng nếu chúng ta dùng đối tượng kiểu “list” làm key sẽ có phát sinh lỗi.

```
>>> phones = {['I','L']:"LG"}

Traceback (most recent call last):
```

```
  File "<stdin>", line 1, in <module>
TypeError: unhashable type: 'list'
>>> phones = {('I','L'):"LG"}
>>> phones
{('I', 'L'): 'LG'}
>>>
```

Nhiều key có thể chứa cùng một value, nhưng các key là duy nhất (1 key không thể đại diện cho 2 giá trị khác nhau.

```
>>> phones = {'i':"iphone", 'i':"Iph", 's':"Samsung", "ss":"Samsung"}
>>> phones
{'i': 'Iph', 'ss': 'Samsung', 's': 'Samsung'}
>>>
```

Truy cập một key không tồn tại trong dictionary

```
>>> phones = {'i': 'iPhone', 's': 'Samsung', 'n': 'Nokia'}
>>> phones['k']
Traceback (most recent call last):
  File "<stdin>", line 1, in <module>
KeyError: 'k'
>>>
```

Các toán tử trong dictionary:

Toán tử	Giải thích
len(d)	Trả về số lượng cặp giá trị key-value trong dictionary
del d[k]	Xóa key -value có key là k trong dictionary
k in d	True, nếu k tồn tại trong các keys của dictionary
k not in d	True, nếu key k không có trong dictionary

Cách ghép hai dictionaries:

Sử dụng hàm update().

```
```

```
>>> phones = {"Hoa":1234566,"Kieu":852385238}
>>> phones.update({"Kien":98468439868,"Long":8985375})
>>> phones
{'Long': 8985375, 'Kien': 98468439868L, 'Kieu': 852385238, 'Hoa': 1234566}
>>>
```

Cách duyệt qua toàn bộ dictionary

```
>>> for p in phones:
...     print p, phones[p]
...
Long 8985375
Kien 98468439868
Kieu 852385238
Hoa 1234566
>>>
```

Chuyển dictionary thành list các keys, các values, hoặc cặp key-value thông qua các phương thức dựng sẵn trong Python của dictionary là keys(), values(), items() tương ứng.

```
>>> phones.keys()
['Long', 'Kien', 'Kieu', 'Hoa']
>>> phones.values()
[8985375, 98468439868L, 852385238, 1234566]
>>> phones.items()
[('Long', 8985375), ('Kien', 98468439868L), ('Kieu', 852385238), ('Hoa', 1234566)]
>>>
```

Tạo dictionary từ các lists. Đầu tiên dùng hàm zip() để tạo một list các tuples key-value từ hai lists của keys và values tương ứng. Sau đó ta sẽ tạo một dictionary từ hàm dict().

```
>>> keys = ['Long', 'Kien', 'Kieu', 'Hoa']
>>> numbers = [8985375, 98468439868L, 852385238, 1234566]
>>> listOfKeyValue = zip(keys,numbers)
>>> listOfKeyValue
[('Long', 8985375), ('Kien', 98468439868L), ('Kieu', 852385238), ('Hoa', 1234566)]
>>> dict(listOfKeyValue)
{'Kieu': 852385238, 'Kien': 98468439868L, 'Long': 8985375, 'Hoa': 1234566}
>>>
```

Kết Luận

Tóm lại, Một dictionary là một tập hợp các cặp “key:value” không theo thứ tự. Các giá trị “key” là các đối tượng không thể thay đổi, do đó ta có thể dùng tuple để khai báo “Key” mà không thể dùng “list”. Tùy vào mục đích sử dụng mà chúng ta có thể chuyển đổi từ dictionary sang một danh sách tuple chứa các cặp “key-value” hoặc list các key và list các value tương ứng hoặc chuyển đổi ngược lại thông qua các phương tức hỗ trợ sẵn trong Python.

Như vậy là cho đến bài học này chúng ta đã lần lượt tìm hiểu được các kiểu dữ liệu cơ bản mà Python hỗ trợ như kiểu số, string, list, tuple, set và dictionary. Vậy căn cứ vào đâu chúng ta có thể đánh giá xem nên sử dụng kiểu dữ liệu nào để thu được kết quả tối ưu nhất. Một trong những tiêu chí đánh giá phải kể đến là độ phức tạp về thời gian của các thao tác trong các kiểu dữ liệu, sẽ được chúng tôi trình bày trong bài học sắp tới.