Chương 5. Từ điển (Dictionaries)

NGUYỄN HOÀNG ANH

CNTT1. PTIT

Thông tin môn học

Ví dụ

Làm việc với từ điển

Lặp qua toàn bộ từ điển

Nesting

Kết chương

5.1. Ví dụ đơn giản về từ điển

Giả sử một trò chơi có các quái vật (aliens) có mầu sắc và có các điểm số khác nhau. Một từ điển đơn giản lưu thông tin về một quái vật cụ thể

```
alien_0={'color':'green','point':5}
print(alien_0['color'])
print(alien_0['point'])
```

Từ điển alien_0 lưu giá trị mầu sắc và điểm của quái vật

```
green
5
```

5.2. Làm việc với từ điển

một từ điển được đặt trong dấu ngoặc nhọn, {} , với một loạt các cặp khóa-giá trị bên trong dấu ngoặc nhọn:

```
alien_0 = {'color': 'green', 'points': 5}
```

Cặp khóa-giá trị là một tập hợp các giá trị được liên kết với nhau. Khi chúng ta cung cấp một khóa, Python sẽ trả về giá trị được liên kết với khóa đó.

Mọi khóa được kết nối với giá trị của nó bằng dấu hai chấm (:) và các cặp khóa-giá trị riêng lẻ được tách riêng bằng dấu phẩy.

Chúng ta có thể lưu trữ nhiều cặp khóa-giá trị tùy thích trong từ điển.

Từ điển đơn giản nhất có chính xác một cặp khoá-giá trị:

```
alien_0 = {'color': 'green'}
```

Từ điển này lưu trữ một phần thông tin về alien_0, cụ thể là màu của alien. Chuỗi 'color' là một khóa và các liên kết với giá trị là ‘green'

Truy cập các giá trị trong từ điển

Để lấy giá trị được liên kết với một khóa, cần đặt tên từ điển và sau đó đặt khóa bên trong một tập hợp các dấu ngoặc vuông

```
alien_0 = {'color': 'green'}
print(alien_0['color'])
```

```
Green
```

có thể có số lượng cặp khóa-giá trị không giới hạn trong một từ điển. Ví dụ:

```
alien_0 = {'color': 'green', 'points': 5}
```

```
alien_0 = {'color': 'green', 'points': 5}
new_points = alien_0['points']
print(f"You just earned {new_points} points!")
```

```
You just earned 5 points!
```

Thêm các cặp khoá-giá trị mới

Để thêm một cặp giá trị-khóa mới, ta sẽ cung cấp tên của từ điển, sau đó là khóa mới trong ô vuông dấu ngoặc cùng với giá trị mới.

Thêm hai thông tin vào từ điển của alien_0: hai tọa độ x và y của quái vật, giúp hiển thị quái vật tại một vị trí nhất định trên màn hình

```
alien_0['x_position'] = 0
alien_0['y_position'] = 25
print(alien_0)
```

```
{'color': 'green', 'points': 5}
{'color': 'green', 'points': 5, 'y_position': 25, 'x_position': 0}
```

Phiên bản sau của từ điển có bốn cặp khóa-giá trị. Hai cặp đầu chỉ mầu sắc và điểm của quái vật, hai cặp sau chỉ vị trí của con quái vật

Bắt đầu với từ điển rỗng

Để tạo từ điển rỗng, bắt đầu một từ điển với cặp ngoặc nhọn ({}) rỗng và sau đó thêm cặp giá trị-khóa vào từ điển trong một dòng mã riêng

```
alien_0 = {}
alien_0['color'] = 'green'
alien_0['points'] = 5
print(alien_0)
```

```
{'color': 'green', 'points': 5}
```

Sửa giá trị trong từ điển

Để sửa giá trị trong từ điển, thiết lập tên từ điển với giá trị của khóa trong ngoặc vuông, sau đó tới giá trị của khóa mà ta muốn gán cho khóa đó.

```
alien_0 = {'color': 'green'}
print(f"The alien is {alien_0['color']}.")
alien_0['color'] = 'yellow'
print(f"The alien is now {alien_0['color']}.")
```

```
The alien is green.
The alien is now yellow.
```

Sửa giá trị trong từ điển

ví dụ thú vị hơn, hãy theo dõi vị trí của con quái vật có thể di chuyển với các tốc độ khác nhau.

```
alien_0 = {'x_position': 0, 'y_position': 25, 'speed': 'medium'}
print(f"Original position: {alien_0['x_position']}")
# Move the alien to the right.
if alien_0['speed'] == 'slow':
    x_increment = 1
elif alien_0['speed'] == 'medium':
    x_increment = 2
else:
    x_increment = 3
# The new position is the old position plus the increment.
alien_0['x_position'] = alien_0['x_position'] + x_increment
print(f"New position: {alien_0['x_position']}")
```

```
Original x-position: 0
New x-position: 2
```

Xóa các cặp khóa-giá trị

Khi không còn cần một phần thông tin được lưu trữ trong một từ điển nữa, có thể sử dụng câu lệnh del để xóa hoàn toàn một cặp khóa-giá trị.

```
alien_0 = {'color': 'green', 'points': 5}
print(alien_0)
del alien_0['points']
print(alien_0)
```

```
{'color': 'green', 'points': 5}
{'color': 'green'}
```

Từ điển của các đối tượng tương tự

Cũng có thể sử dụng từ điển để lưu trữ một loại thông tin về nhiều đối tượng.

Ví dụ, giả sử ta muốn thăm dò ý kiến một số người và hỏi họ ngôn ngữ lập trình yêu thích của họ là gì

```
favorite_languages = {
        'jen': 'python',
        'sarah': 'c',
        'edward': 'ruby',
        'phil': 'python',
    }
```

```
language = favorite_languages['sarah'].title()
print(f"Sarah's favorite language is {language}.")
```

```
Sarah's favorite language is C.
```

Sử dụng get() để truy cập các giá trị

Việc sử dụng các khóa trong dấu ngoặc vuông để truy xuất giá trị mà chúng ta quan tâm từ từ điển có thể gây ra một vấn đề tiềm ẩn: nếu khóa ta yêu cầu không tồn tại, chương trình sẽ gặp lỗi.

```
alien_0 = {'color': 'green', 'speed': 'slow'}
print(alien_0['points'])
```

```
Traceback (most recent call last):
          File "alien_no_points.py", line 2, in <module>
          print(alien_0[‘points’])
     KeyError: 'points'
```

Có thể sử dụng phương thức get() để đặt giá trị mặc định sẽ được trả về nếu khóa được yêu cầu không tồn tại

```
alien_0 = {'color': 'green', 'speed': 'slow’}
point_value = alien_0.get('points', 'No point value assigned. ')
print(point_value)
```

```
No point value assigned
```

5.3. Lặp qua toàn bộ từ điển

Một từ điển Python có thể chỉ chứa một vài cặp khoá-giá trị hoặc hàng triệu cặp. Vì từ điển có thể chứa một lượng lớn dữ liệu, Python cho phép chúng ta lặp qua từ điển.

Từ điển có thể được sử dụng để lưu trữ thông tin theo nhiều cách khác nhau; do đó, tồn tại một số cách khác nhau để lặp lại chúng.

Ta có thể lặp lại tất cả các cặp giá trị khóa của từ điển, thông qua các khóa của từ điển hoặc thông qua các giá trị của nó.

Lặp qua tất cả các cặp khóa-giá trị

xem xét một từ điển mới được thiết kế để lưu trữ thông tin về người dùng trên một trang web. Từ điển sau sẽ lưu trữ username, first name, and last name

```
user_0 = {
        'username': 'efermi',
        'first': 'enrico',
        'last': 'fermi',
    }

① for key, value in user_0.items():
    ② print(f"\nKey: {key}")
    ③ print(f"Value: {value}")
```

```
Key: last
Value: fermi

Key: first
Value: enrico

Key: username
Value: efermi
```

Lặp qua tất cả các cặp khóa-giá trị(t)

Nếu ta lặp qua từ điển favorite_languages, ta sẽ nhận được tên của từng người trong từ điển và ngôn ngữ lập trình yêu thích của họ.

```
① for name, language in favorite_languages.items():
② print(f"{name.title()}'s favorite language is {language.title()}.")
```

```
Jen's favorite language is Python.
Sarah's favorite language is C.
Edward's favorite language is Ruby.
Phil's favorite language is Python.
```

```
favorite_languages = {
        'jen': 'python',
        'sarah': 'c',
        'edward': 'ruby',
        'phil': 'python',
}
```

Lặp qua tất cả các khoá trong từ điển

Phương thức key() hữu ích khi ta không cần phải làm việc với tất cả các giá trị trong từ điển. Hãy xem qua từ điển favourite_languages và in tên của những người đã tham gia cuộc thăm dò ý kiến

```
favorite_languages = {
        'jen': 'python',
        'sarah': 'c',
        'edward': 'ruby',
        'phil': 'python',
}
```

```
for name in favorite_languages.keys():
        print(name.title())
```

```
Jen
Sarah
Edward
Phil
```

Lặp qua các khóa thực sự là hành vi mặc định khi lặp qua từ điển, vì vậy hai đoạn mã này sẽ có cùng đầu ra

```
for name in favorite_languages:
for name in favorite_languages.keys():
```

Lặp qua tất cả các khoá trong từ điển

Có thể truy cập giá trị được liên kết với bất kỳ khóa nào ta quan tâm bên trong vòng lặp bằng cách sử dụng khóa hiện tại.

```
friends = ['phil', 'sarah']
for name in favorite_languages.keys():
    print(name.title())

②   if name in friends:
③       language = favorite_languages[name].title()
        print(f"\t{name.title()}, I see you love {language}!")
```

```
Hi Jen.
Hi Sarah.
    Sarah, I see you love C!
Hi Edward.
Hi Phil.
    Phil, I see you love Python!
```

Lặp qua tất cả các khoá trong từ điển

Có thể sử dụng phương thức key() để tìm hiểu xem một người cụ thể có được thăm dò ý kiến hay không

```
favorite_languages = {
        'jen': 'python',
        'sarah': 'c',
        'edward': 'ruby',
        'phil': 'python',
}
if 'erin' not in favorite_languages.keys():
    print("Erin, please take our poll!")
```

```
Erin, please take our poll!
```

Lặp các khoá của từ điển theo một thứ tự cụ thể

sắp xếp các khóa khi chúng được trả về trong vòng lặp for. Ta có thể sử dụng hàm sorted() để lấy bản sao của các khóa theo thứ tự

```
favorite_languages = {
        'jen': 'python',
        'sarah': 'c',
        'edward': 'ruby',
        'phil': 'python',
}
for name in sorted(favorite_languages.keys()):
    print(f"{name.title()}, thank you for taking the poll.")
```

```
Edward, thank you for taking the poll.
Jen, thank you for taking the poll.
Phil, thank you for taking the poll.
Sarah, thank you for taking the poll.
```

Lặp qua tất cả các giá trị trong từ điển

Giả sử chúng ta chỉ muốn có một danh sách tất cả các ngôn ngữ được chọn trong cuộc thăm dò ngôn ngữ lập trình mà không có tên của người đã chọn từng ngôn ngữ

```
favorite_languages = {
        'jen': 'python',
        'sarah': 'c',
        'edward': 'ruby',
        'phil': 'python',
}
print("The following languages have been mentioned:”)
for language in favorite_languages.values():
    print(language.title())
```

```
The following languages have been mentioned:
Python
C
Python
Ruby
```

Lặp qua tất cả các giá trị trong từ điển

Để xem từng ngôn ngữ được chọn mà không bị lặp lại, chúng ta có thể sử dụng một tập hợp. Set là một tập hợp trong đó mỗi mục phải là duy nhất

```
favorite_languages = {
        'jen': 'python',
        'sarah': 'c',
        'edward': 'ruby',
        'phil': 'python',
}
print("The following languages have been mentioned:”)
for language in set(favorite_languages.values()):
    print(language.title())
```

```
The following languages have been mentioned:
Python
C
Ruby
```

5.4. Nesting

Đôi khi chúng ta muốn lưu trữ nhiều từ điển trong một danh sách hoặc một danh sách các mục dưới dạng một giá trị trong từ điển. Điều này được gọi là nesting.

Chúng ta có thể lồng các từ điển vào bên trong một danh sách, một danh sách các mục bên trong một từ điển hoặc thậm chí một từ điển bên trong một từ điển khác. Nesting là một tính năng mạnh mẽ, như các ví dụ sau đây sẽ chứng minh điều đó.

Danh sách các từ điển

Làm thế nào ta có thể quản lý một đội alien? Một cách là lập danh sách alien, trong đó mỗi alien là một từ điển thông tin về alien đó

```
alien_0 = {'color': 'green', 'points': 5}
alien_1 = {'color': 'yellow', 'points': 10}
alien_2 = {'color': 'red', 'points': 15}

aliens = [alien_0, alien_1, alien_2]
for alien in aliens:
    print(alien)
```

```
{'color': 'green', 'points': 5}
{'color': 'yellow', 'points': 10}
{'color': 'red', 'points': 15}
```

Danh sách các từ điển

Một ví dụ thực tế hơn sẽ liên quan đến hơn ba alien với code tự động tạo ra từng alien. Trong ví dụ sau, chúng ta sử dụng range() để tạo một hạm đội gồm 30 alien

```
# Make an empty list for storing aliens.
aliens = []
①for alien_number in range(30): # Make 30 green aliens.
②   new_alien = {'color': 'green', 'points': 5, 'speed': 'slow'
③   aliens.append(new_alien)
# Show the first 5 aliens.
④for alien in aliens[:5]:
    print(alien)
print("…")
# Show how many aliens have been created.
⑤ print(f"Total number of aliens: {len(aliens)}")
```

Danh sách các từ điển

Ví dụ bắt đầu bằng việc tạo ra một danh sách chứa các aliens. Tại ①, hàm range() trả về một chuỗi các số và nói cho Python biết số lượng các lần lặp của vòng lặp. Mỗi lần lặp, vòng lặp sẽ tạo ra một alien mới ② và đưa alien vào danh sách ③. Tại ④, chúng ta sử dụng danh sách trượt để in ra 5 alien đầu tiên và tại ⑤ chúng ta in ra độ dài của danh sách để chứng minh chúng ta đã tạo ra đầy đủ 30 aliens.

```
{'speed': 'slow', 'color': 'green', 'points': 5}
{'speed': 'slow', 'color': 'green', 'points': 5}
{'speed': 'slow', 'color': 'green', 'points': 5}
{'speed': 'slow', 'color': 'green', 'points': 5}
{'speed': 'slow', 'color': 'green', 'points': 5}
…
Total number of aliens: 30
```

Danh sách các từ điển

Ví dụ: để thay đổi ba alien đầu tiên thành màu vàng, alien có tốc độ trung bình, mỗi alien có giá trị 10 điểm, chúng ta có thể làm như sau:

```
aliens = [] # Make an empty list for storing aliens.
for alien_number in range(30): # Make 30 green aliens.
    new_alien = {'color': 'green', 'points': 5, 'speed': 'slow'}
    aliens.append(new_alien)
for alien in aliens[:3]:
    if alien['color'] == 'green':
            alien['color'] = 'yellow'
            alien['speed'] = 'medium'
            alien['points'] = 10
for alien in aliens[:5]: # Show the first 5 aliens.
    print(alien)
print("...")
```

```
{'speed': 'medium', 'color': 'yellow', 'points': 10}
{'speed': 'medium', 'color': 'yellow', 'points': 10}
{'speed': 'medium', 'color': 'yellow', 'points': 10}
{'speed': 'slow', 'color': 'green', 'points': 5}
{'speed': 'slow', 'color': 'green', 'points': 5}
...
```

Danh sách các từ điển

Mở rộng vòng lặp này bằng cách thêm một khối elif biến những alien màu vàng thành màu đỏ, những alien di chuyển nhanh trị giá 15 điểm mỗi alien. Nếu không hiển thị lại toàn bộ chương trình, vòng lặp đó sẽ giống như sau:

```
for alien in aliens[0:2]:
    if alien['color'] == 'green':
                alien['color'] = 'yellow'
                alien['speed'] = 'medium'
                alien['points'] = 10
    elif alien['color'] == 'yellow':
        alien['color'] = 'red'
        alien['speed'] = 'fast'
        alien['points'] = 15
for alien in aliens[:5]: # Show the first 5 aliens.
    print(alien)
print("...")
```

```
{'color': 'red', 'points': 15, 'speed': 'fast'}
{'color': 'red', 'points': 15, 'speed': 'fast'}
{'color': 'yellow', 'points': 10, 'speed': 'medium'}
{'color': 'green', 'points': 5, 'speed': 'slow'}
{'color': 'green', 'points': 5, 'speed': 'slow'}
...
```

Danh sách trong từ điển

Để sử dụng các mục trong danh sách, chúng ta đặt tên của từ điển và khóa 'toppings', giống như bất kỳ giá trị nào trong từ điển. Thay vì trả về một giá trị duy nhất, ta nhận được danh sách toppings

```
# Store information about a pizza being ordered.
①   pizza = {
    'crust': 'thick',
    'toppings': ['mushrooms', 'extra cheese'],
}
# Summarize the order.
②   print(f"You ordered a {pizza['crust']}-crust pizza "
    "with the following toppings:")
③   for topping in pizza['toppings']:
        print("\t" + topping)
```

```
You ordered a thick-crust pizza with the following toppings:
        mushrooms
        extra cheese
```

Danh sách trong từ điển

Có thể lồng một danh sách vào bên trong từ điển bất kỳ lúc nào ta muốn nhiều giá trị được liên kết với một khóa duy nhất trong từ điển

```
①favorite_languages = {
    'jen': ['python', 'ruby'],
    'sarah': ['c'],
    'edward': ['ruby', 'go'],
    'phil': ['python', 'haskell'],
}
②for name, languages in favorite_languages.items():
    print(f"\n{name.title()}'s favorite languages are:")
    ③for language in languages:
        print(f"\t{language.title()}")
```

```
Jen's favorite languages are:
    Python
    Ruby
Sarah's favorite languages are:
    C
Phil's favorite languages are:
    Python
    Haskell
Edward's favorite languages are:
    Ruby
    Go
```

Từ điển bên trong từ điển

Chúng ta có thể lồng một từ điển bên trong một từ điển khác, nhưng code có thể trở nên phức tạp nhanh chóng khi ta làm như vậy.

```
users = {
    'aeinstein': {
        'first': 'albert',
        'last': 'einstein',
        'location': 'princeton',
    },
    'mcurie': {
        'first': 'marie',
        'last': 'curie',
        'location': 'paris',
    },
}
```

```
①for username, user_info in users.items():
②   print(f"\nUsername: {username}")
③   full_name = f"{user_info['first']} {user_info['last']}"
    location = user_info['location']

④   print(f"\tFull name: {full_name.title()}")
    print(f"\tLocation: {location.title()}")
```

Từ điển bên trong từ điển

Tại ①, chúng ta lặp qua từ điển users. Python sẽ gán mỗi khóa vào biến username và từ điển tương ứng với các username được gán vào biến user_info. Trong vòng lặp chính, chúng ta in ra tên người dùng tại ②.

Tại ③, chúng ta bắt đầu truy cập vào từ điển bên trong. Biến user_info chứa từ điển thông tin của người dùng, bao gồm ba khóa là fisrt, last, và location. Chúng ta sử dụng các khóa này để định dạng tên người dùng cùng vị trí của mỗi người dùng, sau đó in ra kết quả tại bước ④.

```
Username: aeinstein
     Full name: Albert Einstein
     Location: Princeton
Username: mcurie
     Full name: Marie Curie
     Location: Paris
```

Kết chương

Trong chương này, chúng ta đã học cách định nghĩa từ điển và cách làm việc với thông tin được lưu trữ trong từ điển. Ta đã học cách truy cập và sửa đổi các phần tử riêng lẻ trong từ điển và cách lặp lại tất cả thông tin trong từ điển. Ta đã học cách lặp lại từ điển các cặp key-value, các khóa và các giá trị của nó. Chúng ta cũng đã học cách lồng nhiều từ điển trong danh sách, lồng danh sách trong từ điển và lồng từ điển vào bên trong một từ điển.

Trong chương tiếp theo, Chúng ta sẽ tìm hiểu về vòng lặp while và cách nhận đầu vào từ những người đang sử dụng chương trình. Đây sẽ là một điều thú vị vì ta sẽ học cách làm cho tất cả các chương trình có tính tương tác: chúng sẽ có thể phản hồi thông tin đầu vào của người dùng.

Bài tập