Python Set Comprehension

Summary: in this tutorial, you'll learn how to use the Python set comprehension to create a new set based on an existing one.

Introduction to Python Set comprehension

Suppose that you have the following set (https://www.pythontutorial.net/python-basics/python-set/) that consists of three tags:

```
tags = {'Django', 'Pandas', 'Numpy'}
```

To convert the tags in the set to another set of tags in lowercase, you may use the following `for` (https://www.pythontutorial.net/python-basics/python-for-range/) loop:

```
tags = {'Django', 'Pandas', 'Numpy'}

lowercase_tags = set()
for tag in tags:
    lowercase_tags.add(tag.lower())
```

```
print(lowercase_tags)
```

Output:

```
{'django', 'numpy', 'pandas'}
```

How it works:

- First, iterate over each element of the `tags` set.
- Second, convert each tag to lowercase and add it the new set (`lowercase_tags`)

Or you can use the built-in `map()` function with a lambda expression:

```
tags = {'Django', 'Pandas', 'Numpy'}
lowercase_tags = set(map(lambda tag: tag.lower(), tags))

print(lowercase_tags)
```

The `map()` function returns a map object so that you need to use the `set()` function to convert it to a set.

To make the code more concise, Python provides you with the set comprehension syntax as follows:

```
{expression for element in set if condition}
```

The set comprehension allows you to create a new set based on an existing set.

A set comprehension carries the following steps:

- First, iterate over the elements of a set.
- Second, apply an `expression` to each element
- Third, create a new set of elements resulting from the expression.

In addition, the set comprehension allows you to select which element to apply the expression via a `condition` in the `if` clause.

> Note that the set comprehension returns a new set, it doesn't modify the original set.

Back to the previous example, you can convert all the tags in the `tags` set by using the following set comprehension:

```
tags = {'Django', 'Pandas', 'Numpy'}
lowercase_tags = {tag.lower() for tag in tags}

print(lowercase_tags)
```

This syntax definitely looks more concise than a for loop and more elegant than the `map()` function.

Python Set comprehension with an if clause example

Suppose you want to convert all elements of the `tags` set to lowercase except for the `Numpy` .

To do it, you can add a condition to the set comprehension like this:

```
tags = {'Django', 'Pandas', 'Numpy'}
new_tags = {tag.lower() for tag in tags if tag != 'Numpy'}

print(new_tags)
```

Output:

```
{'django', 'pandas'}
```

Summary

- Use Python set comprehension to create a new set based on an existing set by applying an expression to each element of the existing set.