Python List Comprehensions

Summary: in this tutorial, you'll learn about Python List comprehensions that allow you to create a new list from an existing one.

Introduction to Python list comprehensions

In programming, you often need to transform elements of a list (https://www.pythontutorial.net/python-basics/python-list/) and returns a new list.

For example, suppose that you have a list of five numbers like this:

```
numbers = [1, 2, 3, 4, 5]
```

And you want to get a list of squares based on this `numbers` list

The straight forward way is to use a for loop (https://www.pythontutorial.net/python-basics/python-for-loop-list/) :

```
numbers = [1, 2, 3, 4, 5]

squares = []
for number in numbers:
```

```
    squares.append(number**2)

print(squares)
```

In this example, the `for` loop iterates over the elements of the `numbers` list, squares each number and adds the result to the squares list.

> Note that a square number is the product of the number multiplied by itself. For example, square number 2 is 2*2 = 4, square number of 3 is 3*3 = 9, and so on.

To make the code more concise, you can use the built-in `map()` (https://www.pythontutorial.net/python-basics/python-map-list/) function with a lambda expression:

```
numbers = [1, 2, 3, 4, 5]

squares = list(map(lambda number: number**2, numbers))

print(squares)
```

Since the `map()` function returns an iterator (https://www.pythontutorial.net/python-basics/python-iterables/), you need to use the `list()` function to convert the iterator to a list.

Both the `for` loop and `map()` function can help you create a new list based on an existing one. But the code isn't really concise and beautiful.

To help you create a list based on the transformation of elements of an existing list, Python provides a feature called list comprehensions.

The following shows how to use the list comprehension to make a list of squares from the `numbers` list:

```
numbers = [1, 2, 3, 4, 5]
squares = [number**2 for number in numbers]
```

```
print(squares)
```

And here's the list comprehension part:

```
squares = [number**2 for number in numbers]
```

A list comprehension consists of the following parts:

- An input list (`numbers`)
- A variable (https://www.pythontutorial.net/python-basics/python-variables/) that represents the elements of the list (`number`)
- An output expression (`number**2`) that returns the elements of the output list from the elements of the input list.

The following shows the basic syntax of the Python list comprehension:

```
[output_expression for element in list]
```

It's equivalent to the following:

```
output_list = []
for element in list:
    output_list.append(output_expression)
```

Python list comprehension with if condition

The following shows a list of top five highest mountains on Earth:

```
mountains = [
    ['Makalu', 8485],
    ['Lhotse', 8516],
    ['Kanchendzonga', 8586],
```

```
    ['K2', 8611],
    ['Everest', 8848]
]
```

To get a list of mountains where the height is greater than 8600 meters, you can use a `for` loop or the `filter()` (https://www.pythontutorial.net/python-basics/python-filter-list/) function with a lambda expression like this:

```
mountains = [
    ['Makalu', 8485],
    ['Lhotse', 8516],
    ['Kanchendzonga', 8586],
    ['K2', 8611],
    ['Everest', 8848]
]


highest_mountains = list(filter(lambda m: m[1] > 8600, mountains))

print(highest_mountains)
```

Output:

```
[['K2', 8611], ['Everest', 8848]]
```

Like the `map()` function, the `filter()` function returns an iterator. Therefore, you need to use the `list()` function to convert the iterator to a list.

Python List comprehensions provide an optional predicate that allows you to specify a condition for the list elements to be included in the new list:

```
[output_expression for element in list if condition]
```

This list comprehension allows you to replace the `filter()` with a lambda expression:

```
mountains = [
    ['Makalu', 8485],
    ['Lhotse', 8516],
    ['Kanchendzonga', 8586],
    ['K2', 8611],
    ['Everest', 8848]
]

highest_mountains = [m for m in mountains if m[1] > 8600]

print(highest_mountains)
```

Output:

```
[['K2', 8611], ['Everest', 8848]]
```

Summary

- Python list comprehensions allow you to create a new list from an existing one.
- Use list comprehensions instead of `map()` or `filter()` to make your code more concise and readable.