How to Transform List Elements with Python map() Function

Summary: in this tutorial, you'll learn how to use the Python `map()` function with lists.

Introduction to the Python map() function

When working with a list (or a tuple), you often need to transform the elements of the list and return a new list that contains the transformed element.

Suppose, you want to double every number in the following `bonuses` list:

```
bonuses = [100, 200, 300]
```

To do it, you can use a for loop to iterate over the elements, double each of them, and add it to a new list like this:

```
bonuses = [100, 200, 300]

new_bonuses = []

for bonus in bonuses:
    new_bonuses.append(bonus*2)

print(new_bonuses)
```

Output:

```
[200, 400, 600]
```

Python provides a nicer way to do this kind of task by using the `map()` built-in function.

The `map()` function iterates over all elements in a list (or a tuple), applies a function to each, and returns a new iterator of the new elements.

The following shows the basic syntax of the `map()` function:

```
iterator = map(fn, list)
```

In this syntax, `fn` is the name of the function that will call on each element of the list.

In fact, you can pass any iterable to the `map()` function, not just a list or tuple.

Back to the previous example, to use the `map()` function, you define a function that doubles a bonus first and then use the map() function as follows:

```
def double(bonus):
    return bonus * 2


bonuses = [100, 200, 300]

iterator = map(double, bonuses)
```

Or you make this code more concise by using a lambda expression like this:

```
bonuses = [100, 200, 300]
iterator = map(lambda bonus: bonus*2, bonuses)
```

Once you have an iterator, you can iterate over the new elements using a `for` loop.

Or you can convert an iterator to a list by using the the `list()` function:

```
bonuses = [100, 200, 300]
iterator = map(lambda bonus: bonus*2, bonuses)
print(list(iterator))
```

More examples of Python map() function with lists

Let’s take some more examples of using the Python map() function with lists.

1) Using the Python map() function for a list of strings

The following example uses the `map()` function to return a new list where each element is transformed into the proper case:

```
names = ['david', 'peter', 'jenifer']
new_names = map(lambda name: name.capitalize(), names)
print(list(new_names))
```

Output:

```
['David', 'Peter', 'Jenifer']
```

2) Using the Python map() function to a list of tuples

Suppose that you have the following shopping cart represented as a list of tuples:

```
carts = [['SmartPhone', 400],
         ['Tablet', 450],
         ['Laptop', 700]]
```

And you need to calculate the tax amount for each product with a 10% tax 10%. In addition, you need to add the tax amount to the third element of each item in the list.

The return list should be something like this:

```
[['SmartPhone', 400, 40.0],
['Tablet', 450, 45.0],
['Laptop', 700, 70.0]]
```

In order to do so, you can use the map() function to create a new element of the list and add the new tax amount to each like this:

```
carts = [['SmartPhone', 400],
         ['Tablet', 450],
         ['Laptop', 700]]

TAX = 0.1
```

```
carts = map(lambda item: [item[0], item[1], item[1] * TAX], carts)


print(list(carts))
```

Summary

- Use the Python map() to call a function on every item of a list and returns an iterator.