Python List Comprehensions

In programming, you often need to transform elements of a list and returns a new list

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

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

list = []
for i in l:
    list.append(i*i)
    # list.append(i**2)

print(list)
```

Python map() function

syntax of the map() function:

iterator = map(fn, list)

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.

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

def get(list):
    return list**2

value=map(get,l)

for i in value:
    print(i)
```

lambda function using

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

value=map(lambda list:list**2,l)
print(value)

for i in value:
    print(i)
```

Another Example

uses the map() function to returns a new list where each element is transformed into the proper case:

```
l=['gautam','vivek','shailesh']

l1=map(lambda list:list.capitalize(),l)
print(list(l1))
```

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

list comprehensions.

```
list = [1, 2, 3, 4, 5]
list1 = [i**2 for i in list]

print(list1)
```

A list comprehension consists of the following parts:

- An input list (list)
- A variable that represents the elements of the list (i)
- An output expression (i**2) that returns the elements of the output list from the elements of the input list.

[output_expression for element in list]