Python lambda (Anonymous Functions) | filter, map, reduce

In Python, anonymous function means that a function is without a name. As we already know that *def* keyword is used to define the normal functions and the *lambda* keyword is used to create anonymous functions. It has the following syntax:

lambda arguments: expression

- This function can have any number of arguments but only one expression, which is evaluated and returned.
- One is free to use lambda functions wherever function objects are required.
- You need to keep in your knowledge that lambda functions are syntactically restricted to a single expression.
- It has various uses in particular fields of programming besides other types of expressions in functions.

Use of lambda() with filter()

The filter() function in Python takes in a function and a list as arguments. This offers an elegant way to filter out all the elements of a sequence “sequence”, for which the function returns True.

syntax: filter(function, sequence)

Use of lambda() with map()

The map() function in Python takes in a function and a list as argument. The function is called with a lambda function and a list and a new list is returned which contains all the lambda modified items returned by that function for each item.

syntax: map(function, iterable, ...)

Use of lambda() with reduce()

The reduce() function in Python takes in a function and a list as argument. The function is called with a lambda function and a list and a new reduced result is returned. This performs a repetitive operation over the pairs of the list.

The reduce() function accepts a function and a sequence and returns a single value calculated as follows:

Initially, the function is called with the first two items from the sequence and the result is returned.

The function is then called again with the result obtained in step 1 and the next value in the sequence. This process keeps repeating until there are items in the sequence.

The syntax of the reduce() function is as follows:

Syntax: reduce(function, sequence[, initial]) -> value

When the initial value is provided, the function is called with the initial value and the first item from the sequence.

In Python 2, reduce() was a built-in function. However, in Python 3, it is moved to functools module. Therefore to use it, you have to first import it as follows:

```
from functools import reduce # only in Python 3
```