Lambda Operator in Python

A booon to Python Programmer

Akhilesh Pandey

December 15, 2017

Contents

1 Introduction

The Lambda operator or lambda function is used when we want to create a small anonymous function, i.e. a function without a name. These functions are throw-away or use and throw functions, i.e. they are just needed where they have been created. The lambda functions are useful when we are working with functions like filter(), map() and reduce().We will see some examples related to this a little later in this article and probably that will also make clear to the readers why I named this article as " A boon to Python Programmers " .

2 Syntax of lambda operator

The syntax of a lambda operator is :

lambda ⟨ *argument list* ⟩ ⟨ *expression* ⟩. The argument list consists of a list containing arguments separated by comma(,) and expression is an arithmetic expression which must be applied on the arguments provided in the argument list.

3 Advantages of using Python lambda

3.1 Easy to create and use as and when required

The lambda function can be assigned to a variable for improving readability. For example consider a lambda function that take two arguments x and y and returns their sum. We can define a lambda function as:

f= lambda x,y : x+y.

Note that the function is assigned to a variable f. So, when we call f(1,1), it outputs 2.

3.2 It's easy to use in combination with other functions

Apart from its ease to create and use, the advantages of lambda operator is seen when it is used in combination with the map(), reduce(), filter() functions as explained below.

3.2.1 The map() function:

Let us define a function that take a number as its argument and returns its square:

```
    def squareNum(x):
return x*x

    Now create a list of numbers and use map function
    list=[1,2,3,4,5]
newList= map(squareNum, list)
print newList
output: [1,4,9,16,25]
```

However, if we used lambda operator then we don't have to define the function squareNum. The above can be written using lambda as:

```
num=[1,2,3,4,5]
newNum= map(lambda x: x*x , num)
print newNum
output: [1,4,9,16,25]
```

Please check for yourself.

3.2.2 The filter() function:

The filter function is used to filter out all elements of a list for which the function (passed as argument to filter) return True. The function filter(f,l) takes a function f as its first argument. f returns True or False. This function f is applied to every element of the list l. The elements of l are included in the result list only if the f returns true for that element.

```
    L=[1,2,3,4,5,6]
newL= filter(lambda x : x%2 == 0, L)
print newL
output: [2,4,6]
```

3.2.3 The reduce() function:

The function reduce(f,l) applies f to the list l and return a single value. To understand how reduce function works, consider a list l=$[s_1, s_2, s_3, s_4, ..., s_n]$. At first two elements of l will be applied to f which returns say p_1. Now the list l becomes l=$[p_1, s_3, s_4, ..., s_n]$. Please note the length of list reduced by one. Now apply f on p_1 and s_3 which return p_2(say). The list reduces to l=$[p_2, s_4, ..., s_n]$. Finally apply p_{n-2} and s_n to get p_n. Hence finally we have a single value.

3.3 Conclusion

The use of lambda makes easy to create a function and use where required. From its advantages as mentioned above we see programming becomes much easier. Debugging becomes easier because we are using a function where we have created. Thus it can be concluded that lambda is a boon for python programmers.