Python - Define Anonymous Functions using Lambda

The def keyword is used to define a function in Python. The lambda keyword is used to define an anonymous functions in Python. Usually, such a function is meant for one-time use.

Syntax:

```
lambda arg1, arg2... : expression
```

The lambda function can have any number of arguments but there's always a single expression after the : symbol. When this function is called, the expression is its return value.

```
>>>square = lambda x : x * x
```

Here, the lambda expression x : x * x is assigned to a variable square. This lambda function receives x as a parameter and returns the square of x (x * x). We can now call it as a regular function call.

```
>>>square = lambda x : x * x
>>>square(5)
25
```

The above lambda function definition is similar to definition of a normal function, as shown below:

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

Consider the following lambda function with multiple parameters:

```
>>>sum = lambda x, y, z : x + y + z
>>>sum(5, 10, 15)
30
```

The expression does not need to always return a value. It can be void.

```
>>>disp = lambda str: print('Output: ' + str)
>>>disp("Hello World!")
```
Output: Hello World!

The lambda function can have only one expression (not statement). Obviously, it cannot substitute a function whose body may have conditionals, loops etc.

Lambda functions are useful when we want to give the function as one of the arguments to another function. In Python, the function is a first order object, which means that just as number, string, list, etc. the function can also be used as an argument.

Python has built-in functions which take other functions as arguments. The map(), filter() and reduce() functions are important functional programming tools. All of them take a function as their argument. The argument function can be a normal function or a lambda function.