Name: Karam Rai
Regd. No: 17554

Python Lambda:

In Python, anonymous function is a function that is defined without a name. While normal functions are defined using the **def** keyword, in Python anonymous functions are defined using the **lambda** keyword. Hence, anonymous functions are also called **lambda** functions.

It is often used in functions such as sorted() that take a callable as a parameter (often the key keyword parameter). You could provide an existing function instead of a lambda there too, as long as it is a callable object.

These functions are throw-away functions, i.e. they are just needed where they have been created.

Therefore, Python has two tools for building functions: **def** and **lambda**.

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.

This piece of code shows the difference between a normal function definition ("f") and a lambda function ("g"):

```
>>>def f(x): return x**2
. . .
>>> print f(8)
64
>>>
>>> g = lambda x: x**2
>>>
>>> print g(8)
64
```

As you can see, f() and g() do exactly the same and can be used in the same ways. Note that the lambda definition does not include a "return" statement -- it always contains an expression which is returned. Also note that you can put a lambda definition anywhere a function is expected, and you don't have to assign it to a variable at all.

Lambda functions can be used along with built-in functions like filter(), map() and reduce().

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.

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.

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.