Python Anonymous/Lambda Function

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.

Lambda Functions

A lambda function has the following syntax.

Syntax of Lambda Function

```
lambda arguments: expression
```

Lambda functions can have any number of arguments but only one expression. The expression is evaluated and returned. Lambda functions can be used wherever function objects are required.

Example of Lambda Function

Here is an example of lambda function that doubles the input value.

```

# Program to show the
# use of lambda functions

double = lambda x: x * 2

print(double(5))
```

Output

```

10
```

Explanation

In the above program, `lambda x: x * 2` is the lambda function. Here *x* is the argument and `x * 2` is the expression that gets evaluated and returned. This function has no name. It returns a function object which is assigned to the identifier `double`. We can now call it as a normal function. The statement

```
double = lambda x: x * 2
```

is nearly the same as

```
def double(x):
   return x * 2
```

Use of Lambda Function

We use lambda functions when we require a nameless function for a short period of time. In Python, we generally use it as an argument to a higher-order function (a function that takes in other functions as arguments). Lambda functions are used along with built-in functions like `filter()`, `map()` etc.

Example use with filter()

The `filter()` function in Python takes in a function and a list as arguments. The function is called with all the items in the list and a new list is returned which contains items for which the function evaluats to `True`.

Here is an example use of `filter()` function to filter out only even numbers from a list.

```
# Program to filter out
# only the even items from
# a list using filter() and
# lambda functions

my_list = [1, 5, 4, 6, 8, 11, 3, 12]

new_list = list(filter(lambda x: (x%2 == 0) , my_list))
print(new_list)
```

Output

```
[4, 6, 8, 12]
```

Example use with map()

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

Here is an example use of `map()` function to double all the items in a list.

```
# Program to double each
# item in a list using map() and
# lambda functions

my_list = [1, 5, 4, 6, 8, 11, 3, 12]
```

```
new_list = list(map(lambda x: x * 2 , my_list))
print(new_list)
```

Output

```
[2, 10, 8, 12, 16, 22, 6, 24]
```