Functions

Map:

Reduce

Filter

Anonymous functions:

Types of Arguments (formal arguments ,keyword arguments , default arguments)

Anonymous Function or lambda Function:

In Python, anonymous function is a [function](https://www.programiz.com/python-programming/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.

A lambda function has the following syntax.

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.

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

# Output: 10

print(double(5))

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](https://www.programiz.com/python-programming/function-argument)). Lambda functions are used along with built-in functions like filter (), map () etc.

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 evaluates 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

my\_list = [1, 5, 4, 6, 8, 11, 3, 12]

new\_list = list(filter(lambda x: (x%2 == 0) , my\_list))

# Output: [4, 6, 8, 12]

print(new\_list)

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()

my\_list = [1, 5, 4, 6, 8, 11, 3, 12]

new\_list = list(map(lambda x: x \* 2 , my\_list))

# Output: [2, 10, 8, 12, 16, 22, 6, 24]

print(new\_list)

Reduce:

The function reduce(func, seq) continually applies the function func() to the sequence seq. It returns a single value.

>> reduce(lambda x,y: x+y, [47,11,42,13])

113

List comprehension:

List comprehensions provide a concise way to create lists.

It consists of brackets containg an expression followed by for clasue.then zero or more for or if clauses .

new\_list = [expression(i) for i in old\_list if filter(i)]

[ expression for item in list if conditional ]

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Keywords in Python programming language | | | | |
| False | class | finally | is | return |
| None | continue | for | lambda | try |
| True | def | from | nonlocal | while |
| and | del | global | not | with |
| as | elif | if | or | yield |
| assert | else | import | pass |  |
| break | except | in | raise |  |