**How to use Python Lambdas**

**Introduction to the Lambda Function in Python**

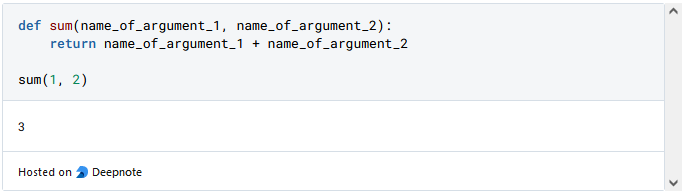


Photo by [Henry & Co.](https://unsplash.com/@hngstrm?utm_source=medium&utm_medium=referral) on [Unsplash](https://unsplash.com?utm_source=medium&utm_medium=referral)

Python lambdas are so-called anonymous functions with which you can quickly define functions that can have multiple inputs, but only one output. Such anonymous functions are not only used in [Python](https://databasecamp.de/en/python-coding), but also in other programming languages like Java, C#, or C++.

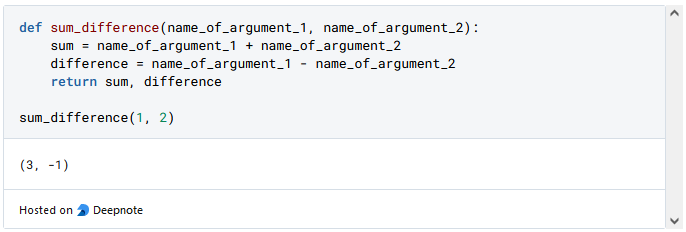
**How are normal functions defined in Python?**

In [Python](https://databasecamp.de/en/python-coding), the definition of a function is marked with “def”. Then its name is defined and the arguments with names are listed in round brackets. The following function named “sum” simply takes two arguments and adds them together:



The advantage of these functions is that they can be called via a unique name. This helps both to better describe the functionality and to reference the function again at a later time. In addition, the arguments can also have names that help to better understand the result. Another advantage is that explicitly defined functions can also output more than one argument.

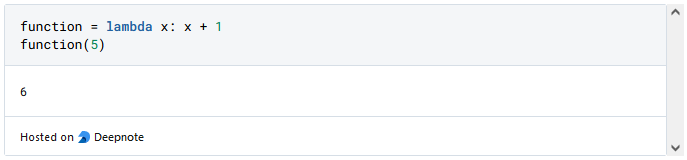
Our initial function, for example, can output the difference between the two arguments in addition to the sum:



**What are anonymous functions?**

However, there is also a second way how functions can be defined in [Python](https://databasecamp.de/en/python-coding). With the help of so-called anonymous functions, these can be defined with just a few lines, usually even just one. These anonymous functions also exist in other programming languages, such as Java or C#. In [Python](https://databasecamp.de/en/python-coding), the argument “lambda” is used for this, which is why anonymous functions in [Python](https://databasecamp.de/en/python-coding) are often simply called Python lambdas.

These functions can be easily defined with the parameter “lambda”, the naming of the variables, and the expression that the function should calculate:

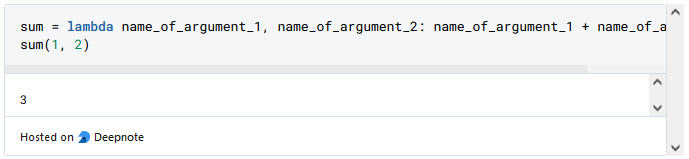


This example also clearly shows why Python lambdas are anonymous functions: The function itself cannot be given a name, it can only be stored in a variable (“function”).

So if we want to recreate the sum function from our initial example using Python Lambdas, here’s what it looks like:

sum = lambda name\_of\_argument\_1, name\_of\_argument\_2: name\_of\_argument\_1 + name\_of\_argument\_2

sum(1, 2)



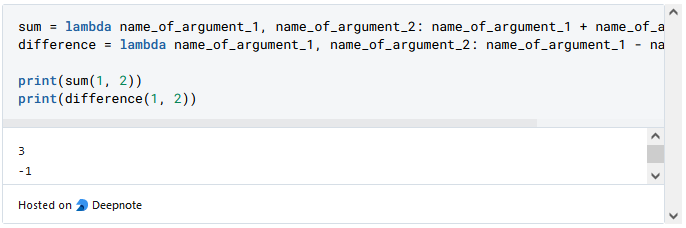
As you can see, more than one argument can be passed to Python Lambdas. However, we cannot recreate the function “sum\_difference” with Python lambdas, because the anonymous functions can only output one result. So instead, two different functions must be defined and also called twice:

sum = lambda name\_of\_argument\_1, name\_of\_argument\_2: name\_of\_argument\_1 + name\_of\_argument\_2

difference = lambda name\_of\_argument\_1, name\_of\_argument\_2: name\_of\_argument\_1 - name\_of\_argument\_2

print(sum(1, 2))

print(difference(1, 2))



**Why should you use Python Lambdas?**

There are several cases where it can be useful to use Python lambdas:

* When simple functions are defined that have only one output and a small number of inputs. In large projects it can make sense not to define such a function explicitly, in order to save space and not to cause unnecessary confusion. Furthermore one saves problems with the inheritance, etc. in extensive classes.
* The same argument is also valid if functions are used only once. Then one can do without the explicit definition since anyway nobody must access the function again.
* In addition, anonymous functions can ensure that the content of the function is faster and easier to understand since it is defined in a single line. Explicitly defining functions can result in losing some comprehensibility.
* There are functions in Python, such as “filter” or “map”, that take functions as input. So in these cases, it makes sense to use Python Lambda.

**How does the filter() function work with Python lambdas?**

The function “filter()” can be used to filter a [list](https://databasecamp.de/en/python-coding/python-lists) with a certain condition. The result is a new [list](https://databasecamp.de/en/python-coding/python-lists) that consists only of those elements that meet the condition. Without the explicit function you could also solve this as follows:

list\_1 = [1, 2, 3, 4, 5, 6]

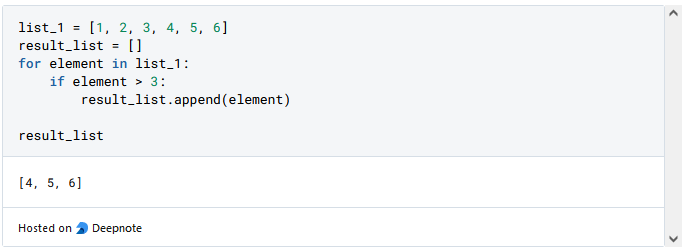
result\_list = []

for element in list\_1:

if element > 3:

result\_list.append(element)

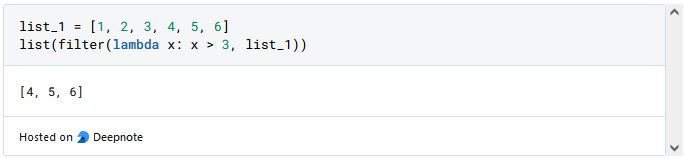
result\_list



The function “filter()” has a total of two inputs, firstly the [list](https://databasecamp.de/en/python-coding/python-lists) to be filtered and secondly the condition to be used for filtering. This filter condition must be a function, which is why Python lambdas are suitable for this:

list\_1 = [1, 2, 3, 4, 5, 6]

list(filter(lambda x: x > 3, list\_1))



For such applications, Python Lambdas are of course also an optimal alternative.

**This is what you should take with you**

* Anonymous functions are used to quickly define functions that do not have to have a name.
* This can be useful if functions are to be used only once or if you want to keep the namespace of variables and functions small.
* In Python, anonymous functions are used with the parameter “lambda”, which is why anonymous functions in Python are also called Python lambdas.