# What is Lambda Function?

Do you remember when you were still going to school? Can you recall the moments of frustration that math used to have on you and your grades? The moments when you said to yourself “I’m never going to need math in my life!” Those moments most likely are giving you trouble if you decided to be a programmer later in your life. Mathematics is the essence of programming and creating functions. In computer programming, the lambda function is also known as the anonymous function, lambda abstraction, lambda expression, and a function literal. It is a function definition that is not tied to a name or the identifier. The anonymous functions are usually arguments that are passed to higher-order functions or even used for building the result of a higher-order function that must return a function. In case that the feature has been used only once or even in the limited number of times, an anonymous function might be lighter syntactically, than using a named one. Anonymous functions are found everywhere in functional programming languages, including other languages consisted of the first-class functions. That’s where the functions complete the same role for the function type as literals do for different types of data.

## How Did Lambda Function Came To Existence?

Like everything else in life which is fantastic, some genius needed to think of that “something” first. Origins of anonymous functions date back to 1936 when Alonzo Church invented the lambda calculus which was long before electronic computers came into existence, therefore all the functions were anonymous. Several programming languages introduced anonymous functions by the keyword Lambda. Anonymous functions are often called lambdas or lambda abstraction. Anonymous functions were a part of programming languages even since Lisp in 1958. Even today, vast and ever-growing number of modern programming languages support anonymous functions and hopefully you will promote it too by using it if you haven’t already.

## Using Lambda Functions

Lambda functions can be used for holding the functionality that doesn’t need to be named, and they are also excellent for short-term use. Some of the examples would be currying and closures. Using anonymous functions is not always the only solution to solving the issue or problem you’ve encountered. Every anonymous function can be defined as a named function and therefore, called by the name. Whether you will use or not use the anonymous function, it’s just a matter of personal programming style. Programmers sometimes use the anonymous function to enclose a particular and non-reusable code without messing up the code with a lot of small one-line standard functions. In some programming languages, it is entirely reasonable to implement lambda functions for particular purposes like binding events to a callback or to instantiate the function for some specific values. Some of those values might be more efficient, more readable and less susceptible to error than calling a more-generic named function. If an overview of all functions is something you might use, visit our website at https://dashbird.io.

## Lambda Functions Sorting, Closures, And Currying

Which programming language is your favorite? I’m a fan of Python, and therefore I’ll give an explanation based on Python 3. If you try to sort in a non-standard way, it might be easier enclose the sorting logic as an anonymous function. If not, you’d need to create a named function. Most programming languages do provide you with a generic sort function that puts a sort algorithm into action that will then sort the arbitrary objects. In this case, this function accepts typically an arbitrary function that will determine how to make a comparison between to elements to define if one is greater or lesser than the other. Imagine sorting a list of a particular object by the length of each specific object.

Closures are functions that are evaluated in an environment that is enclosing bound variables. It is impractical creating a function for every single comparison function. Whatever the reason is for using the closure, the anonymous function is the entity containing the functionality that does the comparing.

Currying is the function changing process so that the function will take fewer inputs. Utilizing anonymous functions might not be common with currying, but it still can be used.

## Do All Programming Languages support lambda Functions (Anonymous Functions)?

Various programming languages support anonymous functions entirely or even partly as some variants, or not supporting anonymous functions at all. The programming languages that don’t support anonymous functions are C, Pascal, and Object Pascal as all of them are statically typed languages. Meanwhile, some statically typed languages can support anonymous functions like ML which is statically typed language but also supports anonymous functions. A dialect of Object Pascal is the Delphi programming language which has been upgraded to support the anonymous functions, as well as the C++ (by the C++11 standard). There are programming languages which consider functions as first-class functions and generally have anonymous function support allowing the functions to be easily defined and passed like any other kind of data types. Some of those languages are JavaScript, Lisp, Python, Ruby, ML, Perl, Dylan and others. If I forgot to mention some, please add them in the comment section.

## A Real-Time, Practical Usage Of Anonymous Function

Anonymous functions are handy in cases of name collision. To avoid name collisions, using the anonymous functions will help you out big time. Frequent method names are a bit of distraction for programmers. When you’re thinking about the logic and constructs, thinking about method names can be a big distraction. Therefore, it is quite convenient to write a function anonymously, and when multiple different places are using it, you can make it a named one and chose into which namespace it should be used. Having a real-time log of your functions can be quite convenient. If you agree, feel free to explore our site at https://dashbird.io and discover all the other quality features of AWS lambda that we offer to our users.

## Conclusion

If you are still here, I’m happy to tell you that I will try to come up with a brief conclusion of everything in this article. My conclusion would be that the first significant point about lambdas is that they most likely create closures – a block of code which binds to the environment that they came from. There are also lambdas that don’t develop closures, but as such, they aren’t entirely useful and thus, not very frequent. That is the reason why the closure is used as an alternative term for lambda. The second difference is not mainly defined as a difference, but it is just as important, if not more so in practice. Lambda supporting languages allow you to use minimal syntax to identify them. It might not seem like an important point, but it also might be crucial because it is the key to make it natural for using them often. If you take a look at Ruby, Lisp or even Smalltalk, you will notice that lambdas are used all over. Also, they are much more frequently used than some other similar structures in other languages. The ability they possess, which is to bind to local variables is also the part of that, but the most significant reason might be that the notation in using them is quite simple and clear. Let me know if I've missed something in the comment section below.