Anonymous functions (C# Programming Guide): <https://docs.microsoft.com/en-us/dotnet/csharp/programming-guide/statements-expressions-operators/anonymous-functions>

“You can use a [lambda expression](https://docs.microsoft.com/en-us/dotnet/csharp/language-reference/operators/lambda-expressions) or an [anonymous method](https://docs.microsoft.com/en-us/dotnet/csharp/language-reference/operators/delegate-operator) to create an anonymous function. We recommend using lambda expressions as they provide more concise and expressive way to write inline code” 20-July-2015

. A lambda expression that has one parameter and returns a value can be converted to a [Func<T,TResult>](https://docs.microsoft.com/en-us/dotnet/api/system.func-2) delegate. In the following example, the lambda expression x => x \* x, which specifies a parameter that's named x and returns the value of x squared, is assigned to a variable of a delegate type:

According to Petey from The Startup, a lambda expression, in the world of computer programming, is a function not bound to an identifier and it can be used as a variable or passed in as a method parameter.

According to Microsoft C# Programming Guide (Microsoft, 2015), lambda expression was introduced in C# 3.0 which makes anonymous methods a much more concise code. Figure 1 below shows the evolution of delegate creation from C# 1.0 through C# 3.0.

References: