

Applause from Cory Althoff, Tariq Ellis, and 6 others

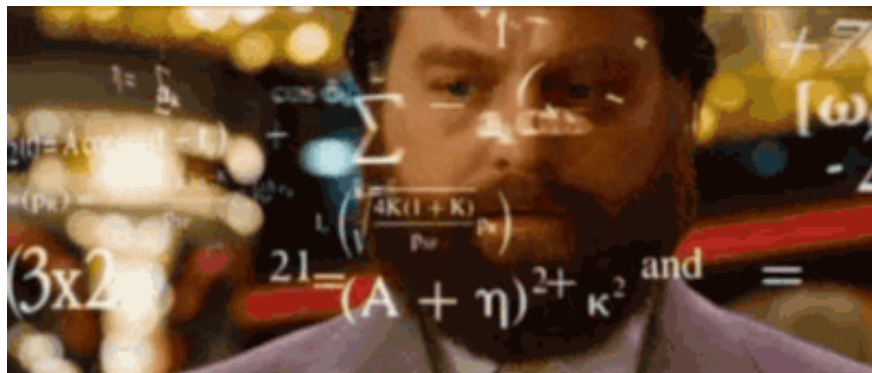


Adrian D. Finlay

@thewiprogrammer. Writer @hackernoon. Code, LOTS of it. Mangos, LOVE THEM! Barbering. Health. Travel. Business. & more! Network w/ me @ adriandavid.me/network

Nov 12, 2017 · 4 min read

Lambda Expressions in Java, Python, C#, C++



GfyCat

No, Lambda Expressions are not as complicated as Church's Lambda Calculus. No need to worry.

Lambda Expressions are simply an expression of an anonymous function. This is a very important concept in Functional Programming and is something like a literal expression for a function (such as how 1020L may be interpreted as a long integer literal in certain languages). This is a widespread, cornerstone concept in Functional Programming that has been added to OOP/Procedural centered languages like Java to reap some of the benefits of functional programming, which is useful for things like processing data. It also provides neat, concise syntax for implementing

functions on the fly. They are based off functional programming concepts used in the construction of expression trees.

In this article, I will demonstrate basic use among some common, popular languages. Instead of engaging in a tutorial of lambda expressions in each language, I will instead simply show the same example in various languages. Note also that I will not cover a language's full coverage of Lambda Expressions (or the equivalent) rather, some basic use to get your feet wet.

I decided to piggyback off of [Quincy Larson](#)'s recommendation to use Medium's **code blocks** but because they are visually unappealing to me and **an utter nuisance to indent**, I will not be using them again. This is, of course, not the fault of Quincy Larson. I have other small issues with this format but it is an OK start for small snippets. I will perhaps, as Larson also recommends, resort to GitHub gists.

Let's start with Java.

Java

Java provides support for Lambda Expressions via Functional Interfaces. Let's take a peek. Do note that because of my qualms with the format, I have not included the full source below. Look towards the end of the article for the full source.

```
import java.util.Scanner; //For Scanner class
import java.lang.Math; //For pow(), min()
import java.util.InputMismatchException;
//InputMismatchException class
import static java.lang.System.out; //For println(),
print()

//Defines Structure for Lambda Expression
@FunctionalInterface
interface MathOp {
    //Must contain one and only one abstract method
    double binaryMathOp(double a, double b);
}

//Main Class
public class Lambda {
```

```

//Main method
public static void main (String[] args) {

    //Input Stream
    Scanner in = new Scanner(System.in);

    //Lambda Reference, Arithmetic Expressions
    MathOp doMath;
    double expr1=0, expr2 =0;

    //Header
    out.println("\nLet's demonstrate the usefulness of
Airthmetic\n" +
                "operations by using Lambda
Expressions.\n");

    //Prompt for Division
    out.print("LET'S DO SOME DIVISION:\t");
    try{
        expr1 = in.nextDouble();
        expr2 = in.nextDouble();
    } catch (InputMismatchException e) {
        out.println("\n\nYou have entered invalid
input." +
                    "Restarting.\n\n");
        main(args);
    }
    //Lambda Expression for Division
    doMath = (x,y) -> { return x / y; };

    //Print Result
    out.println("The result is:\t" +
doMath.binaryMathOp(expr1, expr2));

    //Method References

    // .... <Omitted>

    //Prompt to find minimum value
    out.print("\nLET'S FIND THE MINIMUM VALUE:\t");

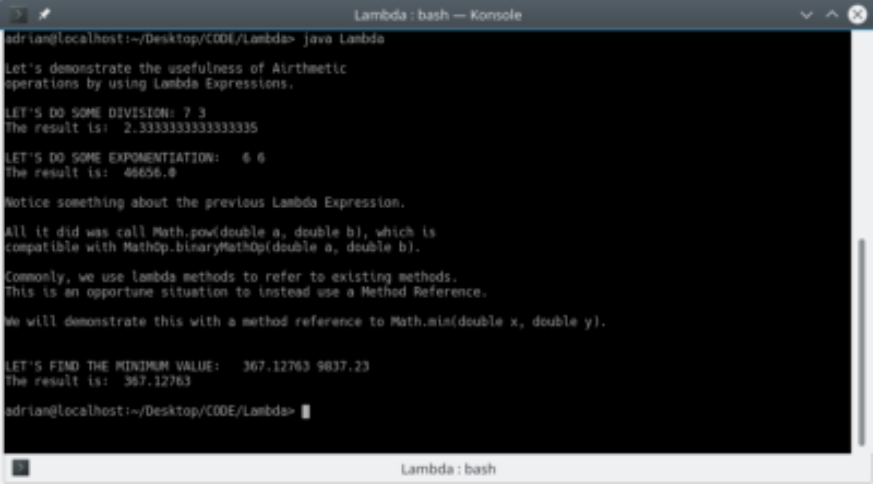
    try{
        expr1 = in.nextDouble();
        expr2 = in.nextDouble();
    } catch (InputMismatchException e) {
        out.println("\n\nYou have entered invalid
input." +
                    "Restarting.\n\n");
        main(args);
    }
    //Method Reference (Lambda Expression)
    doMath = Math::min;

```

```
        //Print Result
        out.println("The result is:\t" +
doMath.binaryMathOp(expr1, expr2) + "\n");

        //Close Input Stream
        in.close();

    }
}
```



```
adrian@localhost:~/Desktop/CODE/Lambda> java Lambda

Let's demonstrate the usefulness of Airthmetic
operations by using Lambda Expressions.

LET'S DO SOME DIVISION: 7 3
The result is: 2.3333333333333335

LET'S DO SOME EXPONENTIATION: 6 6
The result is: 46656.0

Notice something about the previous Lambda Expression.

All it did was call Math.pow(double a, double b), which is
compatible with MathOp.binaryMathOp(double a, double b).

Commonly, we use lambda methods to refer to existing methods.
This is an oportune situation to instead use a Method Reference.

We will demonstrate this with a method reference to Math.min(double x, double y).

LET'S FIND THE MINIMUM VALUE: 367.12763 9837.23
The result is: 367.12763

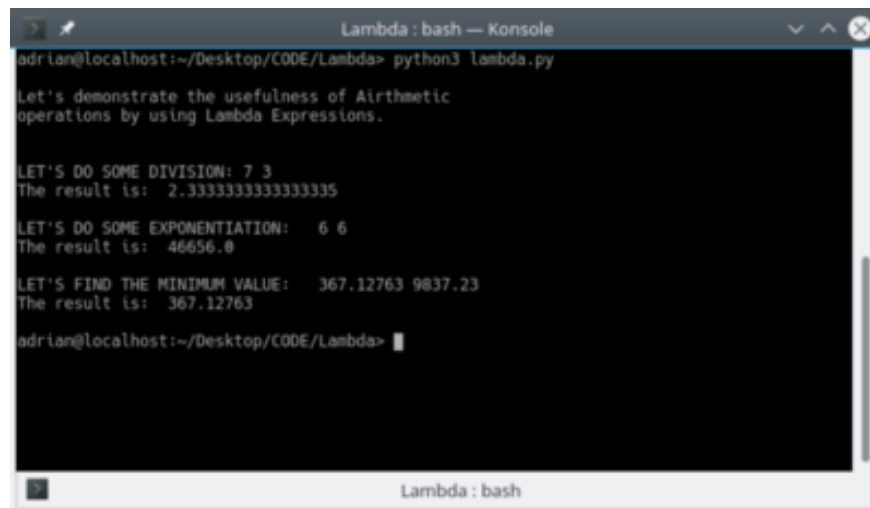
adrian@localhost:~/Desktop/CODE/Lambda>
```

Python

Lambda Expressions work a little differently in python. In python, Lambda Expressions are essentially anonymous functions. Also, note that unlike Java, we can define it's behavior on the fly in the same way.

```
Welcome  Lambda.java  lambda.py x  Untitled-1
Set an interpreter
1  #!/usr/bin/python
2
3  #Lambda Expressions, Expression List
4  div = lambda x,y: x/y
5  exp = lambda x,y: x**y
6  _min = lambda x,y: min(x,y)
7  expr = []
8
9  #main()
10 def main():
11     #Header Prompt
12     print("\nLet's demonstrate the usefulness of Airthmetic\n" +
13           "\noperations by using Lambda Expressions.\n")
14
15     #One can always escape the program by entering the EOF character (such as CTRL+D)
16     #on their shell triggering an EOF Error, ending the application.
17
18     #Let's do some division
19     print("\nLET'S DO SOME DIVISION:\t", end="")
20     try:
21         expr = input().split()
22         expr1 = float(expr[0])
23         expr2 = float(expr[1])
24     except (ValueError,IndexError):
25         print("\n\nYou have entered invalid input. Restarting.\n\n")
26         main()
27     #Call the Lambda Expression, Print the result
28     print("The result is:\t" + str(div(expr1, expr2)) )
29
```

```
Welcome  Lambda.java  lambda.py x  Untitled-1
30 #Let's do some exponentiation
31 print("\nLET'S DO SOME EXPONENTIATION:\t", end="")
32 try:
33     expr = input().split()
34     expr1 = float(expr[0])
35     expr2 = float(expr[1])
36 except (ValueError,IndexError):
37     print("\n\nYou have entered invalid input. Restarting.\n\n")
38     main()
39 #Call the Lambda Expression, Print the result
40 print("The result is:\t" + str(exp(expr1, expr2)) )
41
42 #Let's find the minimum value
43 print("\nLET'S FIND THE MINIMUM VALUE:\t", end="")
44 try:
45     expr = input().split()
46     expr1 = float(expr[0])
47     expr2 = float(expr[1])
48 except (ValueError,IndexError):
49     print("\n\nYou have entered invalid input. Restarting.\n\n")
50     main()
51 #Call the Lambda Expression, Print the result
52 print("The result is:\t" + str(_min(expr1, expr2)) )
53
54 #Run It
55 if __name__ == "__main__":
56     main()
57     print()
58
```



```
adrian@localhost:~/Desktop/CODE/Lambda> python3 lambda.py

Let's demonstrate the usefulness of Airthmetic
operations by using Lambda Expressions.

LET'S DO SOME DIVISION: 7 3
The result is: 2.3333333333333335

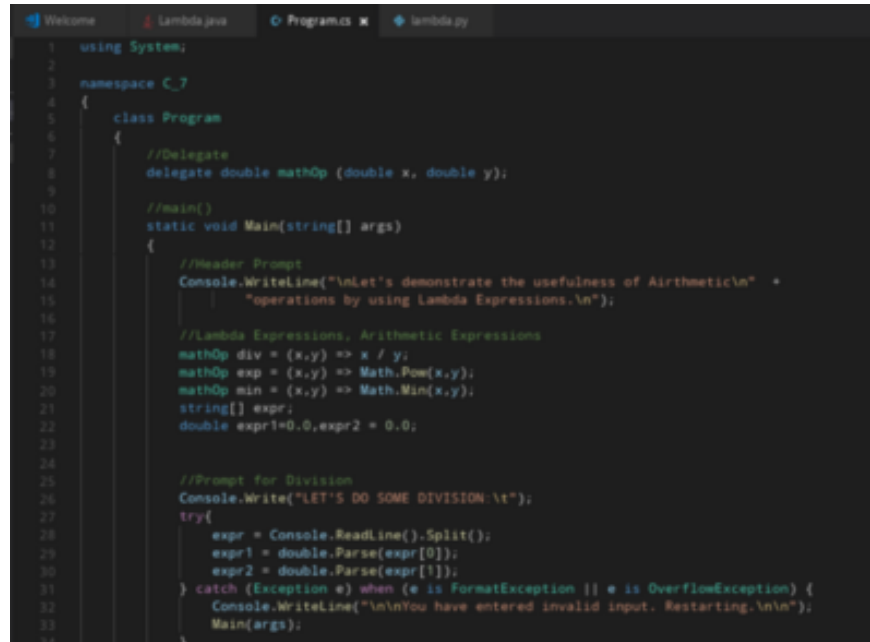
LET'S DO SOME EXPONENTIATION: 6 6
The result is: 46656.0

LET'S FIND THE MINIMUM VALUE: 367.12763 9837.23
The result is: 367.12763

adrian@localhost:~/Desktop/CODE/Lambda> 
```

C#

In C#, Lambda Expressions are anonymous functions that can be used to create expression trees or delegates. We will discuss creating a delegate, which is something like a function pointer. This is much closer to the nature of a Lambda Expression than that of Java and Python.



```
1 using System;
2
3 namespace C_7
4 {
5     class Program
6     {
7         //Delegate
8         delegate double mathOp (double x, double y);
9
10        //main()
11        static void Main(string[] args)
12        {
13            //Header Prompt
14            Console.WriteLine("\nLet's demonstrate the usefulness of Airthmetic\n" +
15                               "operations by using Lambda Expressions.\n");
16
17            //Lambda Expressions, Arithmetic Expressions
18            mathOp div = (x,y) => x / y;
19            mathOp exp = (x,y) => Math.Pow(x,y);
20            mathOp min = (x,y) => Math.Min(x,y);
21            string[] expr;
22            double expr1=0.0,expr2 = 0.0;
23
24
25            //Prompt for Division
26            Console.Write("LET'S DO SOME DIVISION:\t");
27            try{
28                expr = Console.ReadLine().Split();
29                expr1 = double.Parse(expr[0]);
30                expr2 = double.Parse(expr[1]);
31            } catch (Exception e) when (e is FormatException || e is OverflowException) {
32                Console.WriteLine("\n\nYou have entered invalid input. Restarting.\n\n");
33                Main(args);
34            }
35        }
36    }
37 }
```

```
35 //Lambda Expression, Print Result
36 Console.WriteLine("The result is:\t" + div(expr1, expr2));
37
38
39 //Prompt for Exponentiation
40 Console.Write("\nLET'S DO SOME EXPONENTIATION:\t");
41 try{
42     expr = Console.ReadLine().Split();
43     expr1 = double.Parse(expr[0]);
44     expr2 = double.Parse(expr[1]);
45 } catch (Exception e) when (e is FormatException || e is OverflowException) {
46     Console.WriteLine("\n\nYou have entered invalid input. Restarting.\n\n");
47     Main(args);
48 }
49 //Lambda Expression, Print Result
50 Console.WriteLine("The result is:\t" + exp(expr1, expr2));
51
52
53 //Prompt for Minimum
54 Console.Write("\nLET'S DO SOME MINIMUM:\t");
55 try{
56     expr = Console.ReadLine().Split();
57     expr1 = double.Parse(expr[0]);
58     expr2 = double.Parse(expr[1]);
59 } catch (Exception e) when (e is FormatException || e is OverflowException) {
60     Console.WriteLine("\n\nYou have entered invalid input. Restarting.\n\n");
61     Main(args);
62 }
63 //Lambda Expression, Print Result
64 Console.WriteLine("The result is:\t" + min(expr1, expr2) + '\n');
65
66 }
67
68 }
```

```
C#7: bash — Konsole
adrian@localhost:~/Desktop/CODE/C#7> dotnet run

Let's demonstrate the usefulness of Airthmetic
operations by using Lambda Expressions.

LET'S DO SOME DIVISION: 7 3
The result is: 2.33333333333333

LET'S DO SOME EXPONENTIATION: 6 6
The result is: 46656

LET'S DO SOME MINIMUM: 367.12763 9837.23
The result is: 367.12763

adrian@localhost:~/Desktop/CODE/C#7> 
```

C++

In C++, Lambda Expressions are **closures**, that is, they are an unnamed function object. You can use this function on the fly.

```
Welcome  lambda.java  Program.cs  lambda.py  lambda.cpp x
1  #include <iostream>
2  #include <cmath>
3  #include <algorithm>
4
5  //Lambda Expressions, Arithmetic Expressions
6  auto _div = [] (double x, double y) { return x/y; };
7  auto _exp = [] (double x, double y) { return pow (x,y); };
8  auto _min = [] (double x, double y) { return std::min(x,y); };
9  double expr1, expr2;
10
11 //Function Prototype
12 void calc();
13
14 //main()
15 int main () {
16     /* Because it is against ISO C++ to recursively call main(), we'll
17     place the code in a function. */
18     calc();
19
20     //Flush char buffer, close.
21     std::cout<<std::endl;
22     return 0;
23 }
24
25 //calc()
26 void calc () {
27     //Header Prompt
28     std::cout<< "\nLet's demonstrate the usefulness of Arithmetic\n" <<
29     "operations by using Lambda Expressions.\n\n";
30
31     //One can always escape the program by entering the EOF character (such as CTRL+D
32     //or CTRL+C) on their shell triggering an EOF Error, ending the application.
33 }
```

```
Welcome  lambda.java  Program.cs  lambda.py  lambda.cpp x
34 //Prompt for Division
35 std::cout<<"LET'S DO SOME DIVISION:\t";
36 std::cin>> expr1 >> expr2;
37 //It must be a number.
38 if (std::cin.fail() ) {
39     std::cout << "\nYou have entered invalid input. Restarting.\n\n";
40     std::cin.clear();
41     std::cin.ignore(10000,'\n');
42     calc();
43 } //Lambda Expression, Print Result
44 else { std::cout<<"The result is:\t" << _div(expr1, expr2) << '\n'; }
45
46 //Prompt for Exponentiation
47 std::cout<<"\nLET'S DO SOME EXPONENTIATION:\t";
48 std::cin>> expr1 >> expr2;
49 //It must be a number.
50 if (std::cin.fail() ) {
51     std::cout << "\nYou have entered invalid input. Restarting.\n\n";
52     std::cin.clear();
53     std::cin.ignore(10000,'\n');
54     calc();
55 } //Lambda Expression, Print Result
56 else { std::cout<<"The result is:\t" << _exp(expr1, expr2) << '\n'; }
57
58 //Prompt for Minimum
59 std::cout<<"\nLET'S FIND THE MINIMUM VALUE:\t";
60 std::cin>> expr1 >> expr2;
61 //It must be a number.
62 if (std::cin.fail() ) {
63     std::cout << "\nYou have entered invalid input. Restarting.\n\n";
64     std::cin.clear();
65     std::cin.ignore(10000,'\n');
66     calc();
67 } //Lambda Expression, Print Result
68 else { std::cout<<"The result is:\t" << _min(expr1, expr2) << '\n'; }
69 }
```



```
Lambda: bash — Konsole
adrian@localhost:~/Desktop/CODE/Lambda> clang++ lambda.cpp -o lambda -std=c++17
adrian@localhost:~/Desktop/CODE/Lambda> ./lambda

Let's demonstrate the usefulness of Arithmetic
operations by using Lambda Expressions.

LET'S DO SOME DIVISION: 7 3
The result is: 2.33333

LET'S DO SOME EXPONENTIATION: 6 6
The result is: 46656

LET'S FIND THE MINIMUM VALUE: 367.12763 9837.23
The result is: 367.128

adrian@localhost:~/Desktop/CODE/Lambda>
```

Want the source? Grab it here.

afinlay5/LambdaExpr

Gradle source code repository for C++, Java, C#,
Python source code examples posted on persona...
github.com

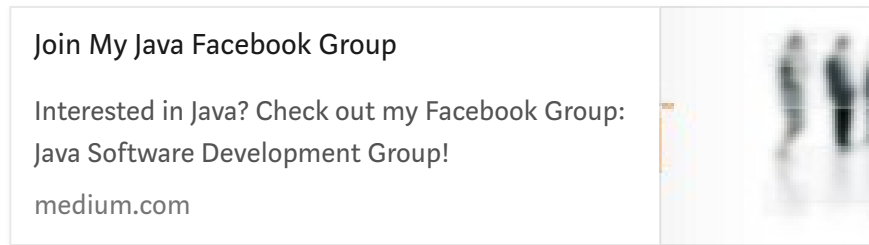


**Like Lambdas? Let me know in the
comments below :)**



Looney Tunes Ending [4]

Interested in Java? Join my Java group on Facebook:



Like my Content? Subscribe to my mailing list:

This embedded content is from a site that does not comply with the Do Not Track (DNT) setting now enabled on your browser.

Please note, if you click through and view it anyway, you may be tracked by the website hosting the embed.

[Learn More about Medium's DNT policy](#)

Don't forget to give it a.... ;)



lEmoji.com

