PC & MOBILE

Programming For Beginners (Part 2)

By **James Bruce** / October 28, 2011 28-10-2011 / 4 minutes

James Bruce
668 articles

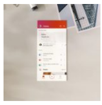
James has a BSc in Artificial Intelligence, and is CompTIA / and Network+ certified. He's t lead developer of MakeUseOf spends his free time playing V paintball and boardgames. He been building PCs since he wa kid.

Trending Now

The 8 Best Fact-Checki for Finding Unbiased T
by Megan Ellis

5 Ways Your WhatsApp Messages Can Be Hacke
by Georgina Torbet

How to Easily Share Fil Between Mac and Wind
by Dan Price



You Can Now Try Micro Office for Mobile
by Dave Parrack

13 df Linux Commands Examples
by Christian Cawley

7 Essential LinkedIn Pr to Guarantee Success
by Emma Roth

Email Facebook Pinterest
Twitter

In part 2 of our absolute beginners guide to programming, I'll be covering the basics of *functions, return values, loops and conditionals*. Make sure **you've read part 1** before tackling this, where I explained the concepts of variables and datatypes. You won't need to do any actual programming yet – this is all still theoretical and language-independent.

The Basics Of Computer Programming 101 - Variables And DataTypes

Having introduced and talked a little about Object Oriented Programming before and where its namesake comes from, I thought it's time we go through the absolute basics of programming in a non-language specific way. This...

READ MORE

Next time, we'll start putting all this into practice with some real code. If you've always promised

To quickly recap, last time I explained what variables were and some of the basic types data they can store. You should be able to explain what the following datatypes are:

Character

String

Integer

Float

Boolean

Array

We also looked at the difference between strongly-typed, and weakly-typed programming languages; and the advantages and disadvantages of each. Lastly, let's make sure you know the difference between statements of assignment, and of equality. Which of the following assigns variable B to variable A, and which tests if they have the same value?

```
a = b;

a == b;
```

Great! If you've come this far, that's an amazing achievement and you're well on your way to making your own software! Let's move on today's lesson.

Conditionals and Loops

There are three structural building blocks of a program that you need to know, and they are

by Mihir Patkar

Philo vs. Sling TV: What Best Streaming Service

by Dan Price

Latest Giveaways!

Elegoo Mars: High Qual Printing for Anyone

by Joe Coburn

Anycubic Photon S: The Resin 3D Printer? (And Than $500)

by Joe Coburn

One Mix 3S Yoga: Not tl Netbook, But Probably Best

by Kris Wouk

Vankyo V600: Budget P Projector That's Brilliar Bright

by Joe Coburn

Latest Videos

Tech History Timeline Tri Game: Catch the Joker |

IF is one-time test. “*IF a is true, then do this*”. IF *a* isn’t true, the program ignores whatever comes next and carries on with the rest of the code. You can also provide an alternative with ELSE. “*If a is true, then do this, ELSE do this*”. It allows you to make decisions in the program depending on a variable. Remember the test for equality that we learnt about earlier?

That could be used in an IF clause, for example:

```
if(a==b)
print “a is equal to b”
else
print “a is not equal to b”
```

FOR is a way to loop over the same code a set number of times. There is no testing involved – it just repeats the same block of code however many times we tell it to.

WHILE is also a loop, but instead of performing the options a pre-determined number of times, it performs a test each time the loop is performed and continues to loop until the test fails. If the test continues to be true, it never stops looping. This can cause problems if you have a bug in your WHILE look, leading to unresponsive programs that crash in an endless loop.

How to Download YouTu
Videos to Your iPhone Ca

Elegoo Mars: Perfect f
Beginners to Resin 3D Prir

Anycubic Photon S: Incredi
Prints from a Sub $500

10 Sites to Download Roy
Free Christmas Musi

These 3 basic structures create the traffic lights and road diversions that control the logical flow around an application.

Functions

A function is just a group of code that's been given a name. By grouping a block of code together and naming it, we can re-use it later and throughout the application without having to rewrite the whole code block again. Not only does it save time and reduces the overall code size of an application, it also means that if there was something wrong, we would only need to change it in one place.

Nearly all programming languages come with a built in set of functions that you can use in your application. For instance, **ECHO** is a function found in many languages that displays some text on screen. There are usually also some functions to deal with reading and writing files, graphical or audible output, keyboard and mouse input, and mathematics. You can use all these built-in functions without knowing how they actually work – all you need to know is the correct function name, and the parameters it requires. Wait.. *parameters*?

Affiliate Disclosure: By buying the proc recommend, you help keep the lights on MakeUseOf. .

We often refer to functions as “*taking*” certain variables and “*returning*” something back to us. It might help to consider a function as a machine you can’t necessarily see inside of. You can put things into it, and you might get something out again, but you don’t actually have to know what goes on inside the machine. The things you put in are the parameters – variables of data that the function will work with. The return value is the output of the function – the data that will be given back to you once it’s finished running through it’s logic.

beginner programming

In the example above, the “signature” of the function is one that accepts two variables (**a** and **b**), and returns one (**c**).

Sometimes functions won’t return any values at all, though it’s common practice in those cases just to return a Boolean value of true or false to show whether it was successful or not. If you called a function to save a file for example, you wouldn’t necessarily want any variables back, but you would like to know if the file was written correctly or if there was an error.

In all applications, variables and data are constantly being “passed around” through thousands of different functions, each of which performs it own distinct purpose in the big scheme of things.

Check What You Learnt Today

To recap today’s points, you should understand what the following do in any programming

- IF
- FOR
- WHILE

You should also understand what a function is, and what I mean when I say "*this function accepts a string and returns a Boolean*".

That's all for this lesson. Next time, we'll try to put some of this knowledge into practice using real code to write a small application, though I haven't decided which language to tackle first. If you have any requests, then perhaps you could post in the comments. What programming languages do you think are the most important today?

Image Credits: **ShutterStock 1**,**2**

17 COMMENTS WRITE A COMMENT

PC & MOBILE

How to Fix It

Controller Project Tutorial

According to Science

or Coding Course!

Programmers and Developers

Using VBA Scripts

Exist Without Functions

Does It Work?