Hello Boo - a beginner tutorial

"Hello, Boo!"

This tutorial assumes you have basic computing knowledge and are comfortable with the command line interface of your operating system. If you're familiar with programming in general, you might just want to browse the code snippets and ignore the rest of the article in general.

*Things required:
*Boo.
*A .NET runtime - either Microsoft's .NET Runtime on Windows, or Mono on Linux.
*Your favorite text editor.
*Being comfortable with the command-line, for now.
*Free time.

Before we even get started, its time to show you the obligatory "Hello, world!" application. Every language tutorial has them, and Boo is no exception! Everybody's gotta start somewhere.

Crack open Your Favorite Text Editor.
On the very first line, type,

```
print "Hello, world!"
```

Save the file as hello.boo, and make a note of the directory its been saved in.
Go to the command prompt and find the directory you installed Boo into, and go to the bin subdirectory. /boo/bin, if you will!

Now, type,

```
booi path_to_directory_where_hello_boo_is\hello.boo
```

You'll see "Hello, world!" printed on screen. Welcome to Boo.

"What is this 'print' thing?"

Print is an expression in Boo that is used to feed output to a device called "Standard Output."
Ignoring the concept of "Standard Output" completely, what it means (for now) is that the "print" expression will just show text in the console, like you just saw with the hello world program.

"Hello, $name"

What if we want to get really specific - what if we want to print out someone else's name instead of just saying hello to the entire freaking planet, eh? We could, of course,

```
print "Hello, Bob!"
print "Hello, Sally!"
...
```

but everytime we wanted to say hello to someone new, we would be in quite a quandary!

But, never fear, for your hero is here, and he will now show you how to read input from the user - don't worry, its really easy and you don't have to worry about it. Crack open hello.boo and replace its contents with,

```
import System
name = Console.ReadLine()
print "Hello, $name"
```

and save the file.

We'll break it down line-by-line in a second, but for now, run "booi" just like you did before, and stare in awe: nothing's happening! All there is is a blinking cursor! Type your name and press enter. I typed "Bongo," because I'm a freak.

```
Hello, Bongo
```

Neat, eh, but what happened?

```
import System
```

System is a namespace - its like a box with lots of delicious snacktreats in it, or, if you're on a diet, like a box full of slightly stale protein bars. The "Console" class is one of these delicious treats, just waiting to be plucked from the box. We could have accessed the "Console" class by using "System.Console," but we didn't - why?
Using the "import" keyword is a way of saying, "dump all the contents of the System namespace into my file so I don't have to keep typing the namespace, 'System,' before everything." Why would you do this? Because you're lazy, that's why.

```
name = Console.ReadLine()
```

Here you are doing two things - you are calling a member of the "Console" class, called "ReadLine()", and storing a value it returns into "name." ReadLine() is a method that waits for the user to type something and press enter, and returns a **string** of characters. This string goes into the "name" object. Thus, were the user - an upstanding citizen such as yourself - to type "Bongo," then "name" would now have the contents "Bongo" after the user pressed the enter key.

```
print "Hello, $name"
```

This is the easiest part of the program - its called "String Interpolation" The curly brace symbols essentially mean, 'embed this object inside of this string,' so when you write "$name" you are really saying, "replace $name with the contents of name." Since we typed in "Bongo" earlier and stored that in the name variable, instead of seeing "Hello, $name" printed on the screen we will instead see, "Hello, Bongo." Take special note: using $<object> actually calls a special member that every object has, called, 'ToString()' - this member returns a string that represents a formatted description of the object. Not all classes implement their own custom ToString() member, so you might see something strange like 'System.DateTime' instead of the actual date and time.

Exercises:
*Create a program that reads in the user's name and prints outs something like, "Your name is $name. Hello, $name!" except that $name is replaced with the user's name.
*Create a program that reads in the user's first name, and then the user's last name, and print them together, like, "Your name is $firstname $lastname." You'll need at least two variables, and if you just read the last sentence, you'll probably have an inkling of how to do it. Re-examine the

```
import System
name = Console.ReadLine()
print "Hello, $name"
```

program if you are feeling lost.

*Tip: There are many more classes available in the System namespace - go to Microsoft .NET Class guide and check out the namespaces available - there are tons of classes inside! Remember to use "import" or else you'll be typing System.Console" all year long.