Tutorial 3 : If

In the first tutorial I mentioned that variables are foundational to computer programming. However, there is another very important statement which goes along with variables. This statement is simply, `if`.

Computer programming is a continual stream of if's. You continually check things like **if** the user is pressing a given key, **if** a car has collided with the wall, **if** you still have any lives left, etc.

It's an easy command to remember and use since it means the same in DarkBASIC as it does in English. Here is an example:

```
if 5>2 then print "Yes, this works"
wait 3000
```

As you probably guessed from the example, the ">" character means "greater than" just like in math and conversely, "<" means "less than". There's also another new word I used – the `then` statement. This is how you get the program to react to a condition. You say, if some condition is true (in this case, if it's true that 5 is greater than 2) **then** do something (in this case print the given sentence).

What if you want to do more than one thing for a given condition? To do this, you can use another version of the if statement:

```
if 5>2
      wait 50
      print "Hi Everybody"
      wait 1000
      print "See, this works!"
endif
wait 3000
```

Here's how the above code works. The computer looks at the first line (`if 5>2`) and determines if 5 really is greater than 2. If it isn't, then it will skip to the statement immediately following the endif statement (in this case, `wait 3000`). If it is true, though, then it will just keep on executing line by line.

Notice also how I indented everything between the `if` and the `endif`. This is not required to get your program to work but it does make it easier to read. I indent my code between `if` and `endif` statements since it makes it easier for me to see what code is inside the condition and what code is outside it. So, you don't have to do it in order to get it to work, but it is a common programming style that will make things easier when you're trying to debug your code or if someone else is trying to add on to the code you've written.

The above example wasn't a meaningful use of an if statement since it's clear beforehand what it would do and I could have done the same thing more simply by just removing the `if` statement altogether and simply telling it to print what I wanted. However, here is an example where it is very useful:

```
input "Is the earth round? (y or n) : ", answer$
if answer$="y" then print "That is correct!"
if answer$="n" then print "No, that is incorrect."
wait 3000
```

Before you even run the code, try to see what it will do. The first line will print out the question and let the user type in their answer. The second line will check to see if what they typed in (which was stored in the string variable `answer$`) is equal to the string "y" and, if so, it will print out a congratulatory message. However, if they didn't type y, that line won't do anything and it'll simply go on to the next line which checks to see if they answered "n". If so, it prints out a different message. Then, regardless of what they answered, it waits 3 seconds before exiting.

Now you can go ahead and run it. Do you see any problem with this code? A good question to ask yourself when trying to find problems with your code is, "What if?" What if somebody types a "t"? It won't print anything. Is that what you want it to do? For the purpose of this example, let's say that if someone types in anything other than the right answer, we want it to tell them they are incorrect. How could we do that?

One way would be to have an `if` statement for every single character on the keyboard.

```
if answer$="q" …
if answer$="w" …
if answer$="e" …
…
```

However, this would take awhile. It would be better if we could just say that if it was correct it should print out one thing and otherwise it should print out a different thing? You can do this with the `else` statement:

```
input "Is the earth round? (y or n) : ", answer$
if answer$="y"
     print "That is correct!"
else
     print "No, that is incorrect."
endif
wait 3000
```

So again, I prompt the user to type their answer to the question and store that answer in the string variable `answer$`. Then I check to see if that variable equals the string "y". If it does, then the computer executes everything between the `if` and the `else`. However, if `answer$` equals anything other than "y", it will execute the code between the `else` and the `endif`. This is exactly what I wanted it to do.

But now, what if someone types "Y"? It says they are incorrect!

What I really want to check in the second line of the code is if answer$ equals small y or capital Y. There's another statement in DarkBASIC that let's me do this. Simply replace the old `if` statement with this new one:

```
if answer$="y" or answer$="Y"
```

The `or` statement let's you check more than one condition. It's important that you restate the whole condition, `answer$="Y"`, after the or. Otherwise, it will give you an error. Also, it's important not to put another `if` after the or. So now, if

someone types in a small or capital "y", it will tell them they were correct. Otherwise, it will say they are incorrect.

Syntactically, these commands are easy. **`If`**, **`endif`**, **`else`**, and **`or`**. However, the concepts of how to use them might be somewhat tricky unless you actually use them yourself and see how they work. Here's a project where you can put these skills to use:

Write a simple "quiz" program with True/False, Multiple Choice, and basic Math questions. Here are 3 sample questions to get you started:

"George Washington was the first president of the United States. (t or f) : "
"A sparrow belongs to what group? (a) Fish, (b) Birds, (c) Insects : "
"What is the sum of 5 and 3? : "

These questions can be easily handled with either a few if statements or an if-else statement. You may also want to use the `or` command to handle capital letters or other common answer variations.

Remember that even though the user will enter a number for the math problems, it will still be a string and thus you'll still need to treat their answer as a string when checking it with the `if` statement (so `if answer$=8` will not work while if `answer$="8"` will).

As always, if you have any questions or problems doing this, please feel free to send me an e-mail (mail@treksoftware.org).