Tutorial 2 : Strings

In the last tutorial, I mentioned I would explain why the text had to be put in quotes. To begin explaining it, I'll start with another example. Try running the following program and see what happens:

```
print "Software"
wait 3000
```

If you just ran it you probably weren't surprised by the output. It printed out Software in the upper-left corner just like you would think But now write the same code without the quotation marks and see what happens:

```
print Software
wait 3000
```

It printed out a 0! Why is that? Well, since you hadn't put Software in quotes, it assumed it was a variable like I covered in the previous tutorial and since DarkBASIC automatically sets variables to 0, it simply printed out the value of the variable Software.

The text in quotes, "Software" is called a *string*. For practical purposes, a string is a series of characters. "Happy Birthday!!" is a string as well as "100392314901" as well as "(!#^&*@$". It doesn't matter what characters they are. The thing that's important to know about strings is how they are treated differently from numbers.

The first difference you already saw. We can print out numbers by themselves (e.g. `print 25`) but strings we have to put in quotes.

Another difference is what happens when you add them together. When we do something like `print 5+10` we'll see 15 on the screen. But if you were to do `print "5" + "10"` you would see 510 on the screen. The reason for this is simple. When

you add strings together (for example “Happy” and “Birthday”) it wouldn’t make sense to add them like numbers. Rather, what makes sense to do is combine them. So the + operator will always combine (also called *concatenate*) strings.

In the last tutorial, I talked about how you can store numbers in variables so you might wonder if you can also do that with strings. For example, could you just say, `variable = “Hello World”`? The answer to the first question is yes but this is the wrong way to do it. The example above would give you an error if you tried to run it but there is a way to store strings in variables. Simply add a $ on the end of the variable name:

```
variable$ = “Hi, how are you doing?”
print variable$
wait 3000
```

String variables can be used for a number of different kinds of applications. One example would be a MadLibs program. MadLibs is a game where the player is instructed to choose a certain kind of word (be it a noun, a name, a fruit, etc) and then all the words he enters are put into a sentence that is often funny to read since it usually doesn’t make sense.

Before you can do that, though, there's one more command that you should know about. It’s called the `input` command. Basically, it allows the user to type in strings while the program is running. Here’s how it works:

```
input “Please enter your name : ”, name$
print “Hi ” + name$ + “, how are you doing today?”
wait 3000
```

See what the program does? The input line prints out the text, “Please enter your name : ” so that the user knows what to type. They then type whatever they want and hit enter. The program then saves whatever they typed as the string variable `name$`. We can then use this in our program - in this case, making a personalized greeting.

Using this information, you can make your own MadLibs program. Here are a few hints:

- Use a series of input commands to store the words you need for the sentence.
- Use different string variables to store each word.
- Construct the sentence using plain text ("One day") with variables added in between the text with each part connected with the + operator.
- To get spaces between words, simply add extra space at the ends and beginnings of strings on either side of variables, or add a string with simply a space (i.e. " ") between two variables.

If you have any problems, feel free to e-mail me at mail@treksoftware.org.