

MM 319 Lab One: Lingo Fundamentals

Introduction

This activity will help you better understand basic Lingo syntax. Later activities will extend your knowledge of Lingo.

Data in Lingo Scripts

The data types available in Lingo are similar in many ways to those you know from other programming environments. There are integers, floats, Boolean values and strings. There are other types as well, such as point, rect (the coordinates describing a rectangle) and color. Lingo supports lists, which we'll come back to later, and has operations for working with strings in many useful ways.

As in “regular” programming, we may work with literal data or variables. Literal numeric values are represented by using the number without any quote marks, such as **27** or **3.14**. Literal strings are represented by enclosing the value in quotes, as in **“Hello World”** and **“How are you 2-day?”** Finally, Boolean literals are **TRUE** and **FALSE**.

Variables are represented by an identifier, which is given its value using an assignment statement. Identifiers may not begin with a numeral, and they can't include symbols that represent arithmetic. Here are some identifiers being initialized by assignment:

```
maxWeight = 400  
daysInMonth = 29  
defaultName = “John Doe”
```

One unusual feature of Lingo variables is that their type is not fixed. You may change the type of a Lingo variable by assigning it a new value. Note that this is not a great idea and very likely to make your programming harder to understand.

Basic Operations

The most fundamental operation in programming is assignment. In Lingo, as many other languages, the assignment operator is the equals sign (=), and an assignment statement consists of a variable name, the assignment operator, and an expression that evaluates to a type. The preceding section shows three simple examples.

Expressions may be more than a single value. An expression may have several values (either literal or variable) combined by other operators in a way that makes mathematical sense. The basic arithmetic operators are the same for Lingo as for other modern programming languages: **+**, **-**, *****, and **/**. To perform the modulo operation, use **mod**. Parentheses may be used to force an operation to be evaluated before it normally would. Note that except for the mod operator, all of this is just the same as in Python!

There are also comparison operators. An expression that includes a comparison operator does not yield a numeric result, but instead gives a Boolean (true/false) answer. Lingo's comparison operators are *mostly* the same as those used in Python and Java: **<**, **>**, **>=**, **<=**. One difference is the comparison operator for equality, which in Python and Java is two equals signs (**=**). Lingo uses only one equals sign here—two will cause a syntax error.

One last category of operations is input and output. For the moment, let's skip input and just use a very basic output command: **put**. This command moves the value of some variable (container) to a new location. If no location is given, the value appears in the message window.

Hands On

Launch Director® and create a new file. Open the message window, either by entering the shortcut Command-M or by selecting the message window from the **Window** menu. (The Command key is the one next to the keyboard with an apple icon on it).

You will notice that the message window is divided into two parts. The upper part is a space for you to enter individual statements; the lower window shows the result of **put** operations. Click in the upper part and type

```
put "Hello World"
```

The phrase "Hello World" should appear in the lower part of the window. The message window serves a purpose much like python's interactive environment. The remainder of this part of the activity is a series of statements to try in the message window. Use these as a model for further exploration.

```
myNumber = 57
put myNumber
put myNumber / 3
put Mynumber
```

Note that Lingo identifiers are **not** case sensitive. This can cause a lot of woe if you are careless.

```
myNumber = "This is a number?"
put myNumber
x = 20
y = 30
z = 5
w = y - x / z
put w
w = (y - x ) / z
put w
s = "Hello"
s2 = "World"
```

Watch the next few carefully—they illustrate concatenation (and how *not* to do it).

```
put s + s2
put s & s2
put s && s2
```

```
s = "3.14"  
put s + 2  
s = "Fred"  
put s / 2
```

Notice that Lingo will convert string value to the number it represents if it is used as a number, but of course if the string *doesn't* represent a number, the results will be just plain silly.

Interacting with Stage Elements

Once you've gotten acquainted with the message window and basic operations, it's time to try interacting with a text box in the screen. Add a text cast member, and edit its properties to make its **framing** property is **fixed** (and then size the box to suit yourself). Name the cast member "Result" in the cast member property inspector.

Now return to the message window and enter this statement:

```
put "Hello World" into member("Result")
```

Like variables, text members are a type of container that we can put values into. A more conventional way of doing this is with a simple assignment statement:

```
member("Result").text = "Goodbye, cruel world"
```

We can also use a text member as an avenue for input:

```
s3 = member("Result").text  
put s3
```

When referring to members as if they are variables, we can identify them in several ways, but using the name (and choosing a good descriptive name) results in much clearer code.

Exploration

You can use the Lingo reference that is available in the message window (and also scripting window) to gain information on other programming structures. Unfortunately, multiple-line commands are tricky to test in the message window, and all but the most simple structures require multiple lines. Your last task for this lab will help you develop some learning strategies as well as preparing you for the next step in the class.

First, in the message window explore the small toolbar at the top of the window, next to the word "Lingo." The first two of these menus are "Alphabetized Lingo" and "Categorized Lingo." Both of these list all the available Lingo commands (and, alas, some that don't exist). If you make a choice from one of these menus, command with its basic syntax will appear in the window. Look for the commands for implementing selection (**if**) and iteration (**repeat**).

Next, open Director Help, and enter the phrase "message window" as the key phrase. From the "hits" you will (with a little looking) find out how to test multiple line statements in the message window. Have fun!