3 Syntax

- **Introduction**
- **Syntax**
  - Statements
    - Colon :
    - Line Continuation _
    - Conditions
- **If … Then … Else … End If**
  - 1. block form syntax
  - 2. One-Line syntax
- **Select Case … Case … Case Else … End Select**
- **Do...Loop**
- **For...Next**
- **While...Wend**
- **Objects**
  - Properties
  - Methods
  - Containers
  - Getting Help
  - Object Browser
- **Reference Libraries**
  - VBA Library
  - Other Reference Libraries
  - Change or check references

Objective: Become familiar with VBA syntax. Understand objects, properties, and methods.

Introduction

When you are going to build a house, you would not use glass for walls, nor would you use bricks for windows. The *type* of material is a consideration when you construct.

Chapter 2, Defining Variables, will be a great reference for you as you begin to build your code.

The best way to learn VBA (on your own) is to start with Excel -- one reason is that there is a built-in macro recorder and you can examine the code that is created. The code that the macro recorder writes is often clunky, so knowledge of streamlining things helps out tremendously -- like, in Excel, use the array style of referencing to cells

cells(4,2)

is the cell in row 4, column 2; commonly known as B4.

Using numbers to reference cells instead of the more familiar Letter-Number (i.e.: A1) style of referencing allows you to define variables and loop through cells more easily.

Syntax

This chapter is going to discuss *syntax*, the structure of the word order in a line of code.

Code lines are composed of **Statements**.

Statements

A **Statement** can declare, define, calculate, test, assign, label, do …. Statements must be syntactically complete and correct.

A **Statement** generally occupies a single line.

Colon :

If you want to put more than one statement on a line, you can use a colon (**:**).

Line Continuation _

If you want to break a statement onto two or more lines, you can use the line-continuation symbol, the underscore (_) character.

Conditions

Life is full of choices. If one thing happens, it will cause another to happen. Simply put: If this, then that.

A **condition** is a numeric or string expression that evaluates to **TRUE** or **FALSE**.

When programming, a primary statement you can use to test a condition and decide what will happen is the **IF** statement.

If … Then … Else … End If

The **IF** statement conditionally executes a group of statements depending on the value of a condition

There are two forms of the **IF** statement syntax.

1. block form syntax

```
If condition Then
    [statements]
[ElseIf condition-n Then
     [elseifstatements]
[Else
    [elsestatements]]
End If
```

When presenting syntax, anything that is optional is enclosed in square brackets. Therefore, the only required lines for this statement are:

```
If condition Then
End If
```

Of course, it would be silly to test a condition if the outcome did not affect any course of action.

If you don't have much to do if the condition is met, you can use a shortened form of the IF statement

2. One-Line syntax

```
If condition Then [statements] [Else elsestatements]
```

Select Case … Case … Case Else … End Select

The Select Case Statement executes one of several groups of statements, depending on the value of an expression.

```
Select Case testexpression
  [Case expressionlist-n
    [statements-n]] ...
  [Case Else
    [elsestatements]]
End Select
```

Do...Loop

A Do statement repeats a block of statements while a condition is True or until a condition becomes True.

```
Do [{While | Until} condition]
   [statements]
   [Exit Do]
   [statements]
Loop
```

Alternatively, you can use this syntax:.

```
Do
   [statements]
   [Exit Do]
   [statements]
Loop [{While | Until} condition]
```

For...Next

A For statement repeats a group of statements a specified number of times.

```
For counter = start To end [Step step]
   [statements]
   [Exit For]
   [statements]
Next [counter]
```

While...Wend

A While statement executes a series of statements as long as a given condition is True.

```
While condition
    [statements]
Wend
```

Objects

Objects have **Properties** and **Methods**.

In VBA, you manipulate objects -- their properties -- and launch their methods. Therefore, becoming familiar with objects will make you a better programmer.

In Excel, you have workbooks and sheets, cells, ranges, graphs, etc -- as your objects

In Access, you have tables and fields, queries, forms, reports, controls, etc -- as your objects

Properties

Properties are descriptive terms for an object. For example, a human object has properties like:

- hair color
- eye color
- height
- weight

A cell (range) in Excel has properties such as:

- row
- column
- background color
- foreground color
- format
- value

Methods

Methods are things that an object can do. For example, with a human object, methods would be:

- eat
- sleep
- run
- jump

A cell (range) in Excel has methods such as:

- activate
- add comment
- check spelling
- clear contents
- copy

Containers

Objects are contained in containers -- like a container with recordsets (in Access) or worksheets (in Excel). Among others, containers have a property called Count, which is how many objects it contains. Individual properties of an item in a container or the container itself can be determined (and often set), and methods can be launched.

Getting Help

... you can press Shift F1 to turn your mouse into a questioning pointer -- then click on a tool or menu option to get context-sensitive help

... you can press F1 in a property to get help for that property (like on the property sheet in the design view of a form or Excel/Access or report in Access)

... you can press F1 on a keyword in a module sheet to get help for that word

Object Browser

Press F2 in a module window the View the Object Browser... a GREAT place to get better knowledge of objects ;)

The Object Browser is used to lookup objects, properties, and methods for objects in loaded reference libraries.

Reference Libraries

Reference Libraries contain information about objects. For instance, if you want to use Automation (formerly OLE Automation), you can set a reference to an application's type library.

There are standard reference libraries that are always loaded.

VBA Library

The VBA Library contains fundamental functions (methods) and properties.

From the Object Browser window, click on the Project/Library drop-down and choose VBA, instead of, for instance, <all libraries>.

Look at the classes (categories) of functions/properties -- click on a class and then click on an item in the right (members) pane.

To get the help for the item that you have selected, press F1.

Note on Library References:
Order as well as presence is important.

Other Reference Libraries

If you want information about objects specific to Excel -- or Sheets, Workbooks, Ranges -- change the library to Excel.

If you want information about objects specific to Access -- like forms, reports, controls, etc -- change the library to Access.

Change or check references

To check or change reference, from the menu bar, choose **Tools, References...** from a module window

Explore the different libraries you have to loaded and see what is available in each.

Check off a reference library that looks interesting to you and explore what is in it. Don't forget to uncheck unnecessary optional libraries when you are done browsing them, as loading libraries consumes resources.