



- **Creating a simple program**
  - VBA Window
  - Project Window
  - Properties Window
  - Insert a New Module
  - Programming a Message
  - MsgBox Function
  - Debugging
  - Correcting code
  - Typing in lowercase
  - Switching between Code and Spreadsheet Window
  - Running a procedure
- **Modifying code**
- **Testing a Return Value – Using Parenthesis**
  - Built-in Constants
- **Getting Help**
  - Getting help in a spreadsheet
- **Getting User Input**
  - InputBox Function
  - Literal Strings

**Objective: Become familiar with the Visual Basic window; create and run simple programs; get help.**

## Creating a simple program

Start Excel. When you are on a blank worksheet,

press   to go to the module window.

### VBA Window



The Visual Basic Window is organized as follows:

TitleBar – WorkbookName – [ProcedureName (Code)]	
MenuBar and Toolbars	
<b>VBA Project</b> VBAProject (ProjectName) Microsoft Excel Objects Sheet1 Sheet2 Sheet3 This Workbook Modules Module1	<b>Code</b>  Sub SubName() codeline1 codeline2 End Sub  Function FunctionName() codeline1 codeline2 End Function  Procedures can be <u>S</u> ubs or <u>F</u> unctions. Functions execute code and return values, like the SUM function. Subs execute code without returning a value.
<b>Properties</b> Module1 Module (Name) Module1	

### Project Window

If your project window is not showing, choose


**View, Project Explorer**

from the menu or press  

## **Properties Window**

If your properties window is not showing, choose

### **View, Properties Window**

from the menu or press 

Properties describe an object. For instance, if our object is a person, properties you may describe could include the following:

- eye color
- hair color
- height
- weight

## **Insert a New Module**


Code is stored in "modules". There can be several procedures on each module sheet.

### *Definition: Module*

*A set of declarations followed by procedures*

Procedures can be one of 2 types:

- **Sub** procedures execute code
- **Function** procedures execute code and return a value.

In the Project Window,  your mouse on "This Workbook" and choose

### **Insert > Module**

from the shortcut menu.

On the right, a blank code window will appear.

## ***Programming a Message***

Lets create a simple program to issue a message to the user (you!).

Type:

```
Sub HelloWorld()  
    MsgBox "Hello World"  
End Sub
```

The first line tells Excel that you are creating a Sub procedure called **HelloWorld**.

The second line invokes the **MsgBox** (Message Box) function.

## ***MsgBox Function***

---

**MsgBox(prompt[, buttons] [, title] [, helpfile, context])**

---

The function name is **MsgBox**.

Arguments for the function are enclosed in parenthesis if a return value will be handled. Parenthesis are not used if the return value of the function will not be tested.

Arguments that are optional are enclosed in square brackets. The only argument that is necessary for the message box function to work is the prompt, which is enclosed in double quote marks if it is a literal string

In our example, the prompt is "Hello World".

If you want to issue a message, a common form of the MsgBox function is

---

**MsgBox prompt, , title**

---

MsgBox is actually a Function, which, of course, means that it returns a value – which button was pressed. In our example, you are only displaying the default button, which is the OK button and because it is the only one the user can choose, you do not need to test to see what was pressed.

In this case, since you do not need to know the return value, the argument(s) is not enclosed in parenthesis when you call the function.

If you want to issue a message and also test what key was pressed by the user, a common form of the MsgBox function is:

---

**MsgBox(prompt, buttons, title)**

---

### ***Debugging***

You can find syntax errors in your code by running Excel's debugger.

From the menu, choose:

#### **Debug, Compile (VBA Project)**

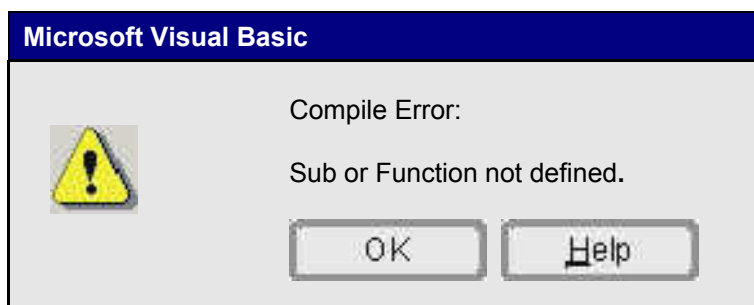
If there are no errors, nothing happens and there are no messages – this is good!

Lets modify our code to *make an error* to see what the debugger will tell us. Change your code to the following:

```
'THIS SUB CONTAINS AN ERROR  
Sub HelloWorld()  
    MBOX "Hello World"  
End Sub
```

Run the debugger (**Debug, Compile** from the menu)

When you do, you will see an error message. Excel will highlight "**MBOX**" and you will see the following message box:



### ***Correcting code***

Change **MBOX** to **msgbox**

and run the debugger again.


### ***Typing in lowercase***

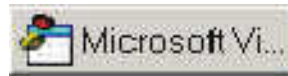
When Excel recognizes a keyword, it changes the case to mixed case. If you move off the line and your case on keywords is not changed, this means that there is an error, usually a typo.

This lowercase conversion to mixed case only applies to keywords, not names that YOU create such as the name of the Sub or Function.

### ***Switching between Code and Spreadsheet Window***

Now that you have created a program, let's switch back to our Excel Worksheet and run it.

On the row with the Windows  button, you will see rectangles for each task.



Visual Basic Window



Excel Worksheet

1. Click on the Excel task
2. Save your file. From the menu, choose **File, Save As...**

Name → **Vbclass-01**

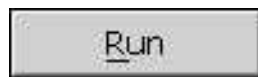
### ***Running a procedure***

From the menu, choose

**Tools, Macro, Macros...**

or press   for a list of macros.

Highlight **HelloWorld** from the list and choose



Make a note of what is inside the dialog box as well as the titlebar.

## Modifying code

When you ran your sub, the titlebar of the message box said "Microsoft Excel" since nothing else was specified. You can create a custom title.

Switch to the code window and change your routine as follows:

```
Sub HelloWorld()  
  
    MsgBox "Hello World",, "My first program"  
  
End Sub
```

The second argument, buttons, do not need to be set if you just want to show the default OK button. Because you are skipping this optional argument and accepting the default, there is simply a comma to act as a placeholder.

Whenever you modify code, always compile before running it. Choose

### **Debug, Compile**

from the menu

Switch back to Excel, Save, and run **HelloWorld** again to see the changed Title bar.

## Testing a Return Value – Using Parenthesis

You can specify which button(s) you want to appear on a message box as well as test to see what button was chosen. In this case, you will need to enclose the arguments to the MsgBox function in parenthesis since you are going to handle the return value.

Switch to your code window and type this Sub below your other code. Skip a line between Subs to make them easier to read.

```
Sub WhatButtonDidIPush()  
  
    If MsgBox("Click a button", vbYesNo, "Test Button") = vbYes Then  
        MsgBox "You clicked YES", , "Button"  
    Else  
        MsgBox "You clicked NO", , "Button"  
    End If  
End Sub
```

## Built-in Constants

To make it easier to specify numeric parameters, Excel has a number of "Constants" built-in. A "Constant" is a value that does not change.

**vbYesNo** is a **Constant** whose actual numeric value is 4.

**vbYes** is a **Constant** whose actual numeric value is 6.

## Getting Help

Getting help on a particular command is easy.

From the code window,  
click your mouse on top of the word **MsgBox** and press

**F1**

The built-in help for whatever command your cursor is on will appear.

### *Getting help in a spreadsheet*

In addition to the **F1** button, you can also get help on toolbar icons and menu choices.

Press **Shift F1** to turn your mouse into a pointer with a question mark. Then click on any icon or any menu option to get specific help.

Try it!

## Getting User Input

Often, you want to be able to use a value in your code that is supplied by the user, so you need to be able to prompt the user for a value.

### *InputBox Function*

The InputBox function displays a prompt in a dialog box, waits for the user to input text or click a button, and returns a string containing the contents of the text box.

---

**InputBox(prompt[, title] [, default] [, xpos] [, ypos] [, helpfile, context])**

---

A common form of the InputBox function is

---

**InputBox(prompt, title)**

---

Create a procedure that collects a name from the user using **InputBox**, then use **MsgBox** to display the name.

```
Sub GetUserName()  
  
    Dim mName As String  
  
    mName = InputBox("Enter your name", "User Name")  
  
    MsgBox "Your name is " & mName, , "Hello"  
  
End Sub
```

## *Literal Strings*

Literal strings are enclosed in quote marks.

In this example, you are concatenating a literal string ("Your name is ") with a variable (mName). Concatenate means to combine.

---

**& is called a concatenation operator**

---

As a good habit, always type a space before and after any operator.

*This page left blank intentionally.*