VBA Made Easy

The VBA Editor

01

www.accessallinone.com

This guide was prepared for AccessAllInOne.com by: Robert Austin This is one of a series of guides pertaining to the use of Microsoft Access. © AXLSolutions 2012 All rights reserved. No part of this work may be reproduced in any form, or by any means, without permission in writing.

Contents

Introduction3
The VBA Editor through the Ribbon4
VBA Editor through the Form Designer4
VBA Editor through the Form Designer Properties Window5
VBA Code not Working - Activating VBA Code5
The VBA Editor Explained6
Code Window6
Project Explorer Tree
Properties Window8
Immediate Window8
Watch Window9
How to select different Forms and Reports (from project explorer)9
How to select different Modules
How to Rename Modules11
Basic Tools for Writing Code
Line Continuation Character12
Indenting Code13
Editor Format to Adjust Colours
Naming Conventions
Select Object Dropdown and Procedures Dropdown14
Procedural View and Full Module View
DoCmd Syntax and Arguments Explained
How to Convert a Standalone Macro
How to Convert a Form's Embedded Macros
When to use Macros and VBA19
Questions21
Answers25

Introduction

The VBA Editor is what we use to enter VBA code for forms, reports, custom functions and more. In fact, all Microsoft Office applications (Word, Powerpoint, Excel, Outlook) use the same VBA Editor, so although we are learning to program with Access you will also be able to transfer ALL your skills and knowledge and even automate between applications.

Note: In this unit you will be seeing many examples of code in order to demonstrate some of the features of the VBA Editor. You are not expected to understand everything and some of it may even seem particularly complicated. But rest assured! We will be covering everything in detail throughout this series.

The first part of the unit will involve getting to grips with the VBA editor and understanding the functions of the various windows

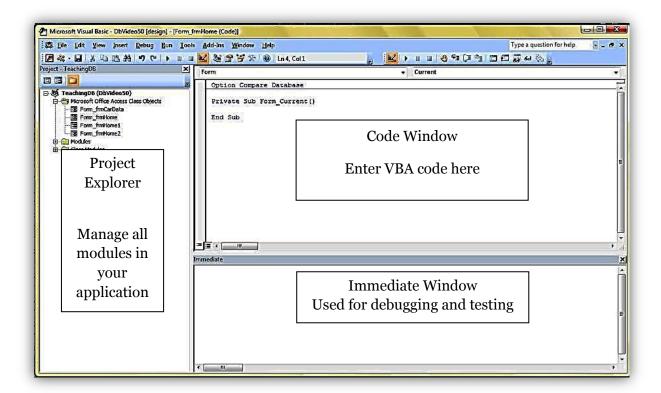


Figure 1.1

Figure 1.1 is the VBA Editor with three areas highlighted; the Project Explorer, Code Window and Immediate Window. This is what is known as an Integrated Development Environment, which means everything you need to write programs and code are all in this one window.

There are a couple ways to open the editor, all of which are quite natural once you learn them.

The VBA Editor through the Ribbon

From the ribbon select Create tab and to the far right is the Macro drop down; select Module. Now you carry out the same sequence of commands and open the VBA Editor.

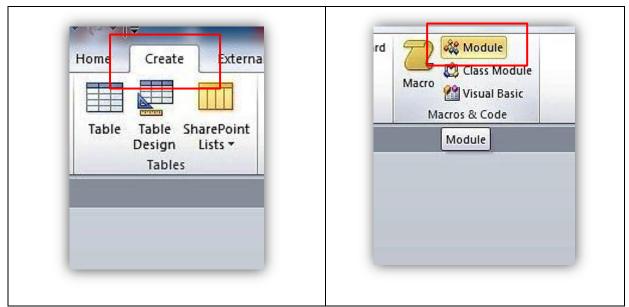


Figure 1.2

VBA Editor through the Form Designer

When in the form designer you can click the VBA Editor Button under Tools to bring up the IDE.



Figure 1.3

VBA Editor through the Form Designer Properties Window

Or, if you open the property window (F4) and click the Events tab, any of the ellipses (...) will open the VBA editor. The form must be in design view, however.

We suggest you do this as an exercise now. It will probably be the most common way you open the VBA editor.

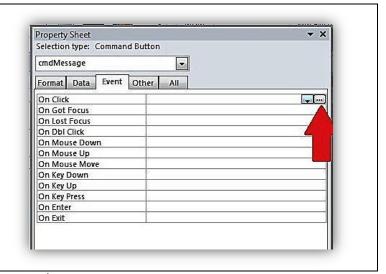


Figure 1.4

VBA Code not Working - Activating VBA Code

If you ever see this dialog:

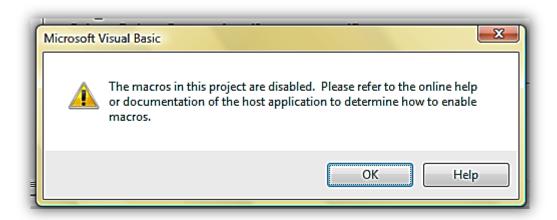


Figure 1.5

It is because you need to activate this:



Figure 1.6

Navigate back to Access front end.

Click on Options... and tick the Enable Content Checkbox. Done.

Note: This is a feature of Access 2007/2010/2013 which disables all VBA code until explicitly allowed to function. You can tell Access not to display this error but you need to set up a "Trusted Location", which is basically a nominated area where you can place programs that you know to be safe.

The VBA Editor Explained

There are five main areas of the editor that you need to know about. Here are four:

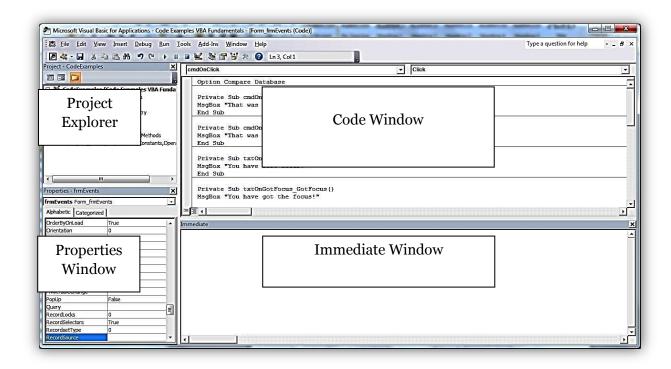


Figure 1.7

Code Window

The <u>Code Window</u> is where all your VBA code will be written. It has <u>syntax highlighting</u>, which means keywords in VBA, - such as Function, CStr, Return and others - will all appear in one colour, numbers in another colour and punctuation, comments and strings in yet another colour. It looks more appealing and makes reading lines of code much easier.

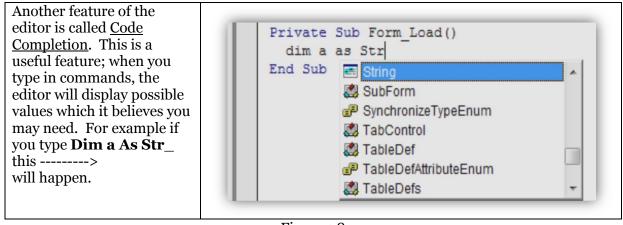


Figure 1.8

Project Explorer Tree

The <u>project explorer</u> shows you all the modules available in your Project - TeachingDB × database and any add-ins or **=** = | libraries you've included. ☐

Mile Teaching DB (DbVideo 50) Modules are kept in three areas: - 📰 Form_frmCarData Form_frmHome * Microsoft Office Access Class Form_frmHome1 Objects Form_frmHome2 Modules * Standard Modules 🦧 Module 1 Class Modules * Class Modules Select a form module and double click it to see any sub procedures or functions it may contain.

Figure 1.9

Microsoft Office Access Class Objects

These are VBA modules that are owned by (or children of) Forms and Reports. Their name will always be prefixed with "Form_" or "Report_". In these modules you will put all the event code that makes your forms and reports perform essential actions – like opening and closing. Unlike Standard Modules, code in these modules is not normally available outside the form, they are private.

Standard Modules

Standard modules contain code which may be accessed by any of the module types. In Standard Modules code will go that doesn't belong in forms or reports; for example a library of business rules or constants and types which are used by Forms and Reports. By default, anything written here is available anywhere else in the project – this is known as global scope.

Class Modules

Each Class Module contains code that revolves around a Class, which is a type of data-type or object. By default, anything written here is available elsewhere in the project.

Properties Window

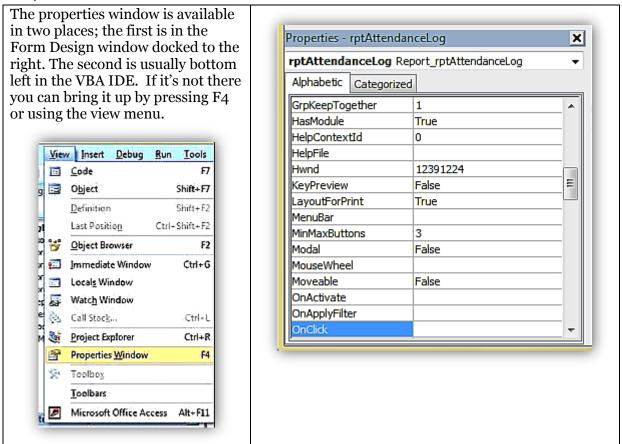


Figure 1.10

Immediate Window

The immediate window is located at the bottom of the screen and is the big blank window that says immediate in the title.

Using this window, you can test code snippets, test out your functions directly rather than through a form's button, and also debug your code.

Here is a small introduction. Try typing **print now()** and see what date and time come up. Then try **Print InputBox("what's your name?")**

```
print now()
24/12/2012 17:17:09

print date()
24/12/2012

print inputbox("What is your name?")
James
```

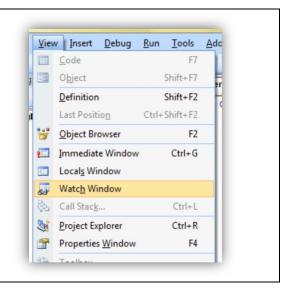
Figure 1.11

Watch Window

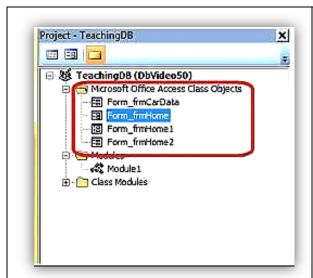
Figure 1.12

It is a little premature to bring in the Watch Window but you will be using it soon enough so just take note.

The Watch Window is used in debugging to watch and keep an eye on the variables in your code. You can also set a trap, so if one of your variables becomes a particular value, you can trigger a break which will cause your program to stop running so you can inspect its state. More on this topic in the next unit.



How to select different Forms and Reports (from project explorer)



In the <u>Project Explorer</u> the items contained within folders are all Modules.

Here we have 4 Form modules and 1 Standard module.

If you select and double click a form, the VBA module code will be displayed.

Ultimately, all forms will have a module of their own because forms look pretty but do little without VBA code.

You can explore this now by opening a form that doesn't have a module and giving it one. There are a few ways to do that but here's the most straightforward.

- Open a form in design view.
- Click on a control, like a button.
- In the Properties window click on the Events tab.
- Click the Ellipsis of the On Click event.
- Access will now automatically create a new module with the name Form_myForm!

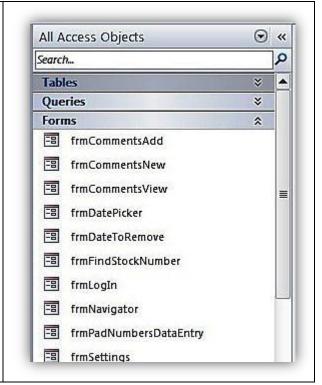


Figure 1.13

How to select different Modules

Modules can be seen in two places, in the Access IDE and the VBA Editor IDE.

If you want to add new modules use the instructions shown previous.

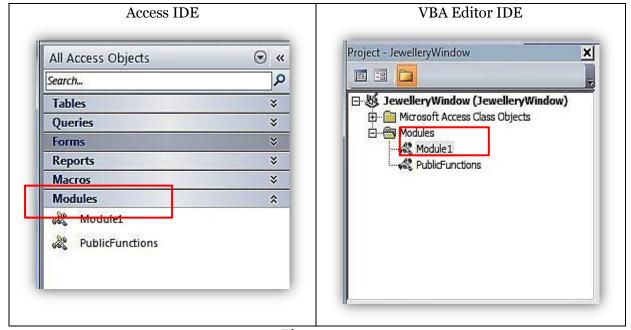


Figure 1.14

How to Rename Modules

Unlike other objects in Access renaming modules is easy, even when they are open.

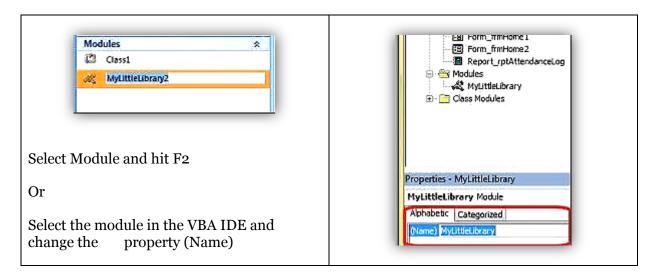


Figure 1.15

Basic Tools for Writing Code

The VBA Editor incorporates a number of useful features which help you whilst you are developing, testing and in production (some of which we have already touched on). Here we'll take a closer look at a few code writing features of the VBA editor.

Line Continuation Character

When we write code, we are often required to create string expressions that are wider than the page itself! Although the entire string is on one line, it makes it more difficult to code as you constantly have to use the horizontal scroll bar to read exactly what is written.



Figure 1.16

The smart people at Microsoft came up with a simple plan; the line continuation character, or space and underscore for short. We use this with an ampersand (&) to make our code easier to read.

```
| Compare Database | Compare Dat
```

Figure 1.17

Indenting Code

Another key assistant to reading our code is the indentation. Indentation gives us a clear indication of code blocks. Indentation is implicit, so typing in a Public Function Name() and pressing the enter key will add the End Function and indent your code by two or four spaces or a tab. This is standard practice across all programming languages.

```
Function showMeIndentation()
Dim a As Integer
Debug.Print "Start Function"
For a = 1 To 20
Debug.Print CStr(a)
Debug.Print CStr(a + 1)
Debug.Print CStr(a + 2)
Next
Debug.Print "End Function"
End Function
```

As you can see, it's easy to see which bits of code are associated with one another.

Figure 1.18

Editor Format to Adjust Colours

The above example also neatly brings us to syntax highlighting. All keywords in VBA are dark blue by standard and anything we write is in a black font (except comments which are green). Syntax highlighting serves to use our sense of colour to add meaning to the code. You can change the colours too! Follow the steps below: Click on the Tools drop down menu, select Options... tab over to Editor Format and change as needed. Working under poor lighting conditions makes the default black on white very uncomfortable, so perhaps a blackened theme would be more suitable.

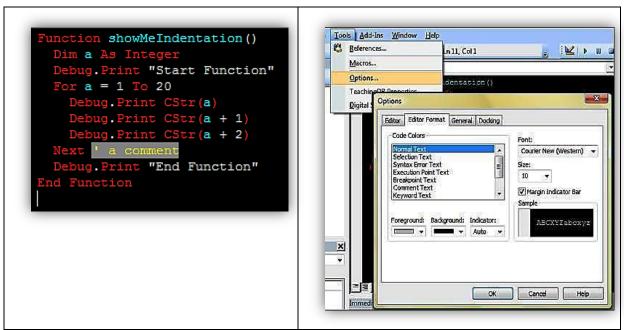


Figure 1.19

Naming Conventions

A naming convention is a way of naming variables and objects which your developer group uses to make your code easier to understand. In Figure 1.19, Function showMeIndentation() is in "camelCase". Here are some additional naming convention details:

- <u>Use meaningful variable names</u> make your variables mean something. Zzxd isn't meaningful, but fileNotFound has a semantic meaning for humans (even though it doesn't affect the computer or VBA in any way).
- <u>Use camelCase for variables and functions</u> that is, for every word in your variable name make the first letter upper-case, except the first letter of the first word. thisIsCamelCase()
- <u>Use CamelCase for classes, and types</u> capitalise your enumerations, user-defined types, class name, but leave functions and variables in camelCase.
- <u>Use UPPER_CASE for constants</u> when you declare a constant the name of that constant is usually capitalised. This means nothing to the compiler but means everything to you.

Amongst VBA developers, a typical naming convention is to start all variables with lower-case letters indicating the variable type (in lower case):

- iMyNumber would be of type Integer
- dblMyOtherNumber would be of type Double
- sText would be of type String

With form and report controls a three letter prefix is very common also:

- txtMyTextBox
- cboMyComboBox
- lblLabel

The real point of a naming convention, however, is to make your code more accessible to others by imposing on you and your colleagues a consistent way of writing code. Feel free within your departments or projects to use whatever naming convention you like, but the key is to be <u>consistent</u>.

If you are working in a large group of developers consider the following:

- keep a module which bears your name and contains the functions for which you are responsible
- prefix all your functions and subroutines with your initials the idea here being that your procedures and function do not clash with other developers

Select Object Dropdown and Procedures Dropdown

Hidden in plain view are two drop down menus just above the Code Editor Window. The drop down menu on the left is used to select and even indicate which control your cursor is currently in and the one on the right lists all available events for that object.



Figure 1.20



Figure 1.21

Procedural View and Full Module View

Another useful feature is the ability to switch between module and procedure view in the code window. Module view is what we normally see but if you have lots of procedures and functions and would like to only view the one you are working on, click the button on the left in Figure 1.22. All the other code will magically disappear!

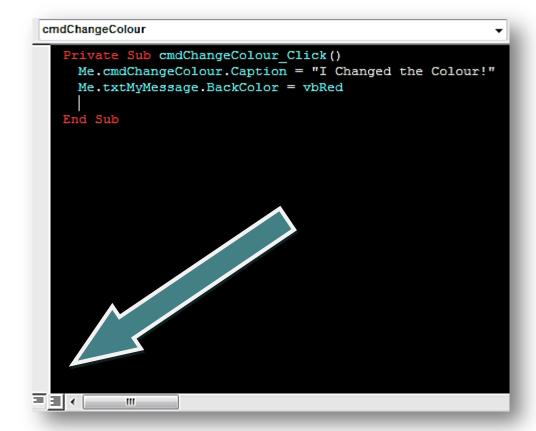


Figure 1.22

DoCmd Syntax and Arguments Explained

One of the most versatile objects in Access VBA is DoCmd. This one object gives you the programmer control over how your application operates, issues orders and gives you access to features that are otherwise confined to the GUI.

For example: on a form you have a button with the caption "Close". Clicking it does nothing, because we haven't added code. So, open a form, add a button control, tab over to Events and click OnClick. Now enter the following code into the procedure:

```
DoCmd.Close 'closes the presently active window

DoCmd.Close acForm, Me.Name 'Closes any windows with the 'Name of Me.Name
```

Figure 1.23

DoCmd has some other useful functions listened below.

```
DoCmd.OpenForm "name" 'does just that
DoCmd.MoveNext 'advances the form's recordset cursor by one
DoCmd.FindFirst 'finds first record in database – may have been given search parameters
DoCmd.FindNext
DoCmd.Maximise
```

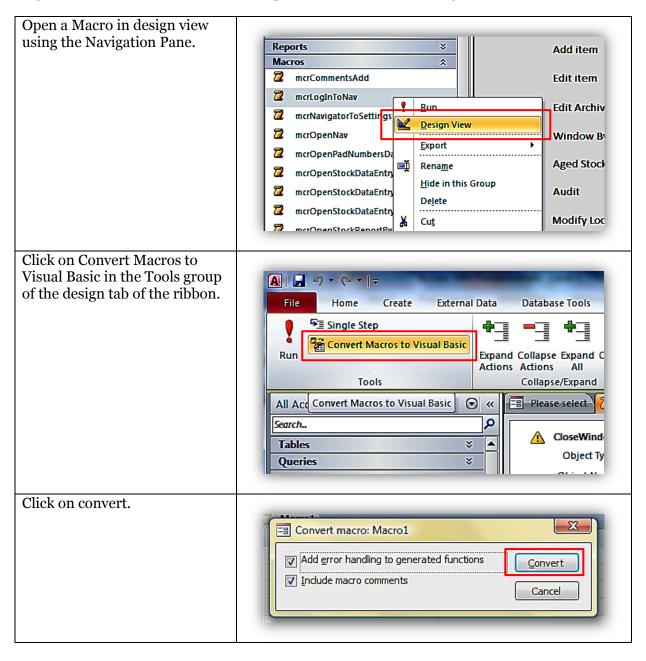
DoCmd.Minimise

DoCmd.RunCommand 'this one method has hundreds of commands relating to the GUI DoCmd.Close acForm, Me.Name 'closes any windows with the Name of Me.Name DoCmd.RunSQL 'excellent for quickly executing an action query when Recordset setup would take too long

Figure 1.24

How to Convert a Standalone Macro

A Macro is an object that includes a list of instructions, and these instructions translate to VBA code. Rather than writing your code in VBA you could, and probably have already, put together a few Macros to close forms, open forms, email data, navigate records, etc.



When the conversion has finished, a new Standard ▼ mcrLogInToNav Module is created with all the code for the macro contained mcrLogInToNav within. Function mcrLogInToNav()
On Error GoTo mcrLogInToNav_Err Just like Macros, the code in DoCmd.Close acForm, "frmLogIn"
DoCmd.OpenForm "frmNavigator", acNormal, "", "", , acNormal the Converted Macro module is mcrLogInToNav_Exit: Exit Function available elsewhere in your project. mcrLogInToNav Err: MsgBox Error\$ Resume mcrLogInToNav_Exit End Function

Figure 1.25

How to Convert a Form's Embedded Macros

Embedded Macros are children of their parent form. You will tell an embedded macro from an event procedure or normal macro because the Property Sheet of the Form Events tab give you [Embedded Macro].

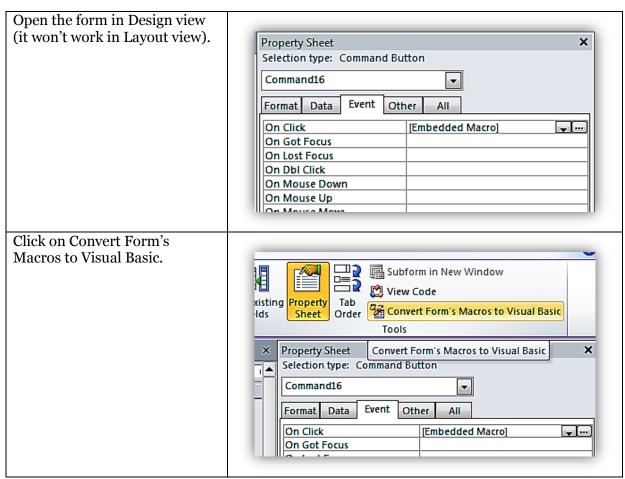


Figure 1.26

When to use Macros and VBA

Macros are objects in the Access IDE that perform many application functions without needing to resort to code in VBA. VBA on the other hand does everything a macro can do and a whole lot more. Whether you choose to use a Macro or VBA will be down to your

preference and familiarity with VBA or macros. In this series, we are learning about VBA so we will exalt its merits over macros. But for many simple tasks macros are just fine.

Questions

1. Examine the following code. Will it do as the user expects? If not why not?

```
1
   Sub Message()
2
3
   Dim strMessage As String
4
5
   strMessage = "What is your name?"
6
7
        MsqBox
        strMessage ,_
8
        vbQuestion ,_
9
        "Name"
   End Sub
```

2. Do the following and answer the questions below.

Create a new form called "Ex_2_Form1".

Add a combo box, a textbox and a label.

- a. What is the third event in the properties window for combo box?
- b. On the label what is the third event property called?
- c. Does the textbox have a caption property?
- d. Select the Detail part of the form, click on Other. What is the name of the object?
- e. Select the form, open Other in property tab. What does "Cycle" mean?
- 3. True or false ...
 - a. Macros are VBA code?
 - b. The Form Editor allows you to view and edit VBA code?
 - c. The Property Sheet in Form View is the same as the Properties window in VBA IDE?
 - d. IDE means Individual Data Execution?
 - e. Macros have fewer commands than VBA code?
 - f. The Immediate window allows code snippets to be tested and executed?
 - g. A Class Module is a VBA symbol of elitism over other languages?
 - h. The Watch window lets you watch the values of variables?
- 4. A module window is open, you have edited some code. You want to change the module's name. How do you do this? (hint: there are two ways).
- 5. Indenting code is a form of what?
 - a. Syntactic sugar.
 - b. Readability aid.

- c. Banging out code
- 6. Which desert animal features predominantly in variable naming conventions?
- 7. For the following prefixes what is the most likely variable or object type?
 - a. txt
 - b. i
 - c. cmb
 - d. lbl
 - e. lng
 - f. s
 - g. frm
 - h. qry
 - i. mcr
 - j. C
- 8. What must you do to convert a macro into a VBA function or procedure?
- 9. Which of the following are methods or variables that belong to the DoCmd object?
 - a. Close
 - b. RunSQL
 - c. MimicForm
 - d. Open
 - e. SetWarnings
 - f. Crash
 - g. Beep
 - h. RunSavedImportExport
 - i. LockNavigationPane
 - j. OpenHeadUpDisplay
 - k. FindNextRecord
- 10. True or False? Code completion is a ...
 - a. ...type of artificial intelligence.
 - b. ...VBA IDE feature.
 - c. ...developer assistant.
 - d. ...programming race.
- 11. Go to the Project Window and add two new Modules. What are their default names?

- 12. You are testing out some programming features and decide to use the Immediate Window to test your functions. What is wrong with the following code?
 - a. Debug.Print (newFunctionTest 1)
 - b. Print "This May"; "be my last chance "+_ "to get this right"
 - c. Dim ab As String
 - d. Ab = Array("a", 3)
 print Ab(2)
- 13. Why might the Project explorer window not have a form's module?
- 14. Open a form in design view and click back into the Navigation Pane Why can't you change the form's name?
- 15. The Line Continuation Character is a what?
 - a. An actor.
 - b. An overstatement.
 - c. A space followed by an underscore.
 - d. A VBA bloodline.
- 16. Procedural View or Full Module View? Which allows an overview of the code in the VBA IDE?
- 17. Embedded Macros are not available in the Navigation Pane. Why?
- 18. Why are Microsoft Office Access Class Objects Modules not visible in the Navigation Pane?
- 19. By default functions and procedures in a Standard Module are...
 - a. Private
 - b. Public
 - c. Protected
 - d. static
- 20. True or False? It is only possible to convert stand-alone macros in VBA.

Answers

- 1. MsgBox doesn't request a user's input
 - The line continuation characters are next to the commas, no whitespace
- 2. a) After Update, b) mouse down, c) detail, d) Detail,
 - e) how the form reacts when one is in the last field and tabs once more
- 3. True or false
 - a. False
 - b. False
 - c. True
 - d. False
 - e. False
 - f. True
 - g. False
 - h. True
- 4. a) right click on module in project windows and rename
 - b) f2 the module in the navigation panel
- 5. (b)
- 6. A camel
- 7. Multiple answers
 - a. Textbox
 - b. Integer
 - c. Combobox
 - d. Label
 - e. Long
 - f. String
 - g. Form
 - h. Query
 - i. Macro
 - j. Class
- 8. Click Convert Form's Macros to Visual Basic
- 9. True or false
 - a. True
 - b. True
 - c. False
 - d. False
 - e. True
 - f. False
 - g. True
 - h. True
 - i. True
 - j. False
 - k. False
- 10. (b) and (c)
- 11. **Module 1** and **Module 2** or a similar consecutive numeration
- 12.
- a. Print is not a function
- b. +_, _ should not be there
- c. Dim is not allowed in the immediate window

- d. Ab(2) is out of bounds, valid values would be 0 or 1
- 13. Because a form has no module yet.
- 14. Because the form is open. Forms must be closed to rename.
- 15. (c)
- 16. Full Module View.
- 17. Because they are embedded into a form or report, their parent is the form.
- 18. Because they are children of their parent form.
- 19. Public
- 20. False