Access made easy

Introduction To VBA

10

Robert Austin

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.

Contents

Introduction to VBA

To whet your appetite and to help get more out of your applications this unit is intended as an introduction to Microsoft Visual Basic for Application, VBA for short. There is a whole course dedicated to VBA so what we have here is only a taster, but work your way through this unit and you'll be able to write your own VBA code in just a few minutes.

Macros and VBA

We should really call this "VBA and Macros" but by now you are quite familiar with macros, so to introduce VBA we will compare it with macros.

When writing your macros each line is an Action; a form is opened, a record is sorted, a query is executed, another form is opened. VBA is not much different except that you have more control in VBA and VBA offers you a vastly more creative resource than macros.

In essence a macro action is a VBA command, for example in Macro speak we open a form with:

- OpenForm formName, Normal

In VBA speak we say:

- DoCmd.OpenForm formName, acNormal

Let's get stuck in, and we'll start by replacing an open form macro with an open form VBA code.

Getting Started!

To start then, create two forms and save them.

Form1

- Add a label with the caption: This is Form1
- Add a button with the caption: Open Form 2
- Name the button cmdOpenForm2
- Save as Form1

Form2

- Add a label with the caption: This is Form 2!
- Add a button with the caption: Open Form 1
- Name the button cmdOpenForm1

Save both of those forms before you progress.

Opening the VBA editor

There are a couple ways to open the editor, but we are going to use one we know of already.

VBA Editor through the Form Designer Properties Window

Open form1 in Design View.

Highlight the button, mouse over to the property sheet and click on the Events tab.

Click on the ellipsis (...) to open the VBA Editor.

Select Code Builder from the dialog box

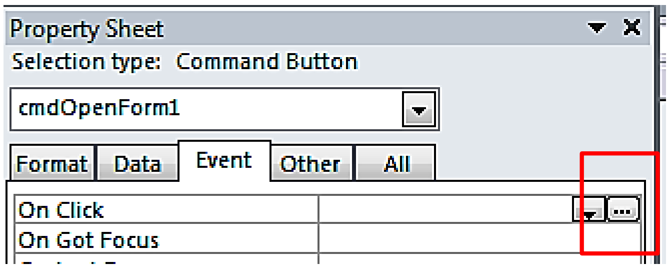
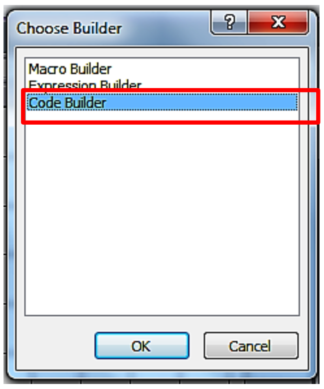
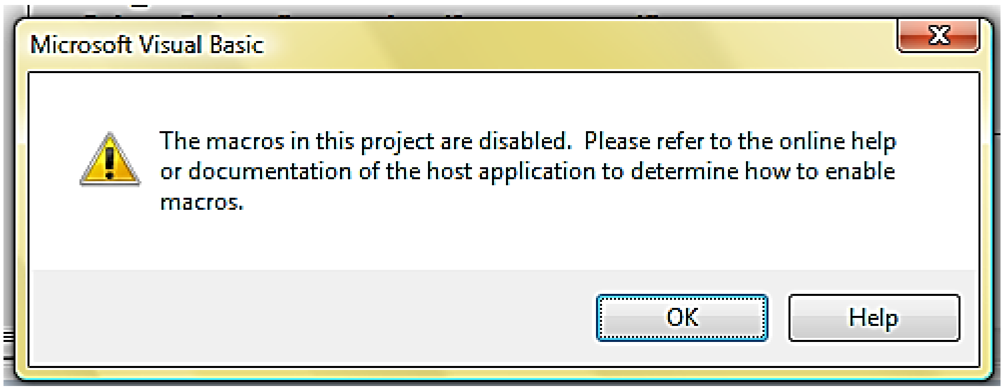


Figure 10.1



If you see this warning - Activate VBA Code

If you see this dialog:

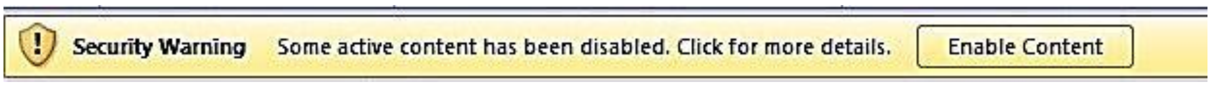


Microsoft Visual Basic

The macros in this project are disabled. Please refer to the online help or documentation of the host application to determine how to enable macros.

Figure 10.2

It is probably because you need to activate this:



Security Warning Some active content has been disabled. Click for more details.

Figure 10.3

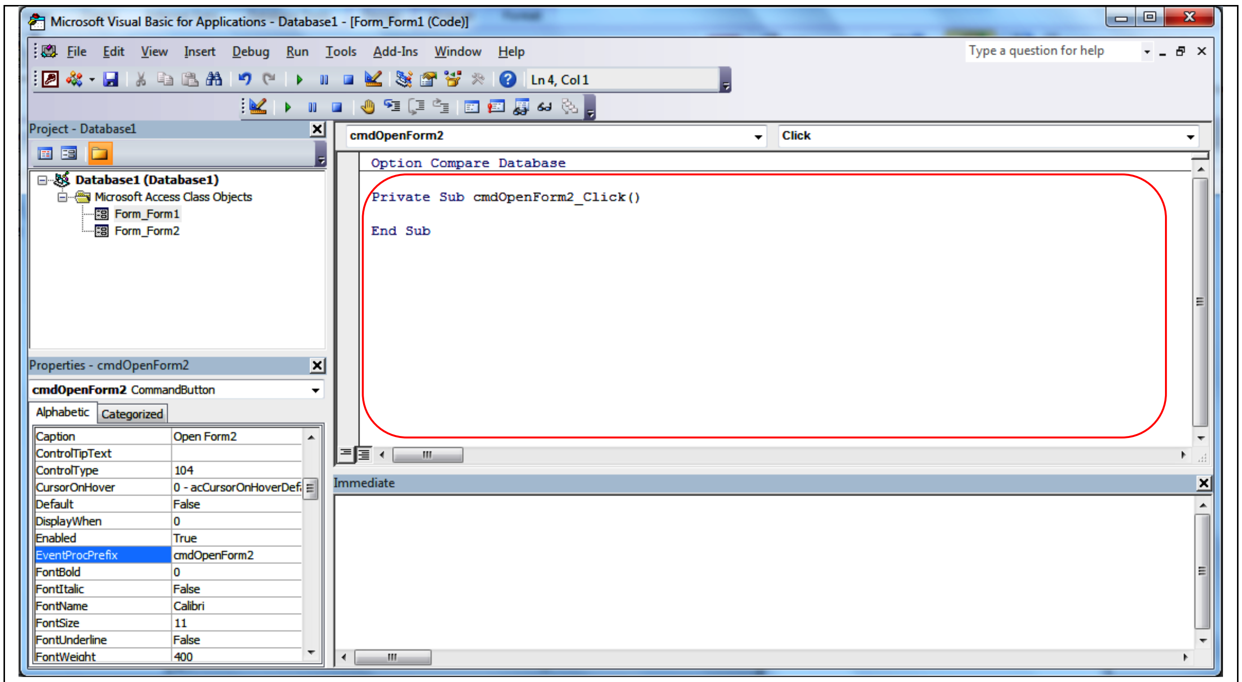
Open and Close the project and click Enable Content when this Warning Appears

This is a feature of Access 2007/2012 which disables all VBA code until explicitly allowed to function. You can setup Access not to display this error but you need to set up a "Trusted Location" which is basically a nominated area of which Access and you assume items in said area are safe to run.

Mostly Harmless

When you click on the ellipsis you will get the window below.

Don't be afraid of it, it is mostly harmless. The only bit we are interested in at the moment is the big white space in the middle – yours will not have a red box around it.



```
Option Compare Database

Private Sub cmdOpenForm2_Click()

End Sub
```

Caption	Open Form2
ControlTipText	
ControlType	104
CursorOnHover	0 - acCursorOnHoverDef
Default	False
DisplayWhen	0
Enabled	True
EventProcPrefix	cmdOpenForm2
FontBold	0
FontItalic	False
FontName	Calibri
FontSize	11
FontUnderline	False
FontWeight	400

Figure 10.4

Click Event

Your cursor will be flickering between the two lines as below in Figure 10.5.

```
Private Sub cmdOpenForm2_Click()
|
End Sub
```

Figure 10.5

`cmdOpenForm2` is the name of the button you highlighted before clicking the ellipsis; `Click()` is the Event of the button; so `cmdOpenForm2_Click()` is the full name of this procedure and it is run when cmdOpenForm2 is clicked.

Make your code look the same as the code below:

```
Private Sub cmdOpenForm2_Click()
  DoCmd.OpenForm "Form2"
End Sub
```

Figure 10.6

Save it and go back to your form using Alt+Tab.

Now click on the Open Form 2 button and, viola! Form2 opens.

Now do the same for form2's button. The button this form will also be called button1 and have exactly the same text in the VBA window. Make your code as the one below.

```
Private Sub cmdOpenForm1_Click()
  DoCmd.OpenForm "Form1"
End Sub
```

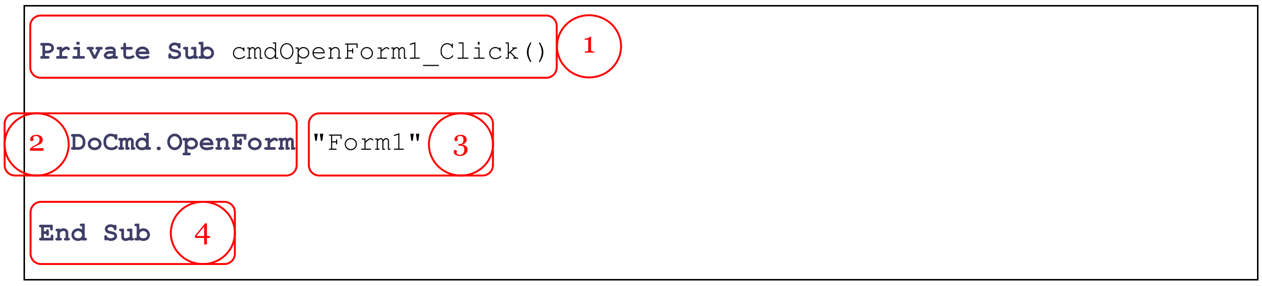
Figure 10.7

Save that, flick back to your form and click the Open Form 1 button.

The Anatomy of a Procedure

Let's dissect the procedure and see what we have.

1. One is the procedure's name.
2. Two is an action! Two can be a long list of actions, or commands, just like a macro.
3. Three is the name of the form. It's in quotation marks because if it were not VBA would not understand it as the name of our form.
4. Four marks the end of the procedure.

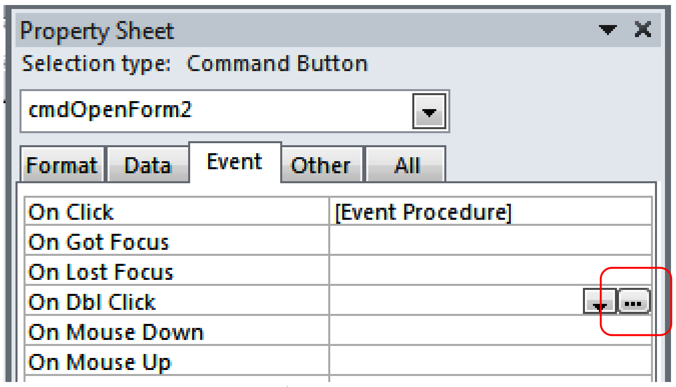


```
Private Sub cmdOpenForm1_Click()

  DoCmd.OpenForm "Form1"

End Sub
```

All procedures begin with `Sub` and finish with `End Sub`.

Double Click Event

Let's do another one. Go back to `Form1`, highlight `cmdOpenForm2`, go back to the Events property sheet and click the ellipsis of On DblClick.


Property Sheet
Selection type: Command Button

cmdOpenForm2

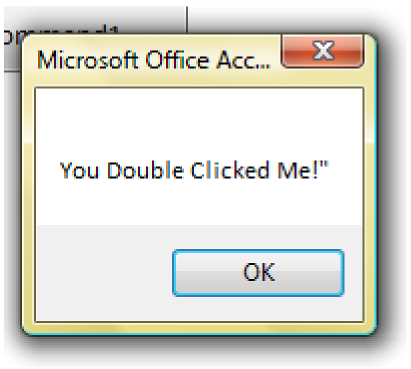
On Click	[Event Procedure]
On Got Focus	
On Lost Focus	
On Dbl Click	
On Mouse Down	
On Mouse Up	

Figure 10.5

In the VBA editor make your code look like the code below:

```
Private Sub cmdOpenForm2_DblClick()
  MsgBox "You Double Clicked Me!"
  DoCmd.Close acForm, "Form1"
End Sub
```

Now, go back to your form and double-click the button.


Microsoft Office Acc...

You Double Clicked Me!"

Figure 10.6

That second line in the procedure, if you have not yet guessed, closes form 1. You can change "Form1" to "Form2" and it will close Form2 instead (as long as Form2 is open).

So, a procedure performs an action or several actions that you put between the `Sub` and `End Sub`.

DoCmd Object

No doubt you will have noticed as you typed Visual Basic was bringing up a set of lists and yellow hints that sometimes get in the way. This is all called *Intellisense* and VBA is trying to be helpful by showing us what we can do with MsgBox and DoCmd and other objects.

As you can see below, DoCmd has a lot of things it can do. All those flying green boxes are procedures, or Actions, and they are mostly the same Actions you saw in the Macros. And that is not a coincidence. Macros mostly work with the DoCmd object in the background which is why you can see them here. Scroll down the list and take a look at the procedures that are available.

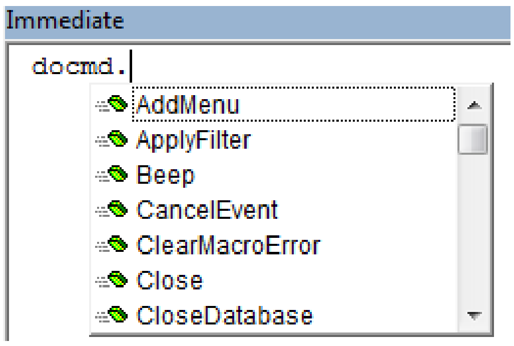


Figure 10.7

In your Click() procedures you can experiment with the DoCmd object.

Functions and Procedures

One other type of routine that VBA has is called a function. A function is different from a procedure only because it returns a value. It is easier to see it in operation than to talk about it. Open Form1 again, highlight cmdOpenForm2 and bring up the [Event Procedure] for OnClick.

We will use a function called `InputBox` to demonstrate functions. Put this code into your procedure and click on your form button to execute it. Then come back here.

```
Private Sub cmdOpenForm2_Click()
  Dim a As String
  a = InputBox("What is your name?")
  MsgBox a
End Sub
```

Did you see what happened? You entered your name into the field and then it was displayed in a message box.

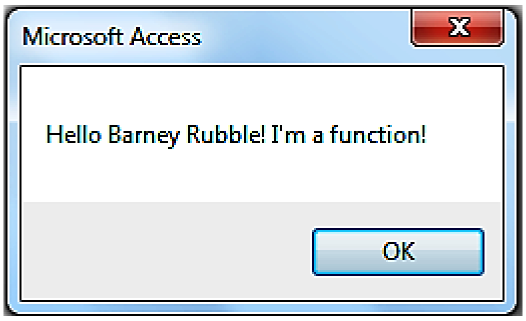
Here **`InputBox`** is a function and what it did was take from you a value and return that value into a, then `MsgBox` displayed the content of a, which was your name. A Function returns a value.

Functions are extremely useful and VBA has hundreds of them built in.

Let's do another one. Again replace the content on `cmdOpeForm2_Click()` and add the function underneath it.

```
 1  Private Sub cmdOpenForm2_Click()
 2    Dim a As String, b As String, c As String
 3    a = InputBox("Enter your name")
 4    b = InputBox("Enter your surname")
 5    c = sayHello(a, b)
 6    MsgBox c
 7  End Sub
 8
 9  Function sayHello(a As String, b As String) As String
10    sayHello = "Hello " & a & " " & b & "! I'm a function!"
11  End Function
```

Now click on the button and you will be asked for two numbers. Make them simple and see what happens.



Hello Barney Rubble! I'm a function!

Figure 10.8

Here we have demonstrated two things. The first is that a function returns a value. InputBox asked for your name and surname and it saved those values into a and b. The function `sayHello` (a,b) took those two values, added them together with some text and saved the new value into c. Lastly, the action MsgBox displayed the result. So, functions return values.

The other things we demonstrated here is that we can save values into letters (those letters Visual Basic calls "variables"). On lines 3 and 4 Visual Basic saves the returned value from InputBox into letters **a** and **b**. Then on line 5 Visual Basic gives the values of **a** and **b** to the function `sayHello()`; function `sayHello`() adds them with other text and returns the result which is saved into **c**.

Line 6 then just finishes off by displaying the value of c.

These techniques are very useful. In the same way we can combine many database fields to print addresses for letters and labels, joins names for client accounts on web pages, format product descriptions, add dimensions, makes lists of textual information and much more – much more than we could have done using macros.

Making a Navigation Form!

Putting it all into practice

This is where you get to test out all the skills you've learnt over the course and add a little VBA code to knit everything together.

In the databases you have used there are a number of tables, queries, other forms, some reports and macros. Using the form designer create a form that opens two of each type of Access object; two tables, two queries, two other forms, two reports and runs some Macros.

You will need a few extra VBA tools and they have been documented below.

Tips

All controls have formatting options, some more than others.

Labels – use labels to add headings to each group of buttons and add emphasis. Fonts can be changed, make bigger, background colours and border colours. Webdings fonts can be used to add pictures.

Buttons – the shape and appearance of buttons can also be changed. Fonts can be enlarged, different fonts selected.

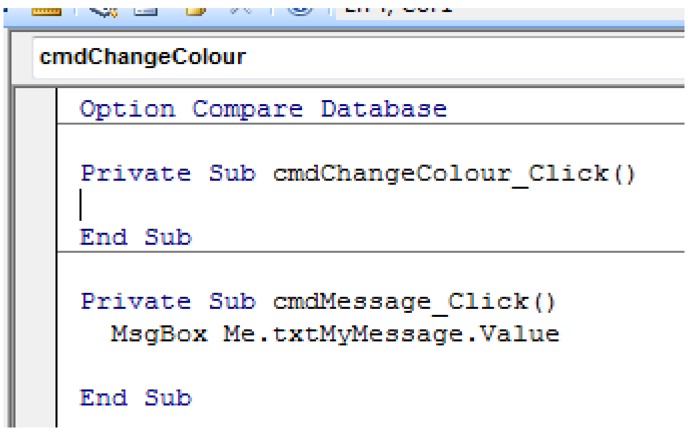
Images – images can be added like icons of discs, bins, reports or any other image. An image can fill the whole form and be sent to the back so all your controls float on top of it.

Use the following commands to open Access objects. Change the text that is underlined to the correct names of the objects you want to open.

```
DoCmd.OpenForm FormName:="Customers", View:=acNormal

DoCmd.OpenReport ReportName:="CustomerPhoneList", View:=acViewNormal

DoCmd.OpenQuery TableName:="Sales Totals Query", , acReadOnly

DoCmd.OpenTable "Employees", acViewPreview

DoCmd.RunMacro "Print Sales"
```

Questions

1) Look at the picture below and answer the following questions
    a. What is the name of the first procedure?
    b. What is the name of the second procedure?

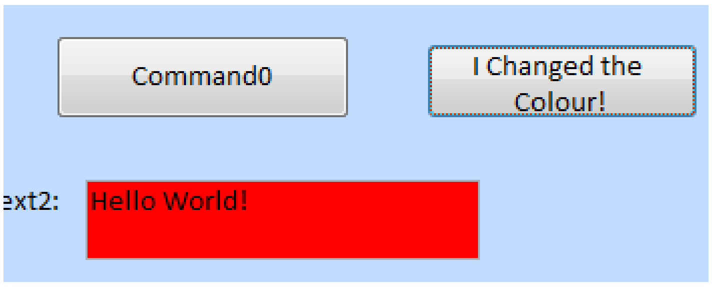


cmdChangeColour

```
Option Compare Database

Private Sub cmdChangeColour_Click()

End Sub

Private Sub cmdMessage_Click()
  MsgBox Me.txtMyMessage.Value

End Sub
```

2) Which of the following events will have happened when these procedure execute?
    a. cmdChangeColour has been clicked
    b. cmdMessage has been double clicked
    c. cmdChangeColour has been deleted
    d. cmdMessage has been clicked
    e. none of the above

3) The code above belongs to the buttons below.
    a. Which procedure will call the second button?
    b. Which procedure will call the first button?

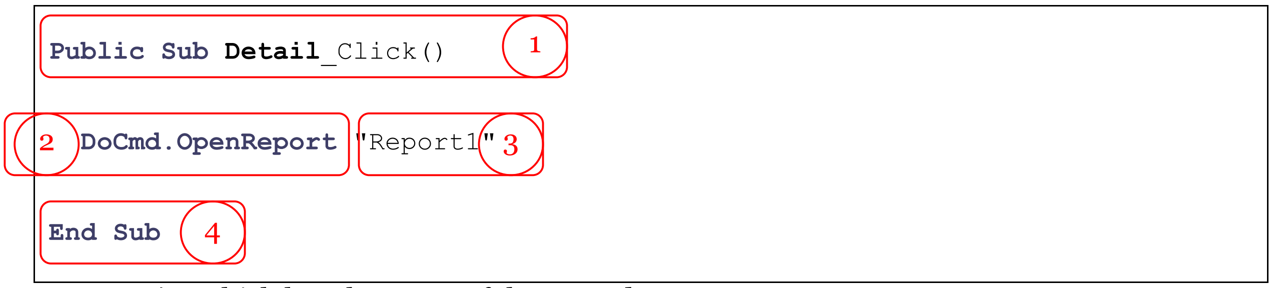


4) Macros and VBA, true or false
    a. Macros cannot open forms
    b. There are tasks a macro can do that VBA cannot
    c. Macros can be more complicated than VBA
    d. Macros cannot directly use functions
    e. Macros can call VBA routines

5) Which of the following are true of functions
    a. Parameters cannot be passed to functions
    b. They return a value

c. Cannot do everything a procedure can
d. InputBox is a function
e. MsgBox is a function

6) See the procedure below and provide 1, 2, 3 or 4.



```
Public Sub Detail_Click()

DoCmd.OpenReport "Report1"

End Sub
```

a) Which has the name of the procedure
b) What box has the end of the procedure
c) How many actions does the procedure perform
d) Which circles the command executed
e) What is the name of the object being acted upon

7) All events are
   a. Sub-routines
   b. Functions

8) Why is intellisense useful?

9) What might the function below do?

```
Private Function found(needle As String, haystack As String) as YesNo

  Found = instring(needle, haystack)

End Sub
```

10) From VBA which of the following Access Objects can you open?
   a. Form
   b. Report
   c. Query
   d. Macro
   e. Module

11) What can DoCmd.Close do?
   a. Close Forms
   b. Open Reports
   c. Close queries

d. Close Modules
e. Open macros

12) Which of the following is not likely to be a command of DoCmd?
   a. OpenDatabase
   b. OpenShop
   c. FindRecord
   d. Save
   e. MoveNext

13) Which of the access objects would be best to make a navigation system from?
   a. Queries
   b. Reports
   c. Forms
   d. Modules
   e. Table

14) Which of the following statements is true of procedures?
   a. They return values
   b. They accept parameters
   c. They are mostly used for actions
   d. They cannot be used in macros
   e. VBA allows us to write our own procedures

15) Write a procedure that will close report called "rptBananaExportPrices"

16) Write a procedure that will open a form called "frmShoppingCart" and "frmShoppingList"

17) Write a function that only returns the value True

18) In the following procedure what is the name given to a, b and c?

```
Private Sub button1_Click()
  Dim a As String, b As String, c As String
  a = InputBox("Enter your name")
  b = InputBox("Enter your surname")
  c = funcAdd(a,b)
  MsgBox c
End Sub
```

19) Identify all the functions in the above procedure

20) Identify all the procedures in the above procedure

Answers

1) See below
    a. cmdChangeColour_Click()
    b. cmdMessage_Click()
2) see below
    a. yes
    b. no
    c. no
    d. yes
    e. no
3) see below
    a. cmdChangeColour_Click()
    b. cmdMessage_Click()
4) see below
    a. false
    b. false
    c. false
    d. false, they can in their condition statements
    e. true
5) see below
    a. false
    b. true
    c. false
    d. true
    e. trick question. MsgBox can act as a function or procedure
6) see below
    a. 1
    b. 4
    c. 1
    d. 2
    e. Report1
7) A) sub-routines
8) Because it shows the developer what routines and functions are available to use
9) Finds the needle in the haystack
10) All
11) See below
    a. True
    b. False
    c. True
    d. True
    e. False
12) (b) only
13) (c) only
14) (b), (c), (e)
15) Must have at least the following elements
    a. Sub rptBananaExportPrices()
    b. DoCmd.Close “rptBananaExportPrices”
    c. End Sub

16) Must have at least the following elements
    a. Sub openTwoForms()
    a.   DoCmd.OpenForm "frmShoppingCart"
    b.   DoCmd.OpenForm "frmShoppingList"
    b. End Sub
17) Must have at least the following elements
    a. Function returnTrue()
    b.   returnTrue = true
    c. End function
18) They are variables
19) inputBox, funcAdd
20) MsgBox