# Quiz 2 solutions and explanations

**IMPORTANT: Even if you do not intend/need to look through the solutions to Quiz 2, you still need to mark this quiz as completed using the blue "Mark as Completed" button in the lower right of this page. By doing so, you will unlock Assignment 2!**

This document is meant to provide clear explanations for the Quiz 2 questions (not the in-video quizzes since they have explanations already). I do NOT provide feedback during the quiz (like I do for the screencasts) because a learner could just guess, obtain the correct answers, then put them back into the quiz and get 100%!

This document is purely for you to learn more and to correct your misconceptions about the material. If you view this document soon after you take the quiz to see why you missed a certain question, it will serve as a great learning tool!

PLEASE DO NOT SHARE THIS DOCUMENT WITH ANYONE! Using this document to complete Quiz 2 is a violation of Coursera's Honor Code (a.k.a. cheating).

**Question 1:**

Which of the following statements regarding material covered in this module are TRUE (multiple answers possible)?

A. Functions can be stepped through just like subroutines by using F8. In other words, the cursor can be placed within the function code and F8 will step through the function just like with subs.

Incorrect. In order to step through functions, you must first set a breakpoint within the function code (usually, the first line of executable code, after any Dim statements). Then, you must enter the function in a cell of the worksheet. When you press Enter, the code will stop at the breakpoint and from here you can use F8 to step through the function line by line.

B. If you wished to run another subroutine that resides in the same workbook, you could use the Call statement.

Correct! This will work.

C. Functions must have arguments.

Incorrect. Some examples are the PI() and NOW() functions in Excel. You can create similar argument-less functions in VBA.

D. Subs must have arguments.

Incorrect. Most subroutines do NOT have arguments.

E. Subs can have arguments.

Correct! Yes, an example is if you wanted to place a 1 in every cell starting in A1, you could pass the number of row and the number of columns into the subroutine. Or, you could delete columns c1 through c2, and the subroutine would need to know c1 and c2.

**Question 2:**

Which of the following VBA statements could you use to implement the following algebraic equation? Select all that apply.

$$y = \frac{1}{\sqrt{2\pi c}}$$

A. y=1/Sqr(2*4*Atn(1)*c)

Correct! Sqr is the correct function for the square root function in VBA. There is no built-in pi function/value, so we can use 4*Atn(1) for pi.

B. y=1/Sqr(2*WorksheetFunction.PI()*c)

Correct! Sqr is the correct function for the square root function in VBA. There is no built-in pi function/value, so we can borrow Excel's PI() function.

C. y=1/Sqrt(2*4*Atn(1)*c)

Incorrect. While 4*Atn(1) is suitable for pi, the square root function in VBA is Sqr, not Sqrt.

D. y=1/Sqr(2*PI()*c)

Incorrect. Sqr is the correct function for the square root function in VBA, but there is no built-in function PI in VBA.

E. y=1/Sqrt(2*WorksheetFunction.pi()*c)

Incorrect. We can use Excel's PI function, but the square root function in VBA is Sqr, not Sqrt.

**Question 3:**

Which of the following procedures (multiple answers possible) show proper syntax or "etiquette"?

A.

```
Option Explicit

Sub IsThisOkay()
Dim x As Double, y As Double
x = 2
y = 1
Range("A1") = x * (2.15 + 0.35 * y) ^ 1.8
End Sub
```

Correct! Option Explicit appears at the top, which requires variable declaration. All variables (x and y) are Dim'med.

B.

```
Option Explicit

Sub IsThisOkay()
Dim x As Double, y As Double
x = 2
y = 1
IsThisOkay = x * (2.15 + 0.35 * y) ^ 1.8
End Sub
```

Incorrect. Option Explicit appears at the top, which requires variable declaration. All variables (x and y) are Dim'med. However, the output of a subroutine is NOT the name of the subroutine, so the second to last line will trigger an error.

C.

```
Option Explicit

Function IsThisOkay(x As Double, y As Double) As Double
Dim x As Double, y As Double
x = 2
y = 1
IsThisOkay = x * (2.15 + 0.35 * y) ^ 1.8
End Function
```

Incorrect. This is a function. Arguments are only Dim'med in the parentheses on the first line and are NOT Dim'med separately in Dim statements. There will be no compiler error, but x and y will be reset to zero, which is not what is desired.

D.

```
Option Explicit

Function IsThisOkay(x As Double, y As Double) As Double
x = InputBox("Please enter x:")
y = 1
IsThisOkay = x * (2.15 + 0.35 * y) ^ 1.8
End Function
```

Incorrect. You should NEVER use input boxes or message boxes in functions!

E.

```
Option Explicit

Function IsThisOkay(y As Double) As Double
Dim x As Double
x = InputBox("Please enter x:")
IsThisOkay = x * (2.15 + 0.35 * y) ^ 1.8
End Function
```

Incorrect. You should NEVER use input boxes or message boxes in functions! Instead, x should be entered as another argument to the function.

F.

```
Option Explicit

Function IsThisOkay(x As Double, y As Double) As Double
MsgBox x * (2.15 + 0.35 * y) ^ 1.8
End Function
```

Incorrect. You should NEVER use input boxes or message boxes in functions!

**Question 4:**

Which of the following statements regarding material covered in this module are TRUE (multiple answers possible)?

A. In the VBA code below, the (x + 5) term is the first term to be calculated in the order of operations.

```
z = (x + 5) / (Abs(y) - 7)
```

Incorrect. Functions are always calculated first, so the Abs function would be calculated before the (x + 5) term.

B. The Application.Run method can only be used to run a procedure in an open file.

Incorrect. This method can be used to run a procedure in a closed file, too!

C. The way to call a function that resides in the same workbook from within a sub in that same workbook is to use the Call statement. For example, to call the MyFunction function shown below in a sub, we could write Call MyFunction.

Incorrect. The way to call a function is just to type in the name of the function with the arguments. For example, if we needed to use MyFunction in a subroutine, we would just type in something like “z = MyFunction(5)”.

D. One way to calculate the base-3 logarithm of a number x in VBA is: log(x)/log(3).

Correct! This is known as the change of base theorem.

E. In the order of operations, function values are always calculated first.

Correct! Function values are always calculated first.

**Question 5:**

What will be displayed in the message box when the CalculateMe subroutine is executed?

```
Option Explicit

Sub CalculateMe()
Dim x As Integer, y As Integer
x = 3
y = Modify(x)
MsgBox ModifyAgain(y, x)
End Sub

Function Modify(y As Integer) As Integer
Modify = y ^ 2 - 2
End Function

Function ModifyAgain(x As Integer, y As Integer) As Integer
ModifyAgain = 2 * x + y
End Function
```

Answer: 17

Solution explanation: Don't let the order of the arguments confuse you! Arguments are just "space holder" variables, and are only valid within individual functions.

When the CalculateMe sub is executed, x is set to 3, then y is set to Modify(3). Modify(3) will take 3, square it, and subtract 2, so the output of the Modify(3) function is 7. So, y in the CalculateMe sub will be 7. The next line in the CalculateMe sub is to message box the answer of ModifyAgain(7,3). Here is where you have to disregard the order of x and y, because again they are just space holder variables. ModifyAgain(7,3) will take 2 times x and add that to y. So, we'll take 7*2 + 3 = 17. This is what will be displayed in the message box in the CalculateMe subroutine.