

## **Assignment 17**

### **1. What are modules in VBA and describe in detail the importance of creating a module?**

**Bottom line:** In this post we compare the different places to store macros, functions, & VBA code in Excel. We specifically look at the Code Modules, Sheet Module, and This Workbook Module to learn the differences between how each works. We also learn how to run macros based on events or actions by the user.

**Skill level:** Intermediate

### **2. What is Class Module and what is the difference between a Class Module and a Module?**

As you probably know a standard module can store procedures and functions which can be either Private or Public (the default) and can be accessed either from within that module only (Private) or from anywhere in the project (Public). Modules are also used to store variables, constants and declarations (i.e. API calls) that will need to be accessed from anywhere in the project.

Class modules allow you to create your own objects which can have their own properties and methods like any other object (range, worksheet, Excel, chart, blah, blah). The best way of describing is by use of a simple example. Say you wanted to have code that would allow you to create an invoice object, assign it a number, customer name, net amount, and VAT. You'd then want to save that invoice as a new workbook. Your code in a standard module might look like this:-

### **3. What are Procedures? What is a Function Procedure and a Property Procedure?**

A property procedure is a series of Visual Basic statements that manipulate a custom property on a module, class, or structure. Property procedures are also known as *property accessors*.

Visual Basic provides for the following property procedures:

- A Get procedure returns the value of a property. It is called when you access the property in an expression.
- A Set procedure sets a property to a value, including an object reference. It is called when you assign a value to the property.

You usually define property procedures in pairs, using the Get and Set statements, but you can define either procedure alone if the property is read-only ([Get Statement](#)) or write-only ([Set Statement](#)).

You can omit the Get and Set procedure when using an auto-implemented property. For more information, see [Auto-Implemented Properties](#).

You can define properties in classes, structures, and modules. Properties are Public by default, which means you can call them from anywhere in your application that can access the property's container.

For a comparison of properties and variables, see [Differences Between Properties and Variables in Visual Basic](#).

#### **4. What are Procedures? What is a Function Procedure and a Property Procedure?**

A Function procedure is a series of Visual Basic statements enclosed by the Function and End Function statements. The Function procedure performs a task and then returns control to the calling code. When it returns control, it also returns a value to the calling code.

Each time the procedure is called, its statements run, starting with the first executable statement after the Function statement and ending with the first End Function, Exit Function, or Return statement encountered.

You can define a Function procedure in a module, class, or structure. It is Public by default, which means you can call it from anywhere in your application that has access to the module, class, or structure in which you defined it.

A Function procedure can take arguments, such as constants, variables, or expressions, which are passed to it by the calling code.

##### **Declaration syntax**

The syntax for declaring a Function procedure is as follows:

VBCopy

```
[Modifiers] Function FunctionName [(ParameterList)] As ReturnType
```

```
    [Statements]
```

```
End Function
```

The *modifiers* can specify access level and information regarding overloading, overriding, sharing, and shadowing. For more information, see [Function Statement](#).

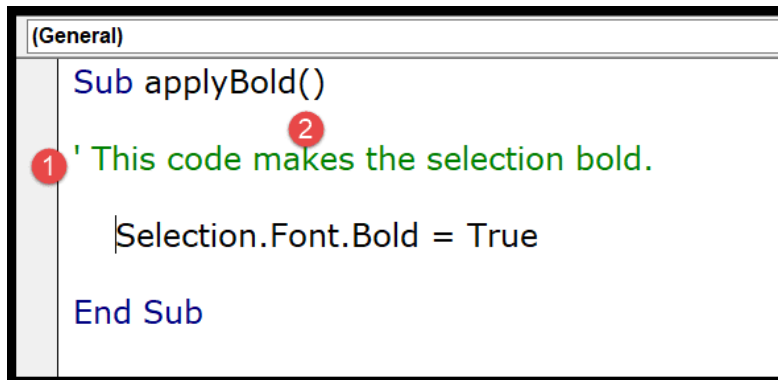
You declare each parameter the same way you do for [Sub Procedures](#).

#### **5. What is a sub procedure and what are all the parts of a sub procedure and when are they used?**

A Sub procedure is **a series of Visual Basic statements enclosed by the Sub and End Sub statements**. The Sub procedure performs a task and then returns control to the calling code, but it does not return a value to the calling code

## 6. How do you add comments in a VBA code? How do you add multiple lines of comments in a VBA code?

1. First, **click on the line** where you want to insert the comment.
2. After that, **type an APOSTROPHE** using your keyboard key.
3. Next, **type the comment** that you want to add to the code.
4. In the end, **hit enter** to move to the new line and the comment will turn green.



The screenshot shows a VBA code editor window with a tab labeled "(General)". The code contains a Sub procedure named "applyBold()". The first line of the procedure is "Sub applyBold()". The second line is a comment: "' This code makes the selection bold." The comment is highlighted in green. A red circle with the number "1" is placed at the start of the comment line, and a red circle with the number "2" is placed at the end of the comment line. The third line of the procedure is "Selection.Font.Bold = True". The fourth line is "End Sub".

```
Sub applyBold()  
' This code makes the selection bold.  
    Selection.Font.Bold = True  
End Sub
```

The moment you do this the entire line of the code will turn green which means that line is comment now.

If you look at the below code where I have used a comment to add a description of the procedure.

So you simply need to add an apostrophe before turning it into a comment and VBA will ignore it while executing the code.