## What is Visual Basic for Applications (VBA)?

## ****Visual Basic for Applications (VBA)**** is an event-driven programming language implemented by Microsoft to develop Office applications. VBA helps to develop automation processes, Windows API, and user-defined functions. It also enables you to manipulate the user interface features of the host applications.

Before we go into further details, let’s look at what computer programming is in a layman’s language. Assume you have a maid. If you want the maid to clean the house and do the laundry. You tell her what to do using let’s say English and she does the work for you. As you work with a computer, you will want to perform certain tasks. Just like you told the maid to do the house chores, you can also tell the computer to do the tasks for you.

The process of telling the computer what you want it to do for you is what is known as computer programming. Just as you used English to tell the maid what to do, you can also use English-like statements to tell the computer what to do. The English-like statements fall in the category of high-level languages. VBA is a high-level language that you can use to bend excel to your all-powerful will.

VBA is actually a subset of Visual Basic 6.0 BASIC stands for **B**eginners **A**ll-Purpose **S**ymbolic **I**nstruction **C**ode.

## Why VBA?

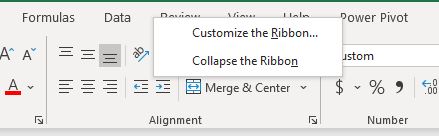
VBA enables you to use English-like statements to write instructions for creating various applications. VBA is easy to learn, and it has easy to use User Interface in which you just have to drag and drop the interface controls. It also allows you to enhance Excel functionality by making it behave the way you want.

.

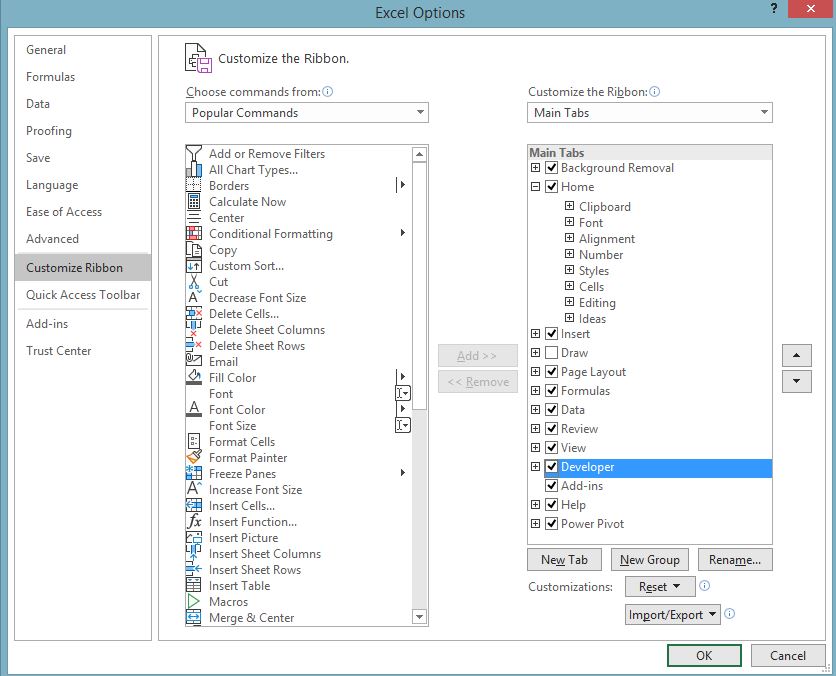
Enable Developer Option in Excel

It hides the Developer tab on the ribbon by default. To customize the ribbon, follow the steps mentioned below:

* Right-click on the ribbon (anywhere) click on the Customize the Ribbon option.

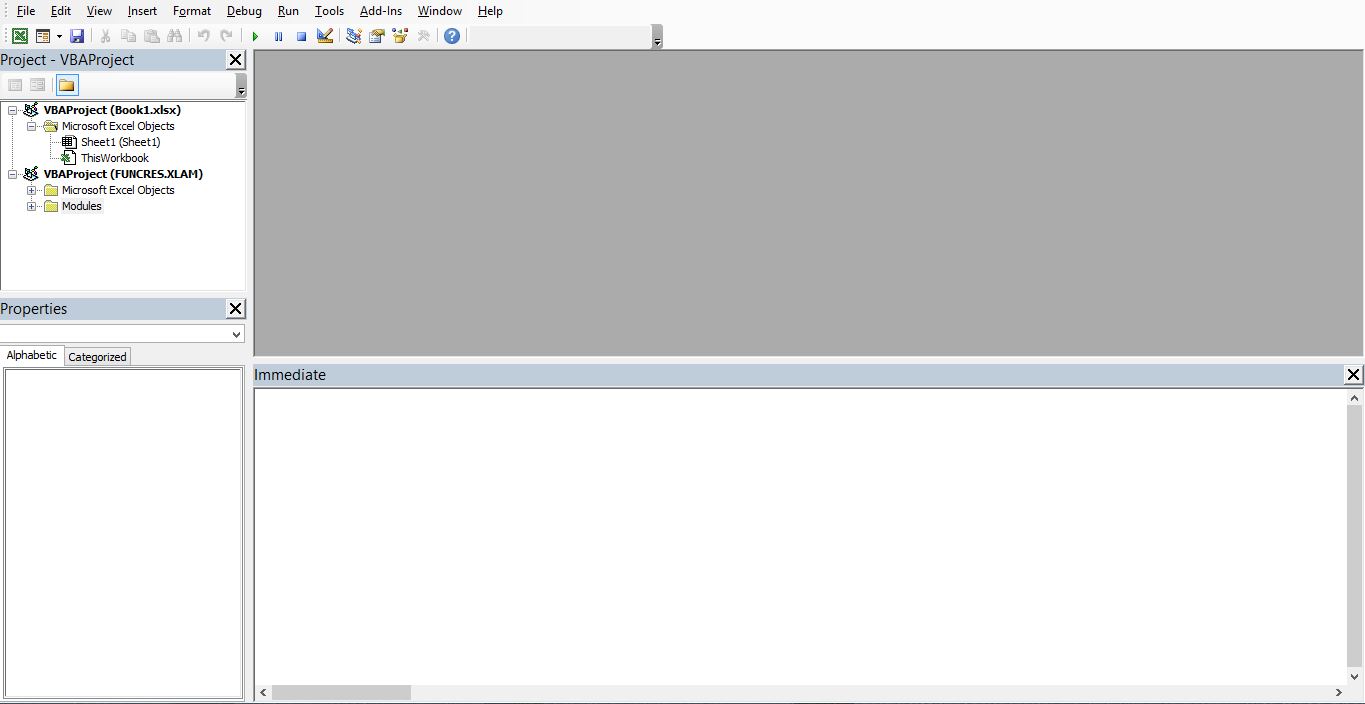


* Go to Customize the Ribbon and select the Developer checkbox.



VBA Editor Interface

You can open the VBA interface by using the ALT + F11 keyboard shortcut, or you can go to the Developer tab and click on Visual Basic.



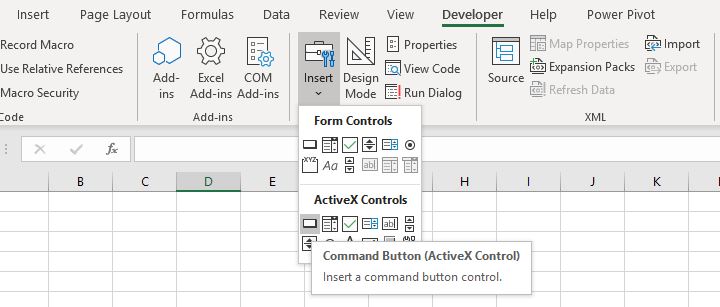
Create an Excel Macro using a Command Button

**Create a Command Button**

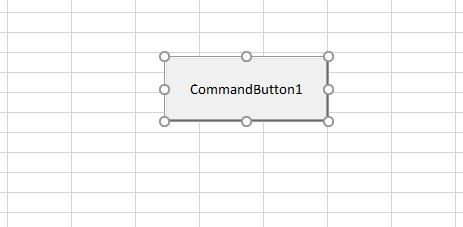
Now that you have enabled the developer tab and have some familiarity with the VBA editor, let’s start creating a macro by using a command button.

To place a command button on your worksheet, follow these steps:

* Go the Developer tab > Insert > ActiveX Controls > Command button.



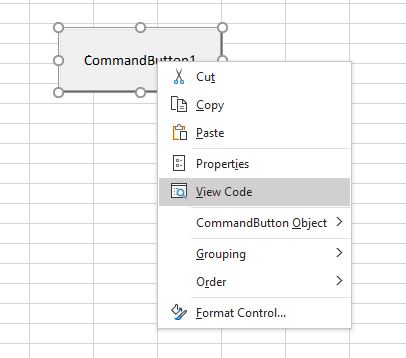
* Drag the command button on your worksheet.



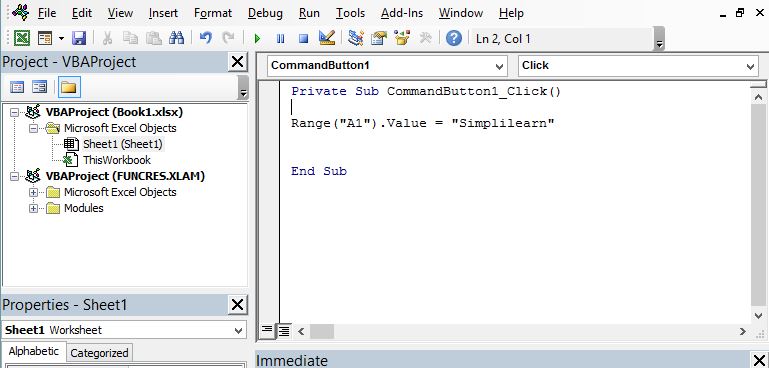
**Assigning a Macro to a Command Button**

To assign a macro to the command button, follow these steps:

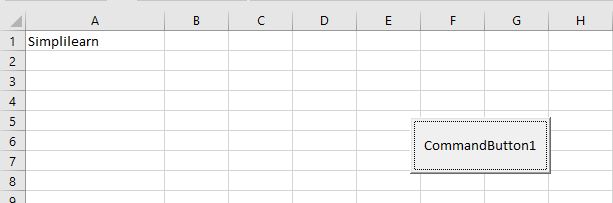
* Right-click on the command buttons and select View Code.



* Add the following lines of code shown below.

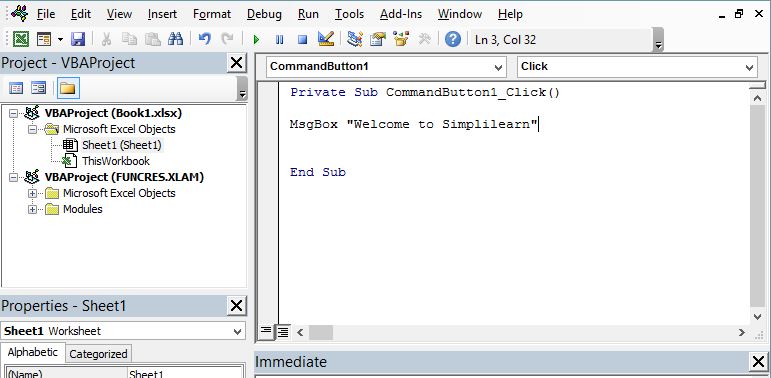


* Close the VBA editor and click on the command button on the worksheet. Make sure to deselect the design mode.

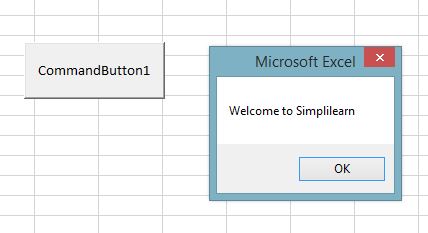


**Create a Message Box**

The MsgBox is a dialog box in Excel that will prompt a message on your worksheet. To create a MsgBox, add the following lines of code to the command button.



Now, when you click on the button, you will get the following message.



Create an Input Box

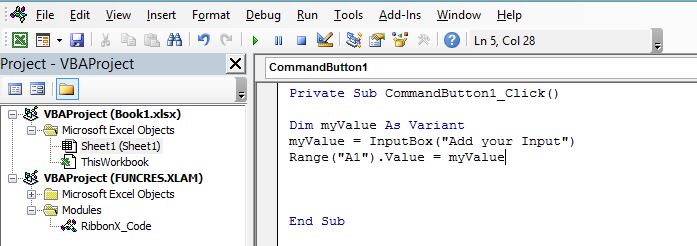
The InputBox function prompts the user to enter the values and returns the information entered in the dialog box.

Syntax:

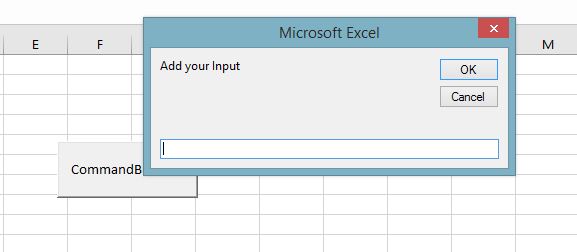
InputBox(prompt[,title][,default][,xpos][,ypos][,helpfile,context])

To add the Input Box to your command button, execute the following steps:

* Declare the variable name and keep the type variant. A variant variable can hold any type of value.
* Add the following code to show the input box.



* Now when you click on the command button, you will get a prompt asking for a value.



* Enter your input and click OK. It will enter your input on cell A1.

**Variables, Constant, and Operators in VBA**

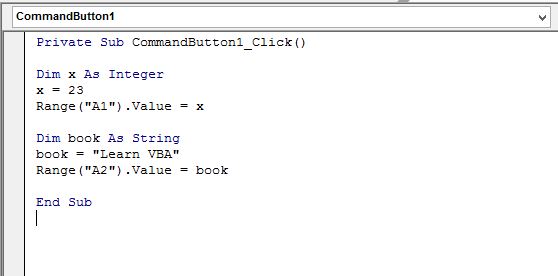
Variables

Variables are the data types that are used to hold a value. We can change the variables during the execution of the program.

Syntax: Dim <<variable\_name>> As <<variable\_type>>

We can divide the VBA data types into two categories:

* Numeric Data Types: Numeric data types consist of byte, integer, long, single, double, currency, and decimal.
* Non-Numeric Data Types: Non-numeric data types consist of string, date, Boolean, object, and variant.



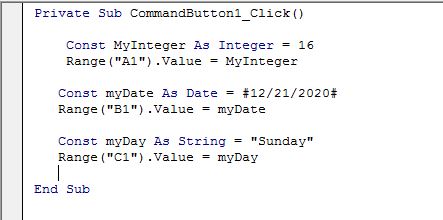
Constants

Constants are fixed value that cannot be changed during the execution of the program.

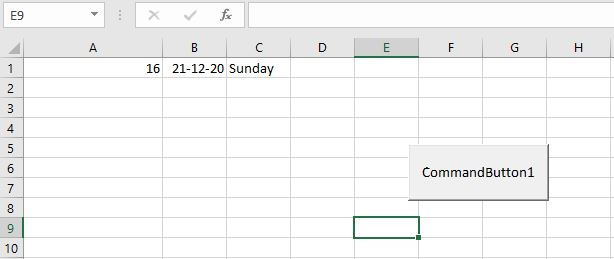
Syntax:

Const <<constant\_name>> As <<constant\_type>> = <<constant\_value>>

Example:



When you click on the command button, you will get the following output:

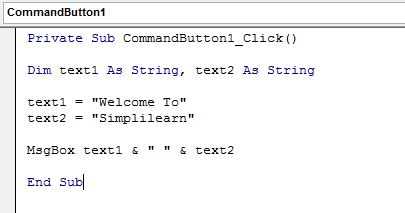


String Manipulation

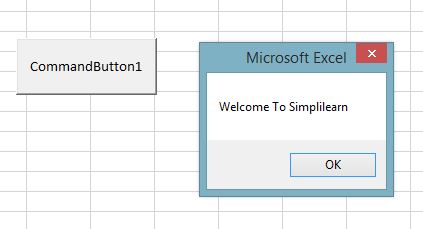
String manipulation refers to the process of analysing, manipulating, and effectively handling string values.

Join Strings

You can join the two strings by using & operator. This is also known as concatenation.

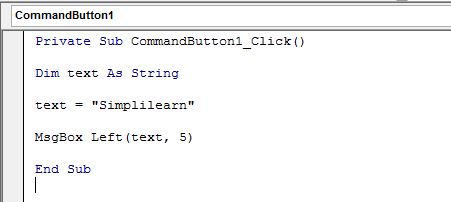


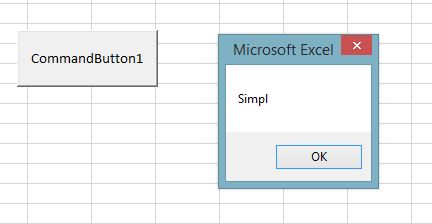
Click on the command button to get the following result.



Left

The left keyword will extract the characters from the leftmost side of the string.





### VBA Hello World!

Now we will demonstrate how to program in VBA programming language. All program in VBA has to start with “Sub” and end with “End sub”. Here the name is the name you want to assign to your program. While sub stands for a subroutine which we will learn in the later part of the tutorial.

Sub name()

.

.

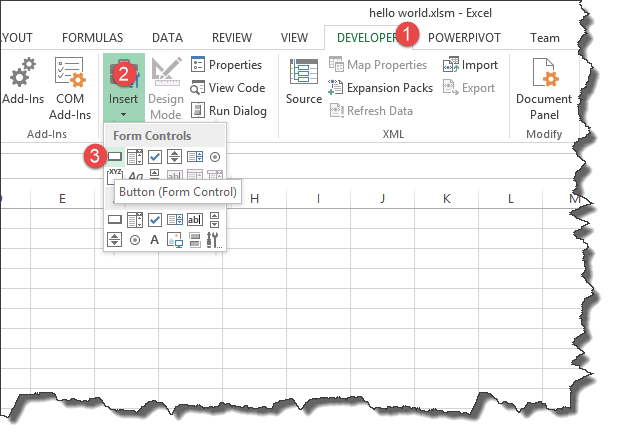
.

End Sub

We will create a basic VBA program that displays an input box to ask for the user’s name and then displays a greeting message

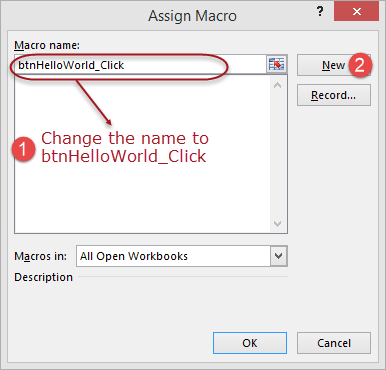
This tutorial assumes you have completed the tutorial on Macros in excel and have enabled the DEVELOPER tab in excel.

* Create a new workbook
* Save it in an excel macro enabled worksheet format \*.xlsm
* Click on the DEVELOPER tab
* Click on the INSERT drop-down box under the controls ribbon bar
* Select a command button as shown in the image below

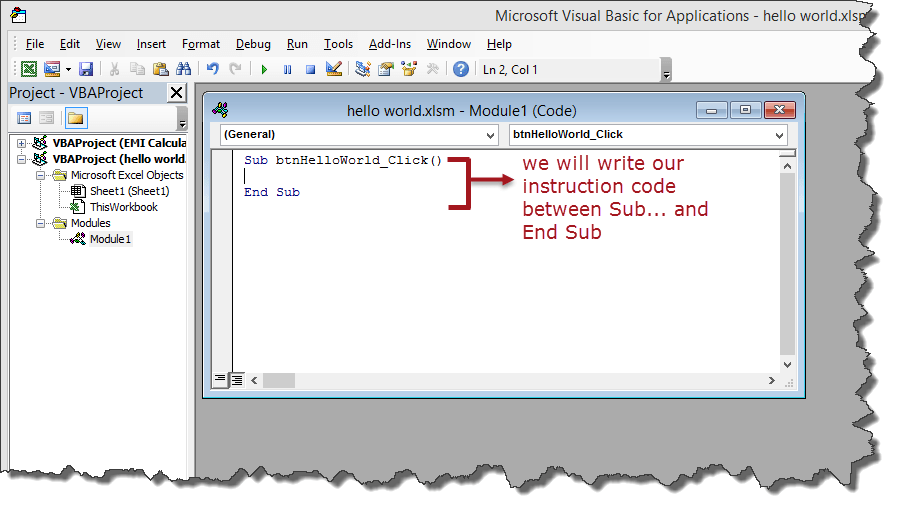


Draw the command button anywhere on the worksheet

You will get the following dialogue window



* Rename the macro name to btnHelloWorld\_Click
* Click on new button
* You will get the following VBA code window



Enter the following instruction codes

**Dim name As String**

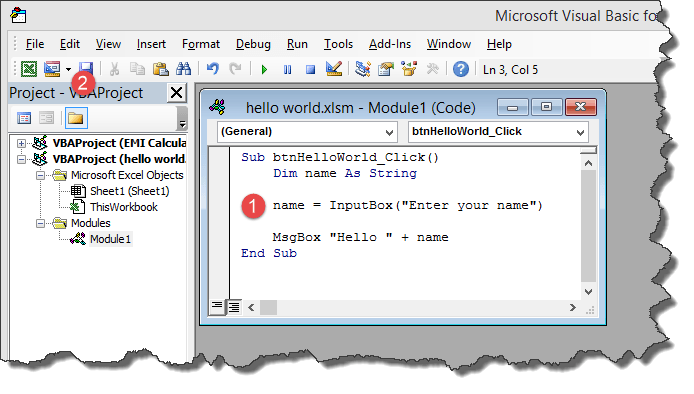
**name = InputBox("Enter your name")**

**MsgBox "Hello " + name**

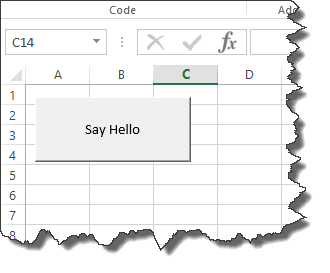
HERE,

* **“Dim name as String”** creates a variable called name. The variable will accept text, numeric and other characters because we defined it as a string
* **“name = InputBox(“Enter your name”)”** calls the built-in function InputBox that displays a window with the caption Enter your name. The entered name is then stored in the name variable.
* “**MsgBox “Hello ” + name”** calls the built-in function MsgBox that displays Hello and the entered name.

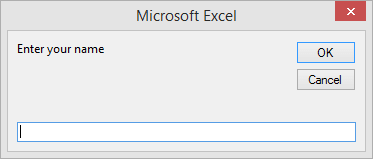
Your complete code window should now look as follows



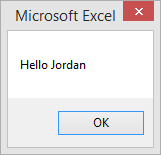
* Close the code window
* Right click on button 1 and select edit text
* Enter Say hello



* Click on Say Hello
* You will get the following input box



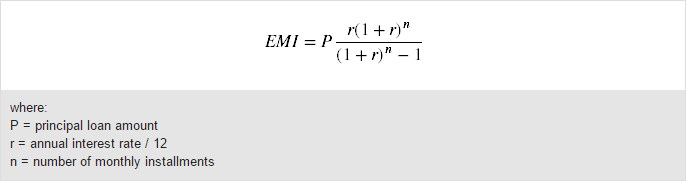
* Enter your name i.e. Jordan
* You will get the following message box



Congratulations, you just created you have created VBA program in excel

## Step-by-step example of creating a simple EMI calculator in Excel

In this tutorial exercise, we are going to create a simple program that calculates the EMI. EMI is the acronym for Equated Monthly Instalment. It’s the monthly amount that you repay when you get a loan. The following image shows the formula for calculating EMI.



The above formula is complex and can be written in excel. The good news is excel already took care of the above problem. You can use the PMT function to compute the above.

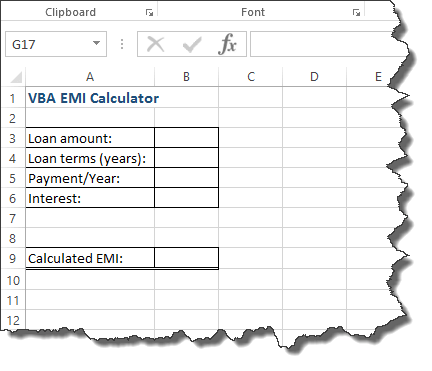
The PMT function works as follows

=PMT(rate,nper,pv)

**HERE,**

* **“rate”** this is the monthly rate. It’s the interest rate divided by the number of payments per year
* **“nper”** it is the total number of payments. It’s the loan term multiplied by number of payments per year
* **“pv”** present value. It’s the actual loan amount

Create the GUI using excel cells as shown below



Add a command button between rows 7 and 8

Give the button macro name btnCalculateEMI\_Click

Click on the edit button

Enter the following code

Dim monthly\_rate As Single, loan\_amount As Double, number\_of\_periods As Single, emi As Double

monthly\_rate = Range("B6").Value / Range("B5").Value

loan\_amount = Range("B3").Value

number\_of\_periods = Range("B4").Value \* Range("B5").Value

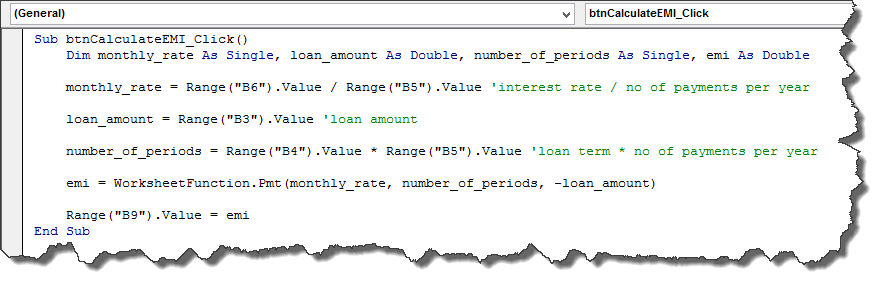
emi = WorksheetFunction.Pmt(monthly\_rate, number\_of\_periods, -loan\_amount)

Range("B9").Value = emi

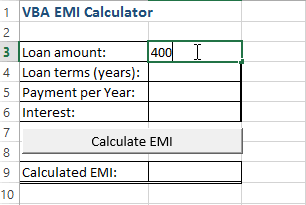
**HERE,**

* **“Dim monthly\_rate As Single,…”** Dim is the keyword that is used to define variables in VBA, monthly\_rate is the variable name, and Single is the data type which means the variable will accept numbers.
* **“monthly\_rate = Range(“B6”).Value / Range(“B5″).Value”** Range is the function used to access excel cells from VBA, Range(“B6”). Value makes reference to the value in B6
* **“WorksheetFunction.Pmt(…)”** WorksheetFunction is the function used to access all the functions in excel

The following image shows the complete source code



* Click on save and close the code window
* Test your program as shown in the animated image below

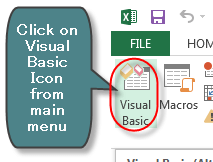


**How to use VBA in Excel Example**

Following steps will explain how to use VBA in Excel.

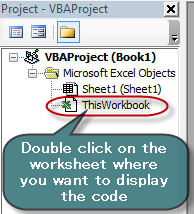
**Step 1)** Open your VBA editor

Under Developer tab from the main menu, click on “Visual Basic” icon it will open your VBA editor.

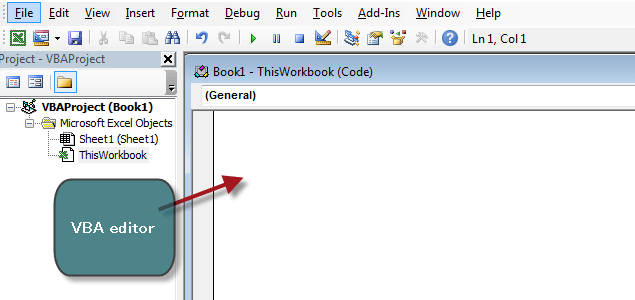


**Step 2)** Select the Excel sheet & Double click on the worksheet

It will open a VBA editor, from where you can select the Excel sheet where you want to run the code. To open VBA editor double click on the worksheet.



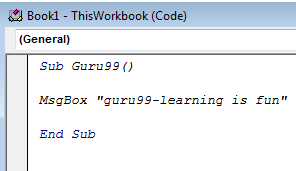
It will open a VBA editor on the right-hand side of the folder. It will appear like a white space.



**Step 3)** Write anything you want to display in the MsgBox

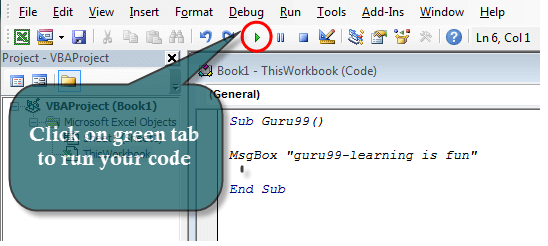
In this step we are going to see our first VBA program. To read and display our program we need an object. In VBA that object or medium in a MsgBox.

* First, write “Sub” and then your “program name” (Guru99)
* Write anything you want to display in the MsgBox (guru99-learning is fun)
* End the program by End Sub



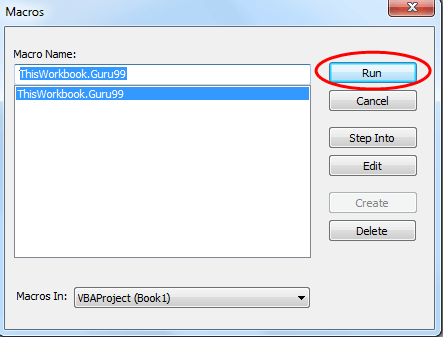
**Step 4)** Click on the green run button on top of the editor

In next step you have to run this code by clicking on the green run button on top of the editor menu.



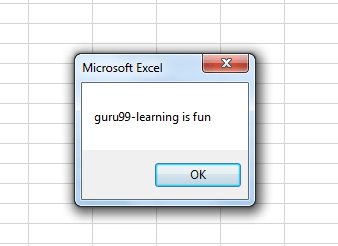
**Step 5)** Select the sheet and click on “Run” button

When you run the code, another window will pops out. Here you have to select the sheet where you want to display the program and click on “Run” button



**Step 6)** Display the msg in MsgBox

When you click on Run button, the program will get executed. It will display the msg in MsgBox.



**If, If-Else, For, and While Loop**

If statement

The ‘If statement’ is a conditional statement that consists of an expression followed by another expression. If the condition is true, the lines of code under the If statement are executed.

Syntax:

If(boolean\_expression) Then

   Statement 1

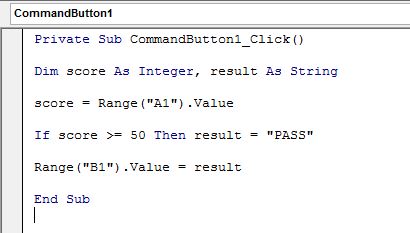
   .....

   .....

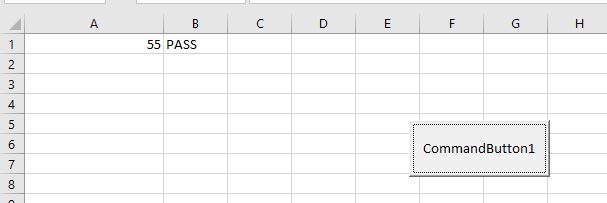
   Statement n

End If

Example:

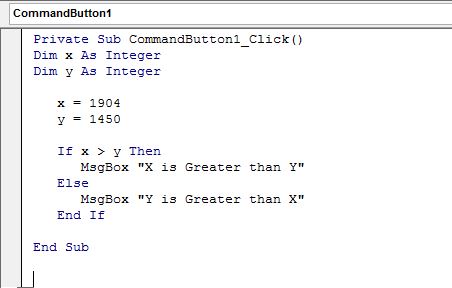


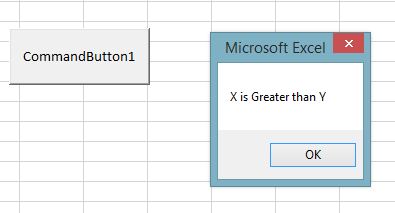
Enter the marks in cell A1 and click on the command button to get the result.



If-Else

The ‘If statement’ is a conditional statement that consists of an expression followed by another expression. If the condition is true, the lines under the body of the If statement are executed. If it says the condition to be False, it executes the statements under the Else Part.

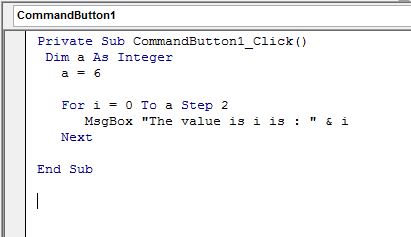




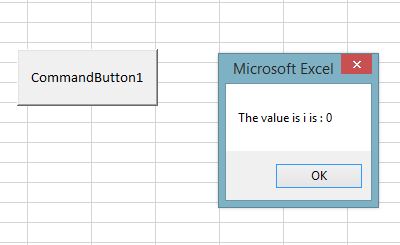
For Loop

For loop is a control flow statement that allows the user to write a loop that can be executed repeatedly.

Example:



The following code will prompt the message box showing the value from 0 to 6 with a gap of 2.



**While Loop**

In a while loop, if the statements are true, they are executed till they encounter the Wend keyword. If the statement is false, the loop is exited, and it jumps to the next statement.

Syntax:

While condition(s)

   [statements 1]

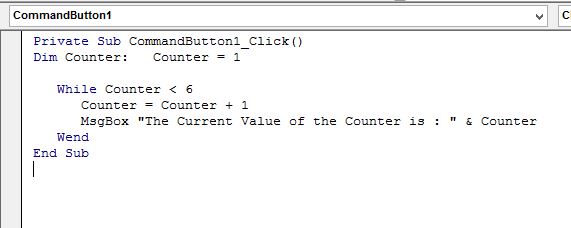
   [statements 2]

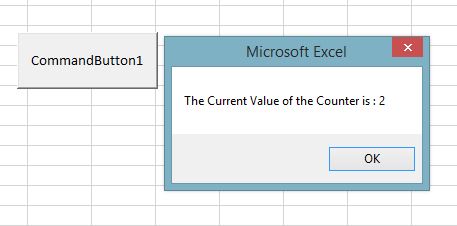
   ...

   [statements n]

Wend

Example:



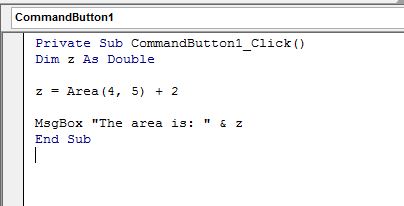


**Functions and Sub Procedures**

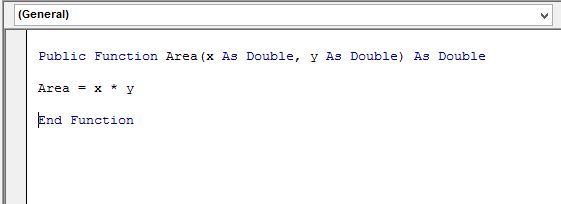
Functions

Functions are reusable code that can be called anywhere in the program. You can use the code repeatedly in your program. To create the function in the VBA window, go to Insert > Module and it will create a new module.

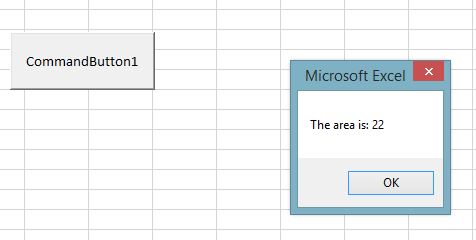
The following code shows an example of a simple VBA function procedure that receives two arguments.



The below-given code calls the function that was defined above:



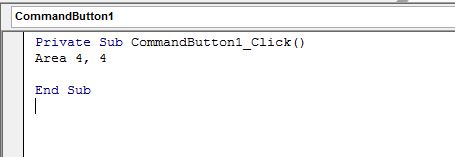
Result:



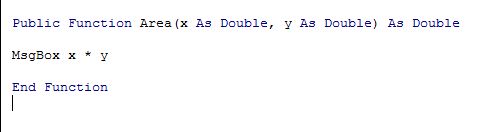
Sub-Procedure

Sub-procedures are similar to functions with some minor differences. The sub-procedure does not return a value and can be called without a call keyword.

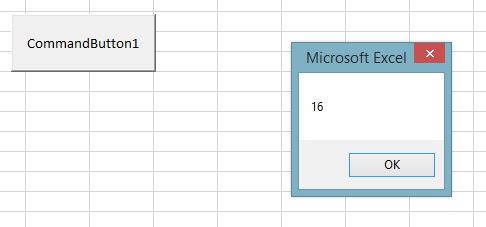
The following code shows an example of a simple VBA Sub procedure that will calculate the area.



The above example illustrates how sub-procedures perform actions but do not return values.



Result:

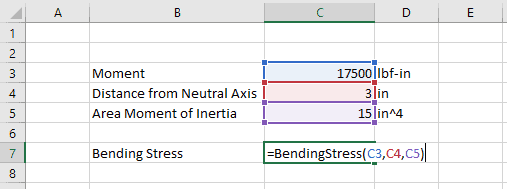


# What’s an Excel User Defined Function?

# A User Defined Function is a procedure (a group of commands) written in VBA that (usually) accepts inputs and returns a result. A UDF cannot modify the formatting of a cell or workbook or move values around on a worksheet.

# Basically, UDFs enable you to create custom functions that act very similarly to the built-in functions that are included in every installation of Excel, such as SQRT, SUM, and MAX.

For instance, below I’ve called a custom UDF that I created called “BendingStress” in a cell to calculate the bending stress in a beam. The arguments to the function are [moment](https://engineerexcel.com/beam-to-beam-moment-connections/), distance from the neutral axis, and moment of inertia.



# Why use a Function?

There are a few different reasons why you might want to consider creating a custom User Defined Function in your worksheet.

The first advantage of functions is that they can clean up your spreadsheets. Instead of the cell after cell of sequential calculations, you can combine many successive calculations into a single function. This can significantly clean up your spreadsheets.

User Defined Functions can increase your productivity by allowing you to store and re-use calculations that you use over and over again. Rather than having to recall an equation from memory or looking it up in a reference, you can build it into a UDF and call the UDF instead of retyping the calculation. This has the added benefit of minimizing typing errors.

Finally, User Defined Functions provide you with all of the flexibility of the Visual Basic for Applications language. With UDF you can take advantage of loops, expanded logic, and other functionality. This is where you can really increase the capability of your engineering spreadsheets.

# How to Create a Function

Let’s walk through creating a User Defined Function. We’ll use the BendingStress function from above as an example.

With the Developer tab enabled, open the Visual Basic Editor by pressing Alt+F11 or by clicking Visual Basic on the far left side of the developer tab.

A picture containing graphical user interface

Description automatically generated

Next, insert a new module in the current workbook by right-clicking on “VBAProject (YourWorkbookName), then selecting Insert>Module.

Graphical user interface, text, application

Description automatically generated

In the Code Window (the biggest window – in the upper right of the screen) start by typing the following:



When you press enter, the Visual Basic Editor automatically adds “End Function” and you end up with this:



The space between “Function” and “End Function” is where we will write our User-Defined Function.

Type “BendingStress” again, an equal sign, and the formula:

Chart

Description automatically generated with medium confidence

That’s it. You just created a User Defined Function. Let’s go back to the worksheet and try to use it.

Graphical user interface, application, table, Excel

Description automatically generated

As soon as you enter the equal sign and the first few letters of the function into a cell, Excel suggests the function. Pressing the Tab key auto-fills the name of the function into the cell.

From there, we can enter the three arguments required by the function as either references to cells containing the values or as static values:

Graphical user interface, application

Description automatically generated with medium confidence

After pressing “Enter”, the UDF returns the result of the calculation to the cell:

Table

Description automatically generated