Introduction

Excel is pronounced "Eks - sel"

It is a spreadsheet program developed by Microsoft. Excel organizes data in columns and rows and allows you to do mathematical functions. It runs on Windows, macOS, Android and iOS.

The first version was released in 1985 and has gone through several changes over the years. However, the main functionality mostly remains the same.

Excel is typically used for:

Analysis

Data entry

Data management

Accounting

Budgeting

Data analysis

Visuals and graphs

Programming

Financial modeling

And much, much more!

Why Use Excel?

It is the most popular spreadsheet program in the world

It is easy to learn and to get started.

The skill ceiling is high, which means that you can do more advanced things as you become better

It can be used with both work and in everyday life, such as to create a family budget

It has a huge community support

It is continuously supported by Microsoft

Templates and frameworks can be reused by yourself and others, lowering creation costs

Introduction programming

Programming refers to writing a set of instructions that tell Excel to perform one or more tasks.

These instructions are written in the Visual Basic for Applications (VBA) as this is the language understandable to Excel.

To write an instruction, one can either write a code in VBA or record a macro in Excel. A macro is recorded to automate repetitive tasks.

When a macro is recorded, VBA generates a code in the background.

For example, to submit an Excel report, a table needs to undergo the same set of tasks each time.

These tasks include applying a pre-defined border, color, alignment, and font. To program in Excel, a macro can be recorded for performing

all these tasks.

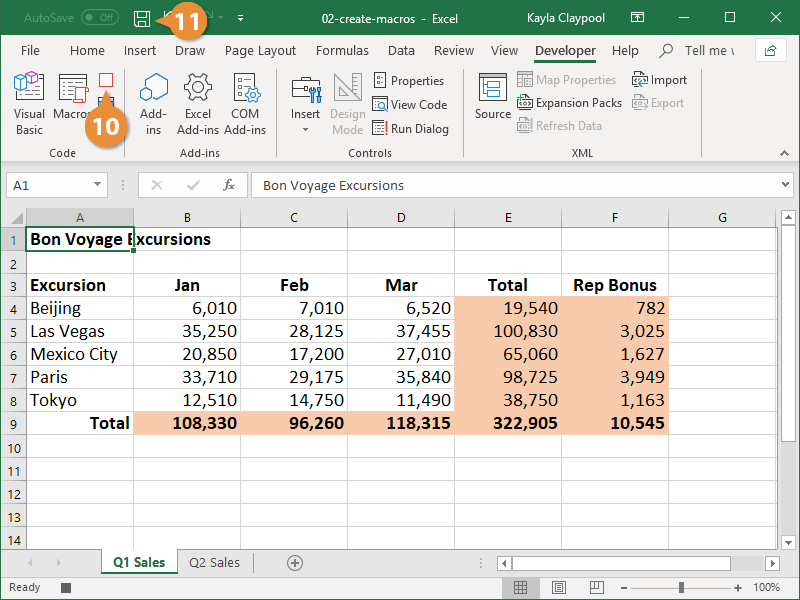
The purpose of programming in Excel is to save the user from performing the same tasks again and again.

Moreover, it helps accomplish multiple tasks at a great speed that would have taken a lot of time, had they been performed manually.

VBA Overview and the IDE

VBA stands for Visual Basic for Applications an event-driven programming language from Microsoft that is now predominantly used with Microsoft office applications such as MSExcel, MS-Word, and MS-Access. It helps techies to build customized applications and solutions to enhance the capabilities of those applications.

Quick Review of Macros Working with Macros



**How to Use the VBA Editor in Excel: Quick Guide 2023**

**Read**

In Excel VBA stands for (Visual Basic for Application Code) where we can automate our task with the help of codes and codes that will manipulate(like inserting, creating, or deleting a row, column, or graph) the data in a worksheet or workbook. With the help of VBA, we can also automate the task in Excel to perform all these tasks we need to insert and run the VBA code properly which will be discussed here.

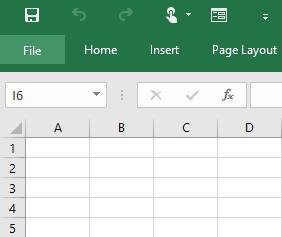
**What is Visual Basic Editor in Excel**

Visual Basic for Application is a programming language that allows you to automate tasks, create custom functions, and build interactive applications within Excel. VBA is an **interface for creating scripts.**If you have done any programming in the integrated development environment(IDE), the VBA editor in Excel looks similar. It lets you create, manage, and run VBA code on your Excel Spreadsheet.

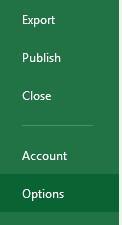
**How to Enable the Developer Tab in Excel**

To get the Visual Basic Editor in Excel, we need to get the Developer tab. For that, we have to follow the following steps:

**Step 1: Click on the File menu at the left top of the Excel tab**

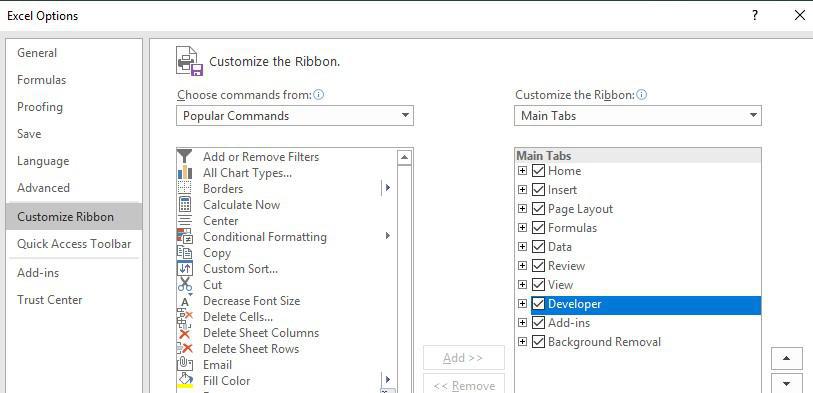


**Step 2: Select Options to get the Excel Options window**



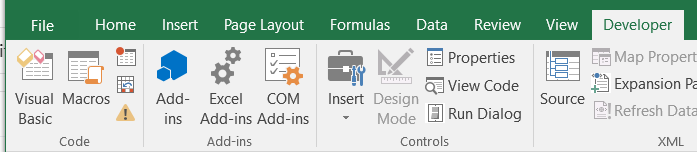
**Step 3: Add Developer Tab**

Select “Customized Ribbon” in the “Excel Options” Window and then select the “Developer” check box in the “Main Tabs”.



**Step 4: Preview Developer Tab**

In the below image, you can see the Developer tab in Ribbon.

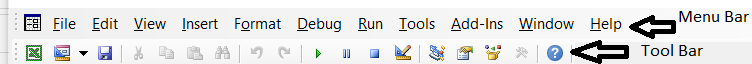


**Visual Basic Editor User Interface**

The VBA interface is a User-friendly interface with various components that helps you to work with VBA code efficiently. Below are some elements:

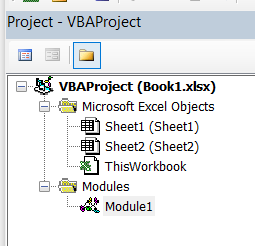
**Toolbar and Menu Bar**

VBA Editor consists of a standard toolbar as well as a Menubar same as other Windows applications. The toolbar provides quick access to common actions, such as running or stopping the code from being executed.



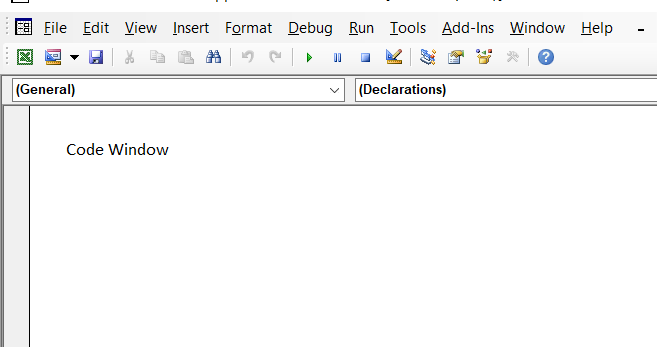
**Project Explorer**

You can find the Project Explorer window in the left pane of the VBA editor. This will display a hierarchical view of all the open workbooks and their components in the current Excel Session.



**Code Window**

Code Window is the central part of the VBA Editor. Here you can write, edit, and view your VBA code. This window is specific to the module in the Project Explorer.



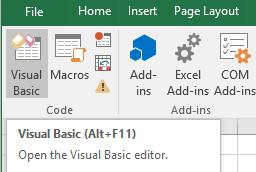
**Immediate Window**

The immediate window is located at the bottom of the VBA Editor. It is used to execute single lines of code and view the results in real time. This will help you in testing and debugging.

**How to Open Visual Basic Editor in Excel**

**Step 1: Select Developer Tab in Excel and Click on Visual Basic**

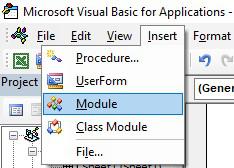
Select the “**Developer”** ribbon and then click on Visual Basic.



***Shortcut Key:****Press****“Alt+F11”****to get the VBA editor.*

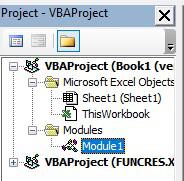
**Step 2: Click on Insert tab and Select Insert**

To insert a module in the VBA editor select the **Insert** tab and then click on Module.



**Module**

**Module 1**will be added to the Modules of **Project- VBAProject.**



***Note****: If the Modules folder does not exist in the VBAProject, it will be automatically created, and a new module will be added inside it. In this designated folder you can store your Excel VBA code after you are prepared to write it.*

**How to Open Visual Basic Editor in Excel**

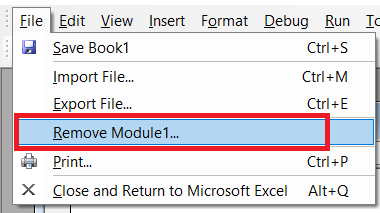
The Excel VBA Editor serves as the central hub within any Microsoft Office application for handling all automation-related coding tasks. It offers versatile functionality:

* **Macro Creation:** It can be utilized for crafting macros, achieved through code composition within the VBA Module or Sheet, or by recording macros.
* **File Integration:** The written code can also be employed to establish connections between various files of diverse extensions.”

**How to Delete a Module in VBA Excel**

**Step 1: Right Click on the left pane and Select Remove Module**

To delete a module, right-click it on the left pane and choose the option “Remove[module name]”.

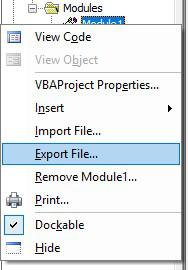


**Read More:**

[How to Delete a Module in Excel VBA?](https://www.geeksforgeeks.org/how-to-delete-a-module-in-excel-vba/)

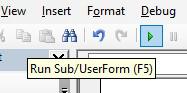
**How to Export a VBA module from Excel**

If we want to save the module on the computer then we can right-click on Module1 to select **Export File.**



**How to Run Macro in VBA Excel**

**Step 1:**Click on**Run Macro**to run the VBA code.



***Shortcut Key to Run VBA in Excel:****Press****F5****Key*

**How to Debug a Macro in Excel**

While working with VBA code, you may face errors or unexpected behavior. VBA editor consists of several tools to [debug your code](https://www.geeksforgeeks.org/debugging-vba-code-in-excel/) and also handle them effectively. A few of them are listed below:

**Breakpoints**

Users can set breakpoints in their code to pause execution at specific lines. Setting breakpoints in your code will closely examine variable values and step through the code line by line. This will help in effective debugging, as the user can observe how the program is behaving at critical points and can identify any issues.

**Immediate Window**

Immediate Window is a feature that can be used for testing purposes and it also interacts with your code in real time. Users can print values, executes every single line of code, and check for errors.

[Error Handling](https://www.geeksforgeeks.org/excel-vba-error-handling/)

It is very important to write error-handling routines in your VBA code to manage unexpected errors gracefully. If something goes wrong during the execution, your code can handle it and continue running smoothly.

**How to Open VBA editor on Microsoft Excel for Mac**

* Use keyboard shortcuts to open the VBA Editor: On Windows, press Alt + F11, and on Mac, press Opt + F11 or Fn + Opt + F11.
* It works even without the developer tab on the ribbon.

**FAQs**

***What is VBA Editor in Excel?***

*The VBA Editor (Virtual Basic for Application Editor) is similar to integrated development in Microsoft Excel that allows users to write, edit, and also manage VBA code. It can be used to create macros, automate tasks, and build functions and applications.*

***How to access the VBA Editor in Excel?***

***Follow the below steps to access the VBA editor:***

***Step 1:****Enable the Developer tab in Excel.*

***Step 2:****Click on the****“Developer”****tab in the Ribbon.*

***Step 3:****Now choose “Visual Basic” to open the VBA Editor.*

***How to run VBA code in Excel?***

*There are many ways by which you can run VBA code in Excel:*

* *Using the****“Run”****button in the VBA editor to execute the entire macro.*
* *Press the combination of “Alt + F8” to open the “Macro” dialog and choose the macro to run.*
* *Assign the VBA code to a button or from the control on the worksheet and click it to run the code.*

***How to Enable VBA Editor in Excel ?***

*To access the VBA Editor, simply press Alt + F11. This straightforward shortcut is convenient for regular VBA coding tasks, allowing you to switch between Excel and the VBA Editor seamlessly.*

***How to handle errors in VBA code?***

[*Error handling*](https://www.geeksforgeeks.org/excel-vba-error-handling/)*is the way to manage unexpected errors in VBA code. You can also use “*[*On Error*](https://www.geeksforgeeks.org/excel-vba-error-handling/#:~:text=Error%20Handling%20Statements-,On%20Error,-It%20is%20used)*” statements to handle errors gracefully.*

Whether you're preparing for your first job interview or aiming to upskill in this ever-evolving tech landscape, [GeeksforGeeks Courses](https://www.geeksforgeeks.org/courses?utm_source=geeksforgeeks&utm_medium=article_bottom_text&utm_campaign=courses" \t "_blank) are your key to success. We provide top-quality content at affordable prices, all geared towards accelerating your growth in a time-bound manner. Join the millions we've already empowered, and we're here to do the same for you. Don't miss out - [check it out now](https://www.geeksforgeeks.org/courses?utm_source=geeksforgeeks&utm_medium=article_bottom_text&utm_campaign=courses)!

**Advantages of VBA**

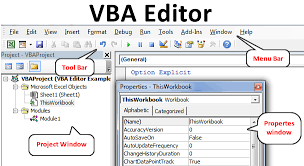
* An easy-to-use and versatile language

Python is a versatile language that you can use to automate many Excel tasks. You can also use various libraries such as Pandas, openpyxl, xlwings, and pyautogui to manipulate data, extract information, generate reports, and automate repetitive tasks.

* Integrated with off-the-shelf products

COTS (Commercial Off-The-Shelf) products and tools are pre-built software components that are available in the market for purchase or licensing. They are designed to perform specific tasks and are readily available for integration into existing software applications or systems.

**Understanding the VBA Editor, Module and Procedures**

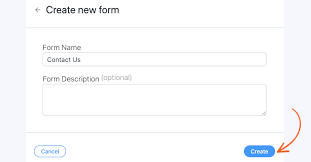
* The Visual Basic Editor
* 
* Parts of the VBA Editor
* Component #1: Menu Bar.
* Component #2: Toolbar.
* Component #3: Project Window / Project Explorer.
* Component #4: Properties Window.
* Component #5: Programming Window / Code Window / Module Window.
* Component # 6: Immediate Window.
* Modules and Macros

macros are used to automate tasks in Excel, while modules are used to organize and store code. Macros are the way to utilize the code written in a module, and the modules are the way to organize and store that code for later use

* About VBA Forms, Sub Procedures & Modules

A sub procedure can take two forms, i.e., private and public. The modifiers “private” and “public” allows users to use the subs differently. The private sub procedure can only be used in the current module. The public sub allows users to use the procedure in all modules present in the workbook.

* Creating Forms



* Using Controls and their Properties

Controls are used to get user input and to display output. E.g., text box, label, check box, list box, combo box, are all controls. Properties of an object are the characteristics used to describe the object.

Running VBA Forms in Excel

In this chapter, you will learn to design a simple form and add data into excel. Step 1 − Navigate to VBA Window by pressing Alt+F11 and Navigate to "Insert" Menu and select "User Form". Upon selecting, the user form is displayed as shown in the following screenshot. Step 2 − Design the forms using the given controls.

**VBA Syntax and Grammar**

* Objects – The Grammar

An object is one of the five major elements of clause structure. The other four are subject, verb, adjunct and complement. Objects are typically noun phrases (a noun or pronoun and any dependent words before or after it). Objects normally follow the verb in a clause: Everyone likes her.

* Using Properties and Methods

Using a method causes something to happen to an object, while using a property returns information about the object or causes a quality about the object to change.

* Understanding Parameters

A parameter is a special kind of variable used in a function to refer to one of the pieces of data provided as input to the function. These pieces of data are the values of the arguments with which the function is going to be called/invoked.

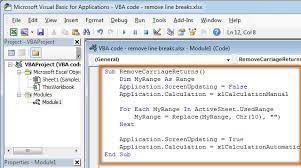
What is the Object Browser?

The Object Browser allows you to browse through all available objects in your project and see their properties, methods and events. In addition, you can see the procedures and constants that are available from object libraries in your project. You can easily display online Help as you browse

* Understanding object hierarchy

An object hierarchy is a concept from computer programming. It references descendants of objects acting as properties of an object. An example of this would be the object controlling a window (at the top of the hierarchy) having another object like the window's border acting as a property of the window.

* Tools for better coding



* Moving to Other Cells

You can move cells in Excel by drag and dropping or using the Cut and Paste commands. Select the cells or range of cells that you want to move or copy. Point to the border of the selection. When the pointer becomes a move pointer , drag the cell or range of cells to another location.

* Editing Specific Cells
* Double-click the cell that contains the data that you want to edit. ...
* Click the cell that contains the data that you want to edit, and then click anywhere in the formula bar. ...
* Click the cell that contains the data that you want to edit, and then press F2.
* **Storing Data in Variables**
* How and when to declare variables

Declaration of a variable in a computer programming language is a statement used to specify the variable name and its data type. Declaration tells the compiler about the existence of an entity in the program and its location. When you declare a variable, you should also initialize it.

* Selecting data types
* Integer: Used to store number values that won't take on decimal form.
* Single: Used to store number values that may take on decimal form. ...
* Double: A longer form of the single variable. ...
* Date: Stores date values.
* String: Stores text. ...
* Boolean: Used to store binary results (True/False, 1/0)

Fixed and dynamic arrays

A fixed array is an array for which the size or length is determined when the array is created and/or allocated. A dynamic array is a random access, variable-size list data structure that allows elements to be added or removed.

Constant

Constant is a named memory location used to hold a value that CANNOT be changed during the script execution. If a user tries to change a Constant value, the script execution ends up with an error. Constants are declared the same way the variables are declared.

**Building Procedure**

* Creating Sub and Function procedures

Sub CalcArea()

Dim a As Double, b As Double

c = a \* b

Sheet3.Activate

Sheet3.Range(“A1”).Value = c

End Sub

* Calling procedures

Sub Code\_1()

Range("A1").Value = "Hello"

End Sub

Sub Code\_2()

Range("A1").Interior.Color = rgbblue

End Sub

* Passing arguments to procedures
* VBCopy
* Dim mbResult As MsgBoxResult
* Dim displayString As String = "Show this string to the user"
* mbResult = MsgBox(displayString, , "Put this in the title bar")
* Communicating with the user through the message box
* Function Area()
* Dim Length As Double
* Dim Width As Double
* Length = InputBox("Enter Length ", "Enter a Number")
* Width = InputBox("Enter Width", "Enter a Number")
* Area = Length \* Width
* MsgBox "Area is " & Area
* End Function
* Gathering user information with the input box

Sub ssFav()

Dim response As String

response = InputBox("What’s your favorite Spreadsheeto course?")

Range("B1").Value = response

End Sub

* Utilizing the File Dialog object and Dialogs collections
* Public Sub SelectSingleFile()
* Dim FD As Office.FileDialog
* Set FD = Application.FileDialog(msoFileDialogFilePicker)
* With FD
* .AllowMultiSelect = False
* .ButtonName = "Select File"
* .Filters.Clear
* .Filters.Add "All Files", "\*.\*"
* .Filters.Add "Excel Files", "\*.xl\*"
* .FilterIndex = 1
* .InitialFileName = Environ$("USERPROFILE") & "\Desktop\"
* .InitialView = msoFileDialogViewDetails
* .Title = "Select a single file"
* If .Show = -1 Then
* Debug.Print .SelectedItems(1)
* Else
* Debug.Print "No file selected."
* End If
* End With
* End Sub
* **Select Multiple Files**
* Hold the Ctrl key to select multiple files.
* Public Sub SelectMultipleFiles()
* Dim FD As Office.FileDialog
* Set FD = Application.FileDialog(msoFileDialogFilePicker)
* With FD
* .AllowMultiSelect = True
* .ButtonName = "Select File(s)"
* .Filters.Clear
* .Filters.Add "All Files", "\*.\*"
* .Filters.Add "Excel Files", "\*.xl\*"
* .FilterIndex = 1
* .InitialFileName = Environ$("USERPROFILE") & "\Desktop\"
* .InitialView = msoFileDialogViewDetails
* .Title = "Select files(s)"
* If .Show = -1 Then
* Dim i As Long
* For i = 1 To .SelectedItems.Count
* Debug.Print .SelectedItems(i)
* Next i
* Else
* Debug.Print "No files selected."
* End If
* End With
* End Sub
* **Select a Folder**
* Properties that only pertain to selecting files do not apply to the folder picker dialog.
* Public Sub SelectFolder()
* Dim FD As Office.FileDialog
* Set FD = Application.FileDialog(msoFileDialogFolderPicker)
* With FD
* .ButtonName = "Select Folder"
* .InitialFileName = Environ$("USERPROFILE") & "\Desktop\"
* .InitialView = msoFileDialogViewDetails
* .Title = "Select a Folder"
* If .Show = -1 Then
* Debug.Print .SelectedItems(1)
* Else
* Debug.Print "No folder selected."
* End If
* End With
* End Sub
* **Open/SaveAs**
* Call the Execute method after using the Show method to execute the action of the dialog.
* Public Sub OpenExcelFiles()
* Dim FD As Office.FileDialog
* Set FD = Application.FileDialog(msoFileDialogOpen)
* With FD
* .AllowMultiSelect = True
* .ButtonName = "Select Excel File"
* .Filters.Clear
* .Filters.Add "Excel Files", "\*.xl\*"
* .FilterIndex = 1
* .InitialFileName = Environ$("USERPROFILE") & "\Desktop\"
* .InitialView = msoFileDialogViewDetails
* .Title = "Select Excel files to open"
* If .Show = -1 Then
* .Execute
* Else
* Debug.Print "No file selected."
* End If
* End With
* End Sub

**Control Structure and Program Flow**

* IF (Else If, Else)Select Case ()

| * **Sl.No** | **Conditional Statement** | **Description** |
| --- | --- | --- |
| 1 | If…Then | Set of statements are executed only if the condition is true. |
| 2 | If.. Then…Else | Set of statements under If block are executed If the condition is true otherwise statements under else block will be executed. |
| 3 | If..ElseIf | Each Else block if again have a conditional statement based on which the statements will be executed. |
| 4 | Nested Ifs | Placing an If statement inside another if statement. |
| 5 | Select Case | Each case statement will have a variable value, based on the selection value mentioned in the select case statement, appropriate case will be executed. |

**IF Statements**

If statements execute a set of actions depending on the condition. If the condition evaluates to true then the code mentioned in the If block will be executed.

**Syntax:**

If condition Then

[statements]

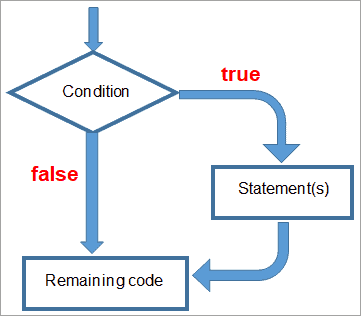
End If

**Condition:**This is the required field. Based on the Boolean result of this condition the action will be performed. If the result is true then the statements in the If block will be executed.

If the condition is Null then it is treated as False.

**Statements:**This set of actions will be performed if the condition is true.

**Flow Diagram**

[](https://www.softwaretestinghelp.com/wp-content/qa/uploads/2020/11/Flow-Diagram.png)

Once the code enters the conditional statement, the expression is verified. If the condition returns true then a set of activities defined under the if block is executed, but if the condition returns false then the program will not enter the if block.

Hence the if block statements are skipped and are never executed. The program directly goes to the line after the End If statement.

**Note:** To write VB Code Open Microsoft Excel (supported version Excel 2007,2010, 2013, 2016, 2019), navigate to **Developer Tab -> Visual Basic** (Alternatively use shortcut Alt+F11). In the VB editor, click on**Insert -> Module**.

Option Explicit

Sub ifExample()

Dim Obtained\_Marks, Total\_Marks As Integer

Obtained\_Marks = 100

Total\_Marks = 100

If (Obtained\_Marks = Total\_Marks) Then

MsgBox "Student obtained a perfect score"

End If

Debug.Print "Results Published"

End Sub

**IF… Then… Else Statements**

If the condition returns a boolean true, then the set of actions defined under the if block will be executed but if the conditional expression returns a boolean false then the statements under the else block will be executed.

**Syntax:**

If (condition) Then

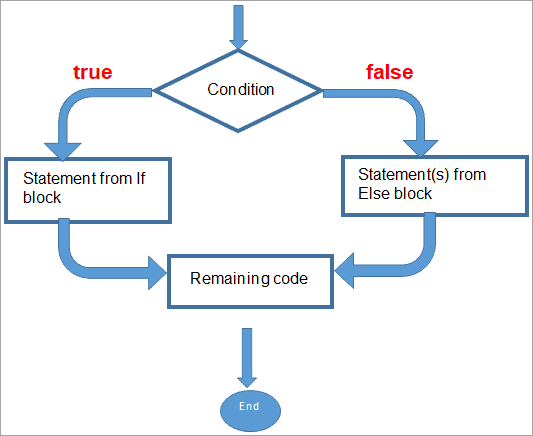
[ Statement (s) ]

Else

[Statement(s)]

End If

**Flow Diagram**

[](https://www.softwaretestinghelp.com/wp-content/qa/uploads/2020/11/Else-Statements.png)

Once the code reaches the conditional statement, it evaluates the value of the expression. The If-block is executed if the condition is true and the Else block is executed if the condition is false. It is not possible to execute both the If and Else blocks in a single run.

**Example:**

Sub ifElseExample()

Dim Obtained\_Marks, Passing\_Marks As Integer

Obtained\_Marks = 35

Passing\_Marks = 35

If (Obtained\_Marks >= Passing\_Marks) Then

MsgBox "Student has passed the exam"

Else

MsgBox "Student did not clear the exam"

End If

End Sub

**ElseIF Statements**

To test a second condition we can add ElseIf statements to a simple If..Then..Else. An If statement is allowed to be followed by multiple ElseIf statements each consisting of a conditional statement.

**Syntax:**

If(condition) Then

[Statement(s)]

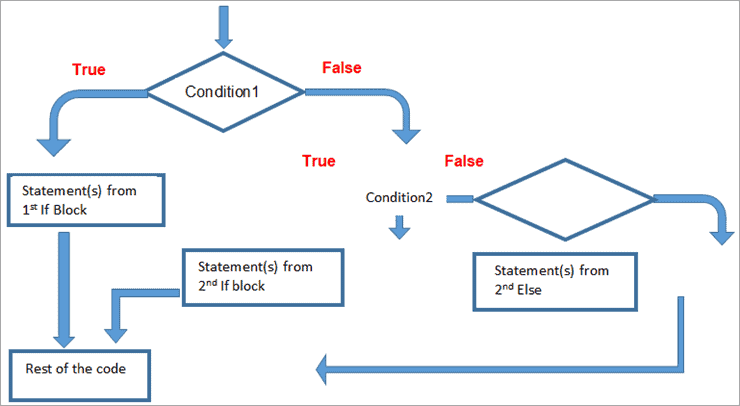
ElseIf (condition)Then

[Statement (s)]

End If

End If

**Flow Diagram**

[](https://www.softwaretestinghelp.com/wp-content/qa/uploads/2020/11/Rest-of-the-code.png)

Once the code reaches the conditional expression, it evaluates either to True or False. If the condition is true then the statements under the 1st IF block will be executed and the control exists in the conditional block, but if the expression returns false then the control will enter the 2nd conditional expressions and repeats the process.

**Example:**

Sub ifElseifExample()

Dim Obtained\_Marks, Passing\_Marks As Integer

Obtained\_Marks = 60

Passing\_Marks = 35

If (Obtained\_Marks < Passing\_Marks) Then

MsgBox "Student did not clear the exam"

ElseIf (Obtained\_Marks >= 60) Then

MsgBox "Student has cleared the exam with firstclass"

Else

Msgbox “Student passed with second class”

End If

End Sub

**Nested IF Statements**

VBA allows us to place control statements inside another control statement.

**Example:** Placing an If statement inside another if statement. This procedure of placing one control statement within another is called to be nested.

Control structures in VBA can be nested to as many levels as you wish. By intending the body of each control statement, it will be better readable.

**Syntax:**

If (condition) Then

Statement(s)

If(condition) Then

Statement(s)

ElseIf (condition) Then

Statement(s)

Else

Statement(s)

End If

Else

Statement(s)

End If

Sub NestedIFExample()

Dim Obtained\_Marks

Obtained\_Marks = 67

If (Obtained\_Marks > 0) Then

If (Obtained\_Marks = 100) Then

MsgBox "Student has got a perfect score"

ElseIf (Obtained\_Marks >= 60) Then

MsgBox "Student has cleared the exam with first class"

ElseIf (Obtained\_Marks >= 50) Then

MsgBox "Student cleared the exam with second class"

ElseIf (Obtained\_Marks >= 35) Then

MsgBox "Student has cleared"

Else

MsgBox " Student did not clear the exam"

End If

ElseIf (Obtained\_Marks = 0) Then

MsgBox "Student scrored a zero)"

Else

MsgBox "student did not attend the exam"

End If

End Sub

**Select Case**

From the above nested if statement we have seen how cumbersome it is to deal with multiple if..else statements. If you misplace a single If or Else then it is difficult to debug and hence it is more error-prone. To deal with such a problem we can use Select Case.

In Select Case, you can enter the block of code to be executed under a particular case statement. Each case statement will have a variable value to identify. Before we begin the execution, we have to specify which case is to be executed by entering the variable value in the Select Case Statement.

**Syntax:**

Select Case testexpression

[ Case expressionlist-n ]

[ statements-n ]]

[ Case Else ]

[ elsestatements ]

End Select

Sub selectExample()

Dim marks As Integer

marks = InputBox("Enter Total Marks")

Select Case marks

Case 100

MsgBox "Perfect score"

Case 60 To 99

MsgBox "First Class"

Case 50 To 59

MsgBox "Second class"

Case 35 To 49

MsgBox "Pass"

Case 1 To 34

MsgBox "Not Cleared"

Case 0

MsgBox "Scored zero"

Case Else

MsgBox "Did not attend the exam"

End Select

End Sub

Private Sub Compute\_Click()

Dim no1, no2 As Integer

Dim op As String

no1 = InputBox("Enter 1st numbers")

no2 = InputBox("Enter 2nd number")

op = InputBox("Enter Operator")

Select Case op

Case "+"

MsgBox " Sum of " & no1 & " and " & no2 & " is " & no1 + no2

Case "-"

MsgBox " Difference of " & no1 & " and " & no2 & " is " & no1 - no2

Case "\*"

MsgBox " Product of " & no1 & " and " & no2 & " is " & no1 \* no2

Case "/"

MsgBox " Division of " & no1 & " and " & no2 & " is " & no1 / no2

Case Else

MsgBox " Operator is not valid"

End Select

End Sub

# For...Next statement

* Article
* 03/30/2022
* 7 contributors

Feedback

## yntax

**For** counter **=** start **To** end [ **Step** step ]  
[ statements ]  
[ **Exit For** ]  
[ statements ]  
**Next** [ counter ]

The **For…Next** statement syntax has these parts:

Expand table

| **Part** | **Description** |
| --- | --- |
| counter | Required. Numeric [variable](https://learn.microsoft.com/en-us/office/vba/language/glossary/vbe-glossary#variable) used as a loop counter. The variable can't be a [Boolean](https://learn.microsoft.com/en-us/office/vba/language/glossary/vbe-glossary#boolean-data-type) or an [array](https://learn.microsoft.com/en-us/office/vba/language/glossary/vbe-glossary#array) element. |
| start | Required. Initial value of counter. |
| end | Required. Final value of counter. |
| step | Optional. Amount counter is changed each time through the loop. If not specified, step defaults to one. |
| statements | Optional. One or more statements between **For** and **Next** that are executed the specified number of times. |

## Remarks

The step [argument](https://learn.microsoft.com/en-us/office/vba/language/glossary/vbe-glossary#argument) can be either positive or negative. The value of the step argument determines loop processing as follows.

Expand table

| **Value** | **Loop executes if** |
| --- | --- |
| Positive or 0 | counter <= end |
| Negative | counter >= end |

After all statements in the loop have executed, step is added to counter. At this point, either the statements in the loop execute again (based on the same test that caused the loop to execute initially), or the loop is exited and execution continues with the statement following the **Next** statement.

**Tip**

Changing the value of counter while inside a loop can make it more difficult to read and debug your code.

Any number of [**Exit For**](https://learn.microsoft.com/en-us/office/vba/language/reference/user-interface-help/exit-statement) statements may be placed anywhere in the loop as an alternate way to exit. **Exit For** is often used after evaluating some condition, for example **If...Then**, and transfers control to the statement immediately following **Next**.

You can nest **For...Next** loops by placing one **For...Next** loop within another. Give each loop a unique variable name as its counter. The following construction is correct:

VBCopy

For I = 1 To 10

For J = 1 To 10

For K = 1 To 10

...

Next K

Next J

Next I

**Note**

If you omit counter in a **Next** statement, execution continues as if counter is included. If a **Next** statement is encountered before its corresponding **For** statement, an error occurs.

## Example

This example uses the **For...Next** statement to create a string that contains 10 instances of the numbers 0 through 9, each string separated from the other by a single space. The outer loop uses a loop counter variable that is decremented each time through the loop.

VBCopy

Dim Words, Chars, MyString

For Words = 10 To 1 Step -1 ' Set up 10 repetitions.

For Chars = 0 To 9 ' Set up 10 repetitions.

MyString = MyString & Chars ' Append number to string.

Next Chars ' Increment counter

MyString = MyString & " " ' Append a space.

Next Words