09: Introduction to Macro and Visual Basic for Application (VBA)

Macro is a programming tool that can be used to automate activities in Microsoft Office applications such as Word, Excel, PowerPoint and Access. You can use macro to capture a series of steps you perform manually in Excel, like formatting cells, copying and pasting data, or applying formulas. Once recorded, the macro can be replayed any time to automate those tasks, saving you time and effort. Macros are a great way to streamline repetitive processes in your spreadsheets.

VBA is the programming language behind macros in Excel. When you record a macro, Excel translates your actions into VBA code. While you can use recorded macros, VBA allows for more powerful automation. With VBA, you can write custom code to perform complex tasks, interact with other programs, and create custom functions in Excel.

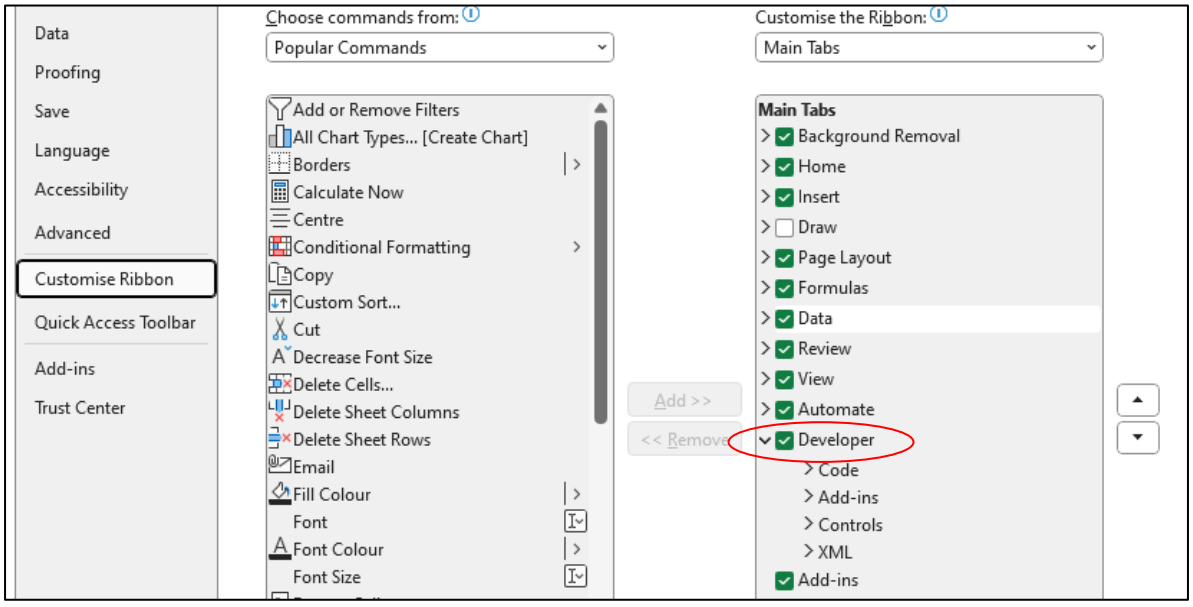
MORE INFO

01. Quick Start : Macro
02. Getting started with VBA in Office

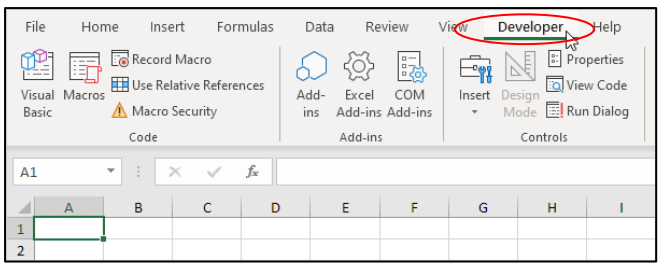
09.1 Enable Macro environment

File → Options → Customize Ribbon

Then tick (✓) the "Developer" check box (as shown in the figure below)



When you have the developer toolbar, you'll see the following tab in the Ribbon



09.2 How to save Excel workbooks which have macros

Excel files have a special code to run when users call to a macro

The security system of the OS will identify Excel files with macros as a virus infected file.

Therefore, you should save files as macro enabled Excel files

You should save the excel workbook as a Macro enabled Excel file ".xlsm"

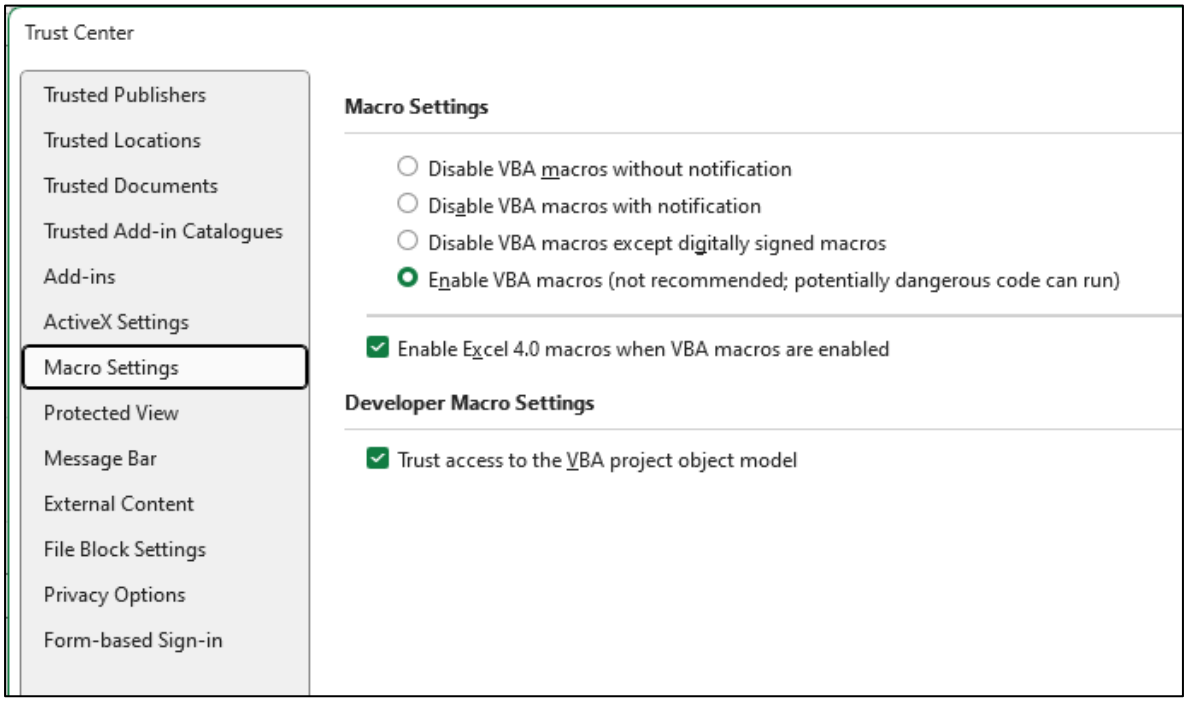
09.3 Change security settings (Trust Center Settings).

When you create and save the Excel file as a Macro-enabled file (xlsm), in addition to data, it will save with a Visual Basic source code (Program). Therefore, even if you save the file as a Macro-enabled file, the OS may disable the "Code" part as a security feature.

To enable VBA macros, go to

Options → Trust center → Trust center settings → Macro Settings

You should change the Trust settings to enable Macros.



If still Macro is disabled, you should add the file location as a trusted location.

Go to Options → Trust center → Trust center settings → trusted location → click on the "Add New Location" button and select the folder that you saved Macro file.

Note – During the practice period, you need to perform **09.1** and **09.3** only **once (per one computer)**.

Why do we need to use Macro?

You can automate repetitive activities using macros. For example, suppose you need to use a same set of formatting in your day-to-day tasks: Set font color Green, Background color Yellow, Size 20 etc. You can create a Macro in Excel to do these formatting and the macro will perform all these formatting when you call that Macro.

Think of it like this: if you spend 10 minutes every day formatting a report, a macro can be created to do that formatting in seconds. That's 10 minutes saved every day, which adds up quickly!

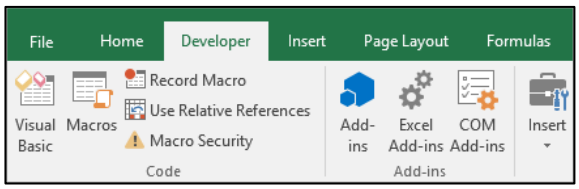
09.4 How to create a Macro?

There are two basic methods. **1) Record a macro** and **2) Write a macro**

Record a macro	**Write a macro**
You instruct Excel to "START" recording and then Excel record all actions. You can stop clicking on the "Stop Recording" button.	You write or edit instructions using Visual Basic programming language.
Simple.	A bit complex.
Programming (VBA) knowledge is not required.	Visual Basic programming knowledge is required.
Limited functions are available.	Virtually unlimited functions.

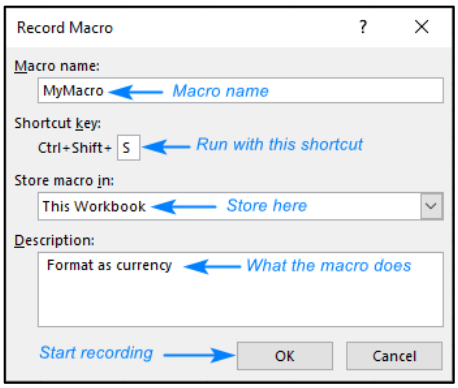
09.5 Record a Macro

Excel stores information about each step you take as you perform a series of commands. You then run the macro to repeat, or "playback," the commands.



To create a macro,

1) Go to Developer Tab → Record Macro.



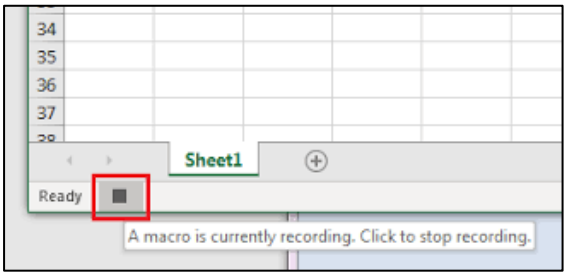
2) Specify the following and click OK:
   a) Macro name (no spaces, can't start from a number)
   b) Shortcut key (if needed)
   c) Location to store the macro
   d) Description - a good reminder of what the macro does (not how, just what)

3) Recording
   Now your Excel is in the Macro recording mode and all the activities such as every click, formatting, cell changes, scroll actions, typing will be recorded within the Macro. Once you complete all the actions you want to "stop recording".

4) How to stop recording?

- The menu tool: You can click on “Stop Recording”. One is by viewing the Macro menu and noting that “Stop Recording” has replaced the option for “Record Macro”

- The other option is in the bottom left corner. The ‘stop’ icon indicates that it is in macro mode and pressing here will stop the recording (likewise, when not in record mode, this icon will be the Record Macro button, which you can use instead of going to the Macros menu).



Activity 09.1 (Part A)

You are given sales information for 07 months in “Activity 09-1.xlsm” excel workbook. Perform the following activities.

01. Select “B1” cell (*Important*) in “Activity 09.1” worksheet → Developer tab → Record Macro
02. Strat a macro by giving the macro name as “FirstMacro” and Shortcut Key “Ctrl+t”
03. Select B1 to E1 cell range → Merge and Center
04. Increase the font size of the title to 14.
05. Adjust the width of the Title cell.
06. Bold the Title
07. Auto adjust the width of the “Product Category” column and “Total Sales” column
08. Select “B3:D10” cell range → Add “All Borders”
09. Bold the column headers
10. Add a fill color to column headers
11. Insert a fill color for province names
12. Stop the macro
13. Insert the same formatting applied above for the “Activity 09.1 (i)” and “Activity 09.1 (ii)” sheets

09.6 Absolute Vs Relative cell addresses

When you record a Macro, you can set two address methods, 01) Relative and 02) Absolute

How to change the mode of Address

If the "Use Relative References" button highlighted, the Macro record addresses as Relative, otherwise the mode is Absolute.

Activity 09.1 (Part B)

Use the same "Activity 09-1.xlsm" excel workbook to perform the following.

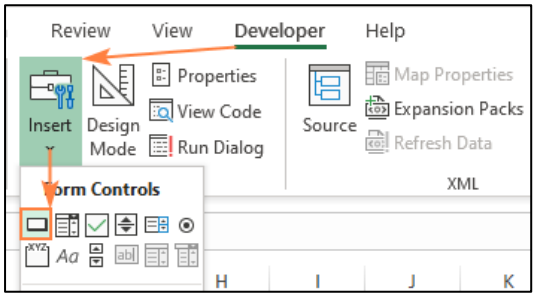
01. Select Developer tab → Enable "Use Relative References" option.
02. Select "B8" cell in "Activity 09.1 (iii)" sheet → 'Record Macro'
03. Strat a macro by giving the macro name as "SecondMacro"
04. Merge and Center the Title.
05. Increase the font size of the title to 14.
06. Adjust the width of the Title cell.
07. Bold the Title
08. Select "B10:D17" cell range → Add "All Borders"
09. Bold the column headers
10. Add a fill color to column headers
11. Insert a fill color for province names
12. Stop the macro
13. Insert the same formatting applied above for the "Activity 09.1 (iv)" sheet.

09.7 Macro Controls

These are elements you insert into your spreadsheet, like buttons, dropdown menus, text boxes etc. Macro controls are essentially a way to provide a user-friendly interface to run your macros. Instead of needing to access the macro code directly, users can simply click a button or make a selection to trigger the automation.

09.7.1 Buttons

A Button control looks like a Microsoft Windows button and runs a macro when clicked. It's a much handier way to access your most commonly used macros and is an easy way to expose custom functionality to other users of your workbook.



Activity 09.1 (Part C)

Use the same "Activity 09-1.xlsm" excel workbook to perform the following.

01. Go to "Activity 09.1 (iv)" worksheet
02. Insert a button to run "SecondMacro"
03. Rename the button as "Apply Formatting"
04. Apply formatting to "June" and "July" sales data

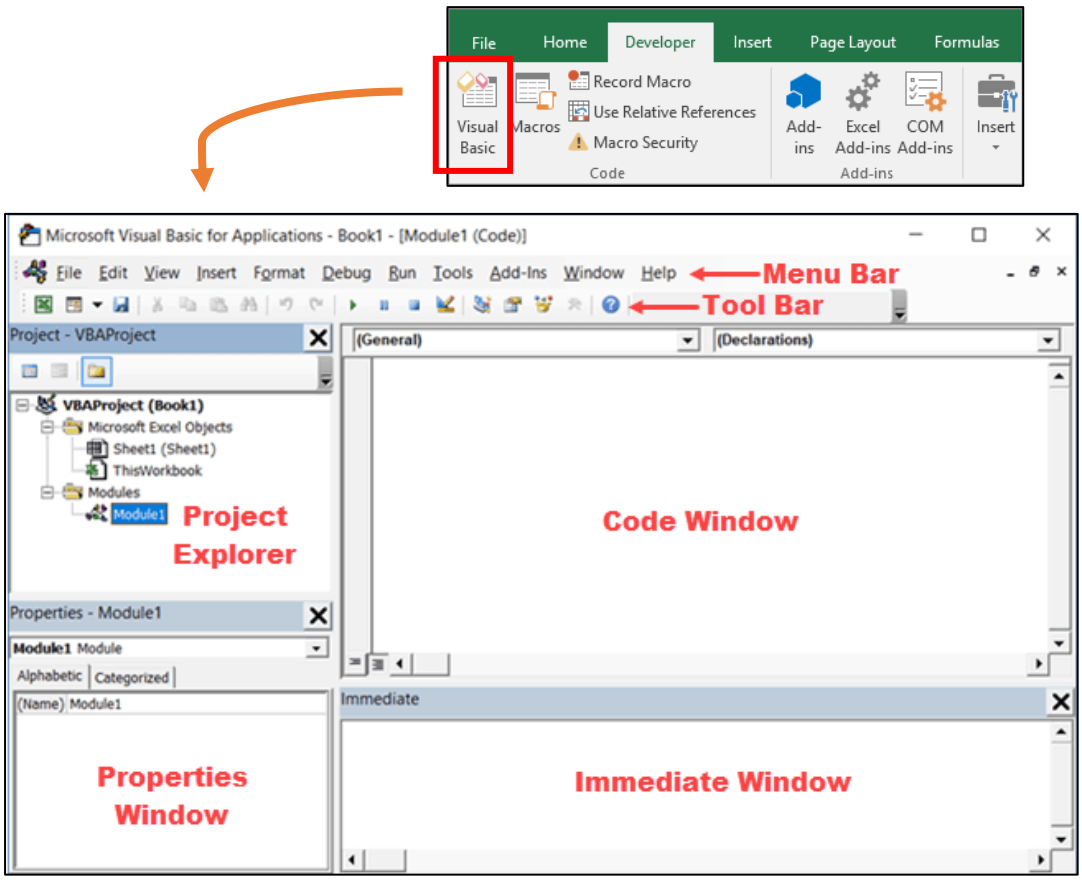
Note - To rename/edit a button, first right-click on the button, then it goes to edit mode, now you can rename the button text.

09.8 Write Macro (like a programmer)

We can write or edit Macros directly by accessing the source code of the Macro.

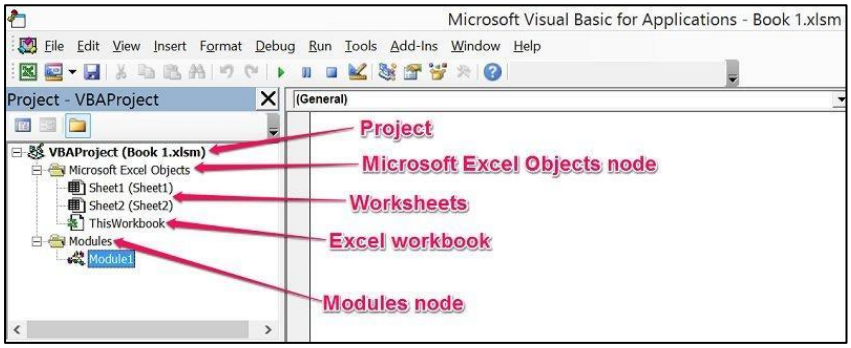
09.8.1 Understand Visual Basic (VB) Editor

Macro will record actions using Visual Basic programming language. Therefore, we can write a new Macro or edit a recorded macro using Visual basic. VB is a user-friendly programming language, and even general users can easily learn it. This section explains the editing environment.



Project explorer

The 'Project Window', also known as the 'Project Explorer', is useful for navigation purposes. This is the section of the Visual Basic Editor where you'll be able to find every single Excel workbook that is currently open. This includes add-ins and hidden workbooks. More particularly, each Excel workbook or add-in that is open at the moment appears in 'Project Explorer' as a separate project. A project is (basically/simply) a set of modules.



Programming Window / Code Window / Module Window

As you may expect, the 'Code Window' of the Visual Basic Editor is where your VBA code appears, and where you can write and edit such code. At the beginning, though, the Programming Window is empty as in the screenshot above.

There is a Code Window for every single object in a project. You can access the window of a particular object by going to the Project Explorer and doing any of the following:

Property Window

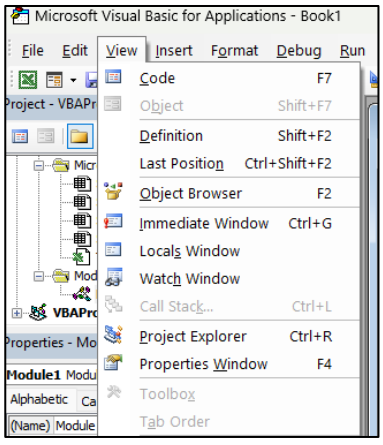
The Properties Window displays the properties of the object that is currently selected in the Project Explorer and allows you to edit those properties.

Immediate Window

The main purpose of the Immediate Window is to help you notice errors, checking or debugging VBA code. The Immediate Window is, by default, hidden. However, as with most of the other windows, you can unhide it.

Manage Windows

Windows can be managed (show/hide) from the "View" menu of the Visual Basic IDE. These buttons are toggled, and you can show/hide clicking on the window names.



09.8.2 Edit a Macro using Visual Basic editor

Activity 09.2

Open a new excel workbook and save it as a macro enabled excel file, giving the name "Activity 09-2". Perform the following in the new workbook.

01. Select Developer tab → Enable "Use Relative References" option.
02. Select "B2" cell and then click 'Record Macro'. Give the macro name as "Third_Macro"
03. Enter your CPM number in the "B2" cell and perform the formatting below
    a. Add a blue fill color to the "B2" cell
    b. Increase the font size to 16
    c. Change the font color to Red
04. Stop macro recording
05. Assign the macro to a Button (Rename the button as "Edit Macro Example")
06. Check whether macro is working properly by Inserting and formatting your CPM number in other any cell.
07. Edit the VB code as follows
    a. Change the fill color to Yellow
    b. Update the CPM number to your MC number
    c. Increase the font size to 18
    d. Change the font color to Green

Structure of the Macro
One module may include more than one Macro code. A Macro code starts with "Sub" (Subroutine) and ends with "End Sub". Macro records instructions in between these two lines. You must not change any "BLUE" color words. They make the structure of the Macro.

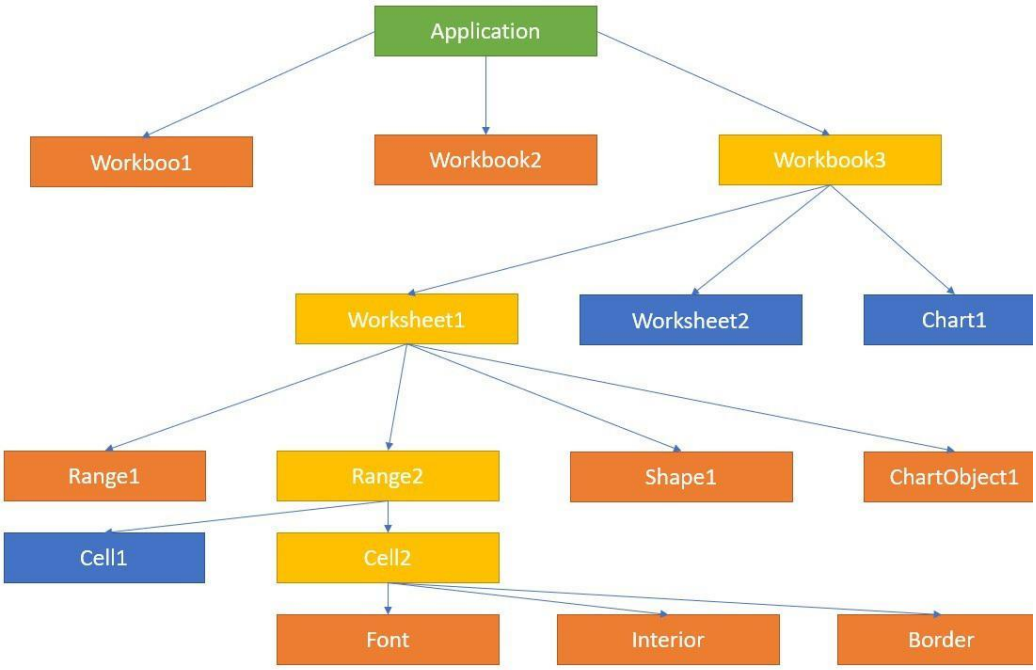
Comments
If there are **green** color text, they will not be executed and called "Comments". Comments are useful for the programmer to insert some details or comments. To add a comment you must start a line with single quote (**'**).

09.9 Excel Objects

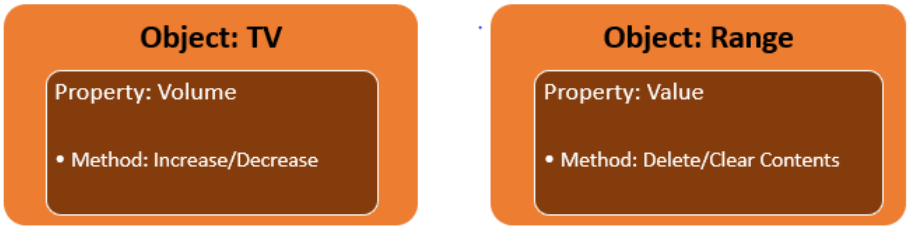
Excel objects collectively form the Excel Object Model. This model provides a way to interact with various elements within an Excel spreadsheet using code. Each object has its own properties (characteristics) and methods (actions you can perform on it).

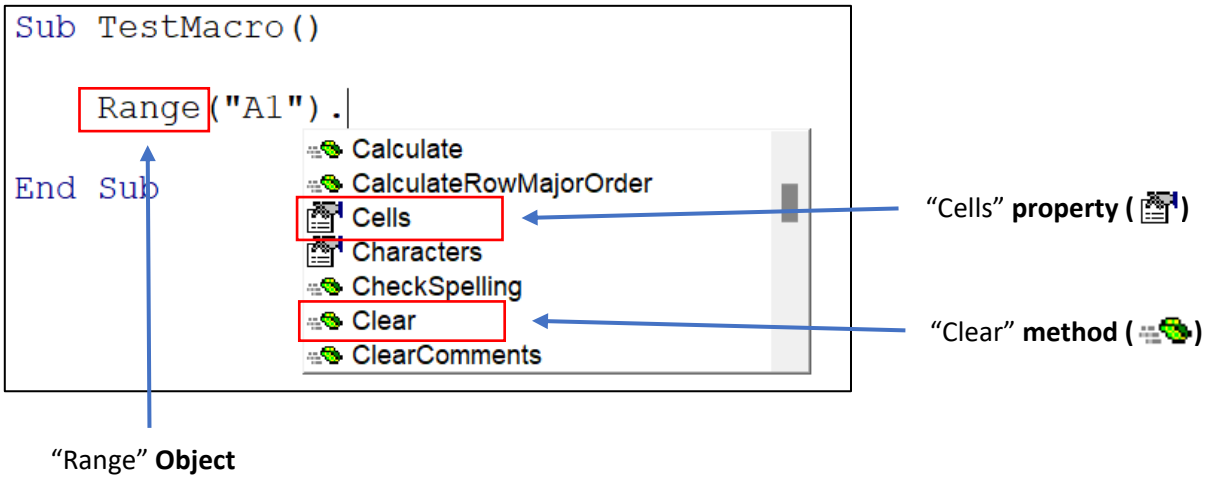
Examples of Excel Objects:

- Workbook: Represents the entire Excel file, containing one or more worksheets.
- Worksheet: An individual sheet within the workbook where you organize your data in rows and columns.
- Range: A selection of cells, which can be a single cell, a row, a column, a block of cells, or even noncontiguous selections. It's the fundamental building block for working with data in your worksheet.
- Chart: A visual representation of your data using elements like bars, lines, or pie charts.
- Cell: The individual intersection of a row and column where you enter or display data.



By understanding and using these objects and their **properties** and **methods**, you can automate tasks in VBA.





Activity 09.3

Open a new excel workbook and save it as a macro enabled excel file, giving the name "Activity 09-3". Perform the following in the new workbook.

01. Create a new macro named "MyName".
    (Path – **Developer** Tab → **Macros** → Provide the macro name → **Create)**
02. Write a VB code to display your name in "B2" cell.
03. Run the macro by selecting the relevant code segment (in VB editor).

01. Create a new macro named "CopyValues".
02. Write a VB code to copy your name from "B2" cell to "D2" cell.
03. Run the "CopyValues" macro by assigning to a button (Rename the button as "Copy Name")

01. Select "D4" cell.
02. Create a new macro named "Active_Cell_Value".
03. Write a VB code to assign number 50 to active cell.
04. Run the macro by selecting the relevant code segment (in VB editor).

Activity 09.4

Open a new excel workbook and try out the following.

Code	Description
ActiveCell.Interior.Color = RGB(0, 100, 255)	Add a fill color to the Active Cell
ActiveCell.Font.Color= RGB(255, 255, 255)	Change the font color in the Active Cell
Application.Quit	Close excel application
Worksheets("Sheet1").Cells(2, 3).Value = "2nd Row 3rd Column"	Alternative method of selecting a cell and inserting a value
Rows(1).EntireRow.Clear	Clear contents of the first row
Rows(1).EntireRow.Delete	Delete the first row
Columns(1).EntireColumn.Clear	Clear contents of the first column
Columns(1).EntireColumn.Delete	Delete the first column

09.10 Variables

Variables are 'places' that you create to store "data" when you run your program. Variables have a name (you think of) and a type (such as Integer, String, Decimal etc.).

- In VBA first you need to declare the variable. This is done (mostly) with the word "Dim"

  E.g.- **Dim** intNumber **As Integer**.

- After declaring, you can initialize the variable. That means assigning a value to the variable.

  E.g.- intNumber = 12

Rules for naming variables

- A variable name must start with a letter.
- The name of a variable cannot contain spaces.
- You cannot use special characters such as #
- You cannot use reserved names such as 'save' and 'print'.

09.10.1 Basic Data Types

	Data Type Name	Type	Data Range and Remarks
Non-numeric	String	Text	Text.
	Date	Date	Date and time.
	Boolean	Boolean	True or False.
	Variant	Any type	Default type if the variable is not declared with any data type. It will accept any kind of data.
Numeric	Byte	Numeric	Whole number between 0 and 255.
	Integer	Numeric	Whole number between -32,768 and 32,767.
	Long	Numeric	Whole number between – 2,147,483,648 and 2,147,483,647.
	Single	Numeric	Floating decimal number between -3.402823E38 and 3.402823E38.
	Double	Numeric	Floating decimal number between -1.79769313486232D308 and 1.79769313486232D308.

Activity 09.5

Open a new excel workbook and save it as a macro enabled excel file, giving the name “Activity 09-5”.

Part 1

Create an excel macro that calculates the area of a rectangle using variables. The user should input the length and width in “B3” and “C3” cells respectively, and the macro should display the calculated area in message box. (Name the macro as “CalArea” and assign it to a button named “Calculate Area”)

Part 2

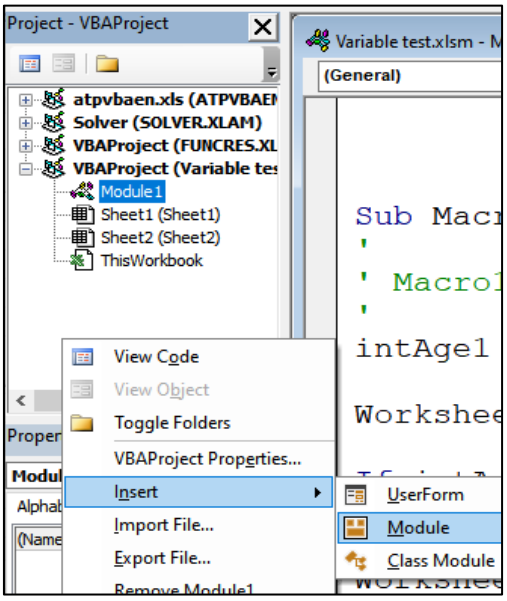
Type your first name in “B6” cell and last name in “C6” cell. Create an excel macro to show your full name (as “Full name : _____ _____”) in “E6” cell. You need to use variables for this.

09.11 User Defined Functions

Excel has a large collection of functions. In most situations those functions are sufficient to get the job done. If not, you can create your own function called "User Defined Function" or "Custom Excel function". You can access a User Defined Function just like any other Excel function. The function returns the answer (processed data) to the calling cell.

User defined functions need to be placed into a module.

Right click on 'Microsoft Excel Objects' → Insert → Module (if there is a module, ignore this step).



Activity 09.6

Open the given "Activity 09-6" macro enabled excel file and perform the following.

Part 1 - Function without inputs

Write an Excel function to return your name once you enter the function in a worksheet. Function name – "MyName"

Part 2 - A function with one input value(variable)

Write a function that accepts number of workdays as an input (as below) and multiply it by 1000 to calculate the basic salary. Function name – "Basic".

Employee Name	Number of workdays	Basic Salary
Amal	25	=Basic (NoofWorkdays)
Nimal	19	
Ram	15	
Mohamad	28	

Part 3 - A function with multiple input values(variables)

Suppose individual employees have different salary rates. Write a function to accept number of workdays and the rate as inputs (as below) and to calculate the basic salary. Function name – "BasicByRates".

Employee Name	Number of workdays	Rate	Basic Salary
Amal	25	800	=BasicByRates(NoofWorkDays*Rate)
Nimal	19	1000	
Ram	15	700	
Mohamad	28	900	

Part 4 - A function with conditions (if statement)

Write a function that accepts number of workdays as an input and multiply it by 1000 to calculate the basic salary. But the number of workdays is greater than 20, the respective employee would receive an extra 5000 as a bonus. Function name – "BasicWithBonus".

Employee Name	Number of workdays	Basic Salary
Amal	25	=BasicWithBonus (NoofWorkDays)
Nimal	19	
Ram	15	
Mohamad	28	

Self-Learning Activity 1

Open a new excel workbook and save it as a macro enabled excel file. Perform the following in the new workbook.

01. Add a new worksheet.
02. Create a new macro named “CopyBetweenSheets”.
03. Type your name in “B2” cell in “Sheet1”
04. Write a VB code to copy your name to “B3” cell in “Sheet2” worksheet.
05. Run the “CopyBetweenSheets” macro by assigning to a button (Rename the button as “Copy to Sheet”)

01. Enter your hometown in “B4” cell.
02. Create a new macro named “DisplayHometown”.
03. Write a VB code to display your hometown (entered in “B4” cell) in a message box as follows. **Your Hometown is : [Your Hometown]**
04. Run the “DisplayHometown” macro by assigning to a button (Rename the button as “Display Hometown”).

01. Insert a new button to “Sheet1” worksheet. (You can rename the button later)
02. Create a new macro at the time of inserting the button, named “ClearData”.
03. Write a VB code to clear “A1:E10” cell range.
04. Rename the button as “Clear” and run the VB code by clicking the button.

Self-Learning Activity 2

Perform the following activities in **Sheet1** of the **Self_Study_01.xlsm** Excel file.

Part 1

A list of products with “Unit Price” and “Amount” has been given to you. Write a macro to calculate the value of each product multiplying each product’s “Unit price” by “Amount”. The sum of values of all products should be calculated after the “Total” cell. Use ‘for’ loops for this task.

Part 2

Assume you are given a task to calculate the basic salary (salary after including the bonus) of the employees in your organization, in the given excel sheet. Write a function named “BasicSalary” to calculate the basic salary based on the conditions below.

- If an employee is a “Manager” and has a service more than or equal to 20 years, the bonus is 20,000. If the service is less than 20 years the bonus is 10,000.
- If an employee is a “Clerk” and has a service more than or equal to 10 years, the bonus is 8,000. If the service is less than 10 years the bonus is 4,000.
- If an employee is an “Accountant” and has a service more than or equal to 15 years, the bonus is 12,000. If the service is less than 15 years the bonus is 6,000.