# Welcome to Excel Automation using VBA/Macros

1. To Automate your Excel Tasks
2. To Save time
3. To get Accurate results
4. To get new job/IJP
5. To get highly paid
6. To create User Interactive Forms
7. One of the biggest advantages of VBA is that it is already present as part of Office. This means that people who want to implement small projects in organizations can do so without having to fund additional money to do it.
8. Excel is everywhere…so you can automate anywhere…☺

## 8 Reasons why you should learn Macros

1.2

More than 1.2 billion people use Microsoft Office in 140 countries and 107 languages around the world.

Source: Microsoft

- Introduction to VBA/Macros
- How to automate your Excel reports
- Loops with 30+ examples
- Writing Conditions (IF Construct)
- Solving the real-time requirements
- Creating Userforms
- Assignments

## Agenda

## What is VBA?

VBA stands for Visual Basic for Applications. **VBA is a programming language which Excel can understand. In VBA language we write macros to automate Excel tasks.**

## What is Macros?

**A Macro is a series of instructions you give Excel in the language called VBA.** Instructions can be anything which you perform in Excel…like Formatting cells, inserting a new sheet, refreshing pivot table or it can be consolidating all your sheets data into master sheet.

```
Sub Macro1()

Sheets.Add

End Sub
```

```
Sub Macro2()

Range(“A1:B10”).Copy
Range(“D1”).PasteSpecial paste:=xlPasteAll
Range(“E1”).PasteSpecial paste:=xlPasteValues

End Sub
```

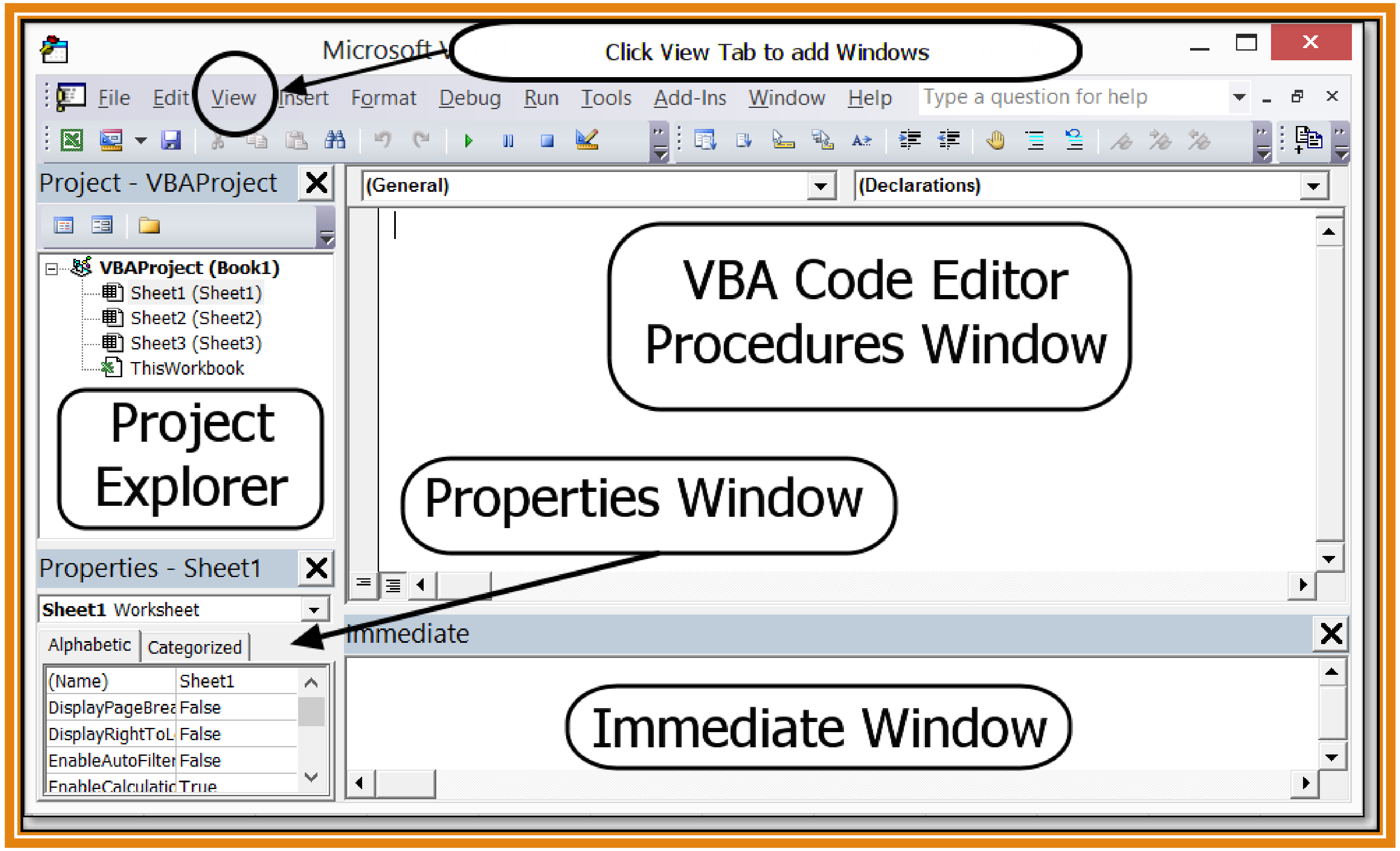
2010 to 2021

- File =>Options => Customize Ribbon => Developer

2007

- Office Button => Excel Options => Popular => Enable Developer Tab

## Enabling “Developer” tab

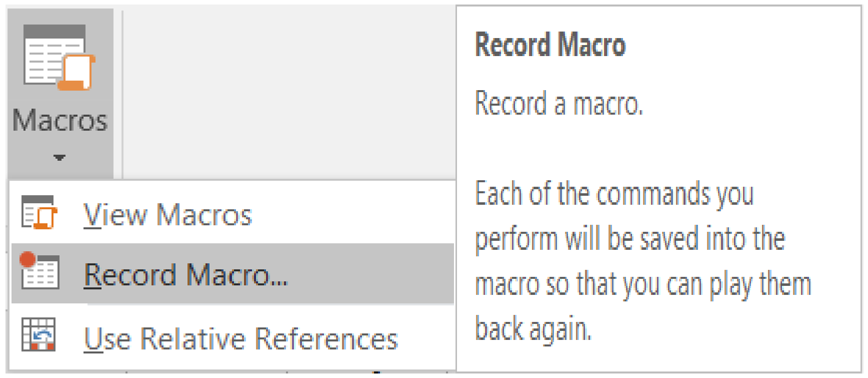


| | |
|---|---|
| (Name) | Sheet1 |
| DisplayPageBrea | False |
| DisplayRightToL | False |
| EnableAutoFilter | False |
| EnableCalculatic | True |

# Different Windows in VBA

1. Project Explorer Window
2. Property Window
3. Code Window
4. Immediate Window
5. Watch Window
6. Object Explorer Window

## Different Windows in VBA



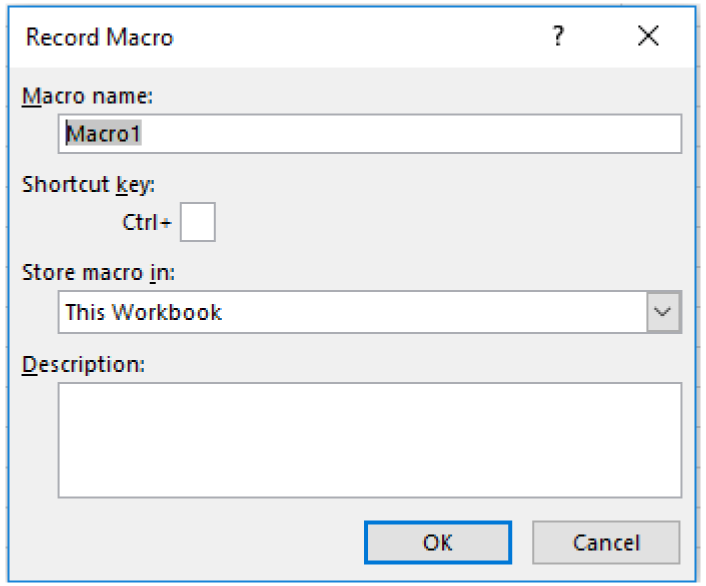
Record Macro

Record a macro.

Each of the commands you perform will be saved into the macro so that you can play them back again.

- Recording a macro is like programming a phone number into your cell phone. You first manually dial and save a number. Then as per your requirement, you can redial those numbers with the touch of a button.
- In the same way, you can record your actions in Excel while you perform them at the first instance. (Recording a Macro)
- After you've recorded a macro, you can play back those actions anytime you wish.
- Excel Macros are recorded using programming language called VBA.

## Recording a Macro



Macro name:

Macro1

Shortcut key:

Ctrl+

Store macro in:

This Workbook

Description:

# Record Macro Dialog Box

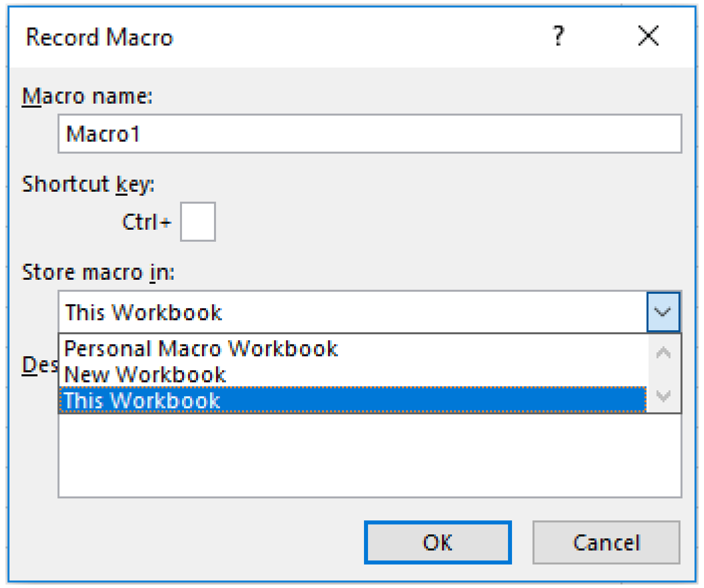
1. **Macro Name:** This is self-explanatory. Excel gives a default name to your macro, such as Macro1, but you should give your macro a name more descriptive of what it actually does.

2. **Shortcut Key:** Every macro needs an event, or something to happen, for it to run. This event can be a button press, a workbook opening, or in this case, a keystroke combination. When you assign a shortcut key to your macro, entering that combination of keys triggers your macro to run. This is an optional field.

3. **Store Macro In:** This Workbook is the default option. Storing your macro in This Workbook simply means that the macro is stored along with the active Excel Workbook. The next time you open that particular workbook, the macro is available to run. Similarly, if you send the workbook to another user, he/she can run the macro as well (provided the macro security is properly set by your user).

4. **Description:** This is an optional field, but it can come in handy if you have numerous macros in a spreadsheet or if you need to give a user a more detailed description of what the macro does

## Parts of the Record Macro Dialog Box


Record Macro

Macro name: Macro1

Shortcut key: Ctrl+

Store macro in: This Workbook

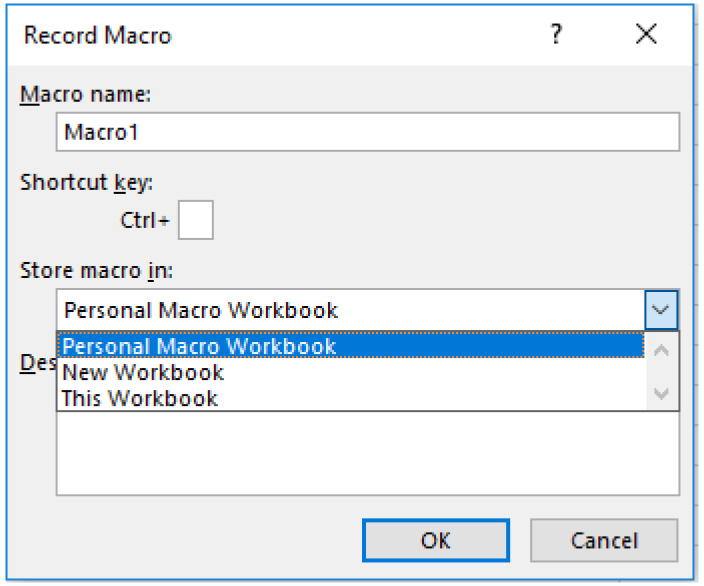
Personal Macro Workbook
New Workbook
This Workbook

- Storing your macro in This Workbook means the macro has been stored along with the active Excel file. The next time you open that particular workbook, the same macro is available to run.

- This behavior is okay as long as you don't need to use that particular macro in other workbooks.

- In case you send the workbook to another user, he/she shall be able to run the macro as well (provided the macro security is properly set by your user).

## Storing Macros in This Workbook


Record Macro

Macro name: Macro1

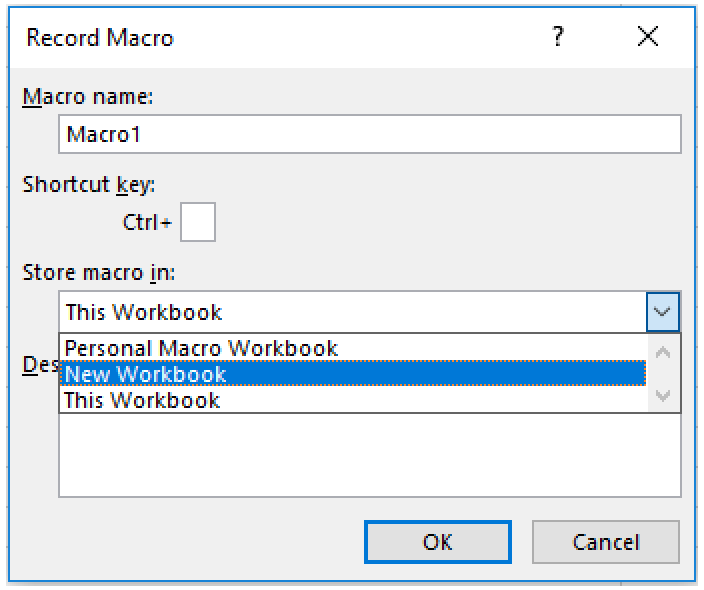
Shortcut key: Ctrl+

Store macro in: Personal Macro Workbook

- Personal Macro Workbook
- New Workbook
- This Workbook

- Any macros that you store in your Personal Macro Workbook on a computer become available to you in any other workbook whenever you start Excel on that same computer.
- To make your macros available every time you open Excel, create them in a workbook called Personal.xlsb (Excel saves your Personal Macro Workbook as Personal.xlsb).
- This is a hidden workbook stored on your computer which opens every time you open Excel.

## Storing Macros in Personal Macro Workbook


Record Macro

Macro name:

Macro1

Shortcut key:

Ctrl+

Store macro in:

This Workbook

- The New Workbook option puts the macros in a newly created workbook.
- This workbook is visible to all and can be easily shared with other people.

## Storing Macros in New Workbook

- Recorded Macro can be used multiple times.
- The Macro Recorder is quick and easy to use.
- The Macro Recorder can help you discover which VBA objects, methods and Property correspond to which part of an applicant’s interface.

## Advantages of Macro Recording

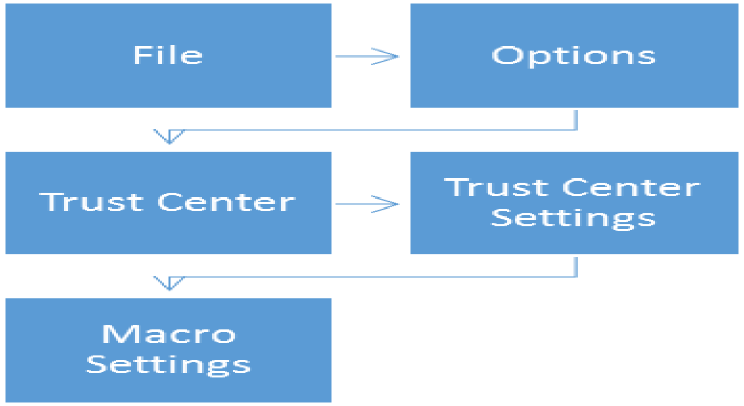
- Recorded macro may contain unnecessary statements, because Macro recorder record everything you do in the application. Ex:- If you scroll up/down, selecting any other windows…etc.
- Macro recorder record many things but not everything…Ex:- If you want to display dialog box or a user form it cannot be done using macro recorder.
- Using Macro recorder you cannot build IF…ELSE…END Construct and also Loops

## Limitations of Macro Recording

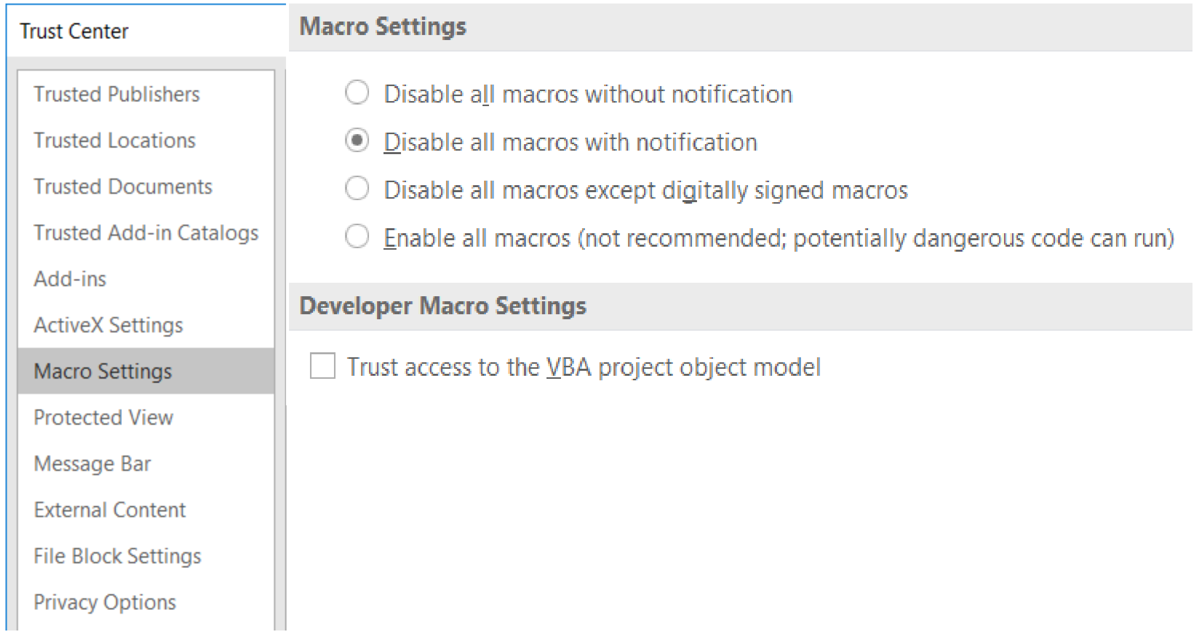
1. Macro name must use a letter as the first character.
2. Macro name can't use a space, period (.), exclamation mark (!), or the characters @, &, $, # in the name. Instead you can use underscore (_) in Macro name
3. Macro Name can't exceed 255 characters in length.
4. Generally, you shouldn't use any names that are the same as the functions, statements, and methods in Visual Basic for assigning Macro name. Ex:- VLOOKUP,SELECT,VALUE

## Naming Rules of a Macro

- The Excel default Macro Security level is Disable all macros with notification.
- VBA Macros (typically attached to email messages) occasionally serve as virus vectors, so by default macros are disabled.
- Follow these steps to change your security settings:



- Note: You can also find Macro Settings option in Developer ribbon


Trust Center

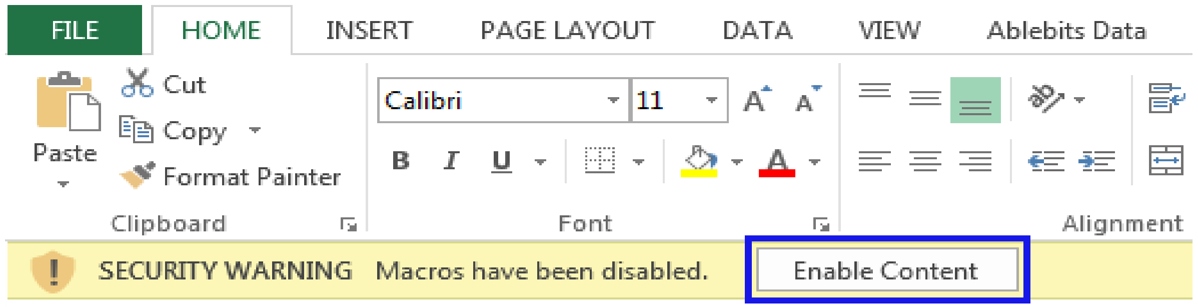
- Trusted Publishers
- Trusted Locations
- Trusted Documents
- Trusted Add-in Catalogs
- Add-ins
- ActiveX Settings
- Macro Settings
- Protected View
- Message Bar
- External Content
- File Block Settings
- Privacy Options

Macro Settings

- ○ Disable all macros without notification
- [✓ ● Disable all macros with notification]
- ○ Disable all macros except digitally signed macros
- ○ Enable all macros (not recommended; potentially dangerous code can run)

Developer Macro Settings

- ☐ Trust access to the VBA project object model



SECURITY WARNING Macros have been disabled. Enable Content

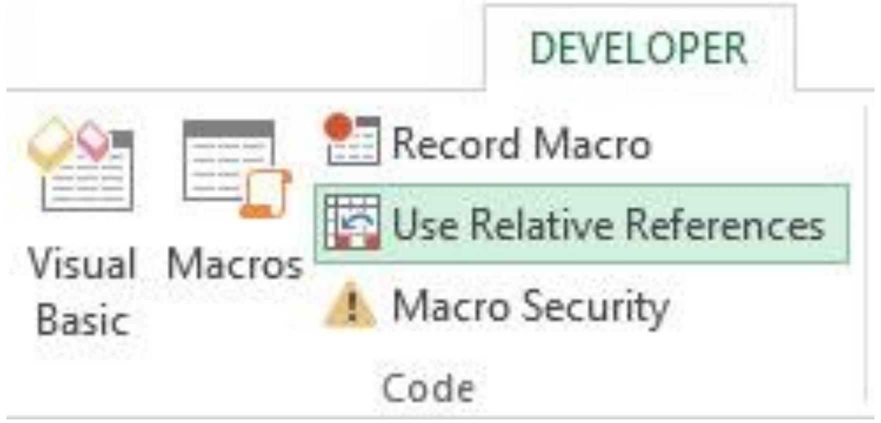
## Configuring Macro Security

- ➢ The Excel default workbook format (.xlsx) does not support Macros.
- ➢ To save a workbook containing one or more macros, you need to use one of the following formats:

| Description | Extension |
|---|---|
| Excel Macro-Enabled  Workbook | .xlsm |
| Excel Macro-Enabled Template | .xltm |
| Excel Binary Workbook | .xlsb |
| Excel 97-2003  Workbook | .xls |

## Macro files to Save As…

- Excel's default recording mode is set as Absolute Reference.
- To make it record relative references instead, click on Macros and then on Use Relative References under the View tab.
- You can toggle between two modes while recording which means your recorded Macros can be a combination of Absolute and Relative references.
- **Difference:** When a macro is recorded using relative reference, Excel will not explicitly select a particular cell as it does while recording an absolute reference macro.



## Absolute & Relative Reference

Pick a Car

Microsoft

Infosys

Wipro

Pick a Car from Microsoft

```
Cars(“Car 1”).Select
```

```
Companies(“Microsoft”).Cars(“Car 1”).Select
```

```
Cities(“Hyderabad”).Companies(“Microsoft”).Cars(“Car 1”).Select
```

```
Countries(“India”).States(“Telangana”).Cities(“Hyderabad”).Compa
nies(“Microsoft”).Cars(“Car 1”).Select
```

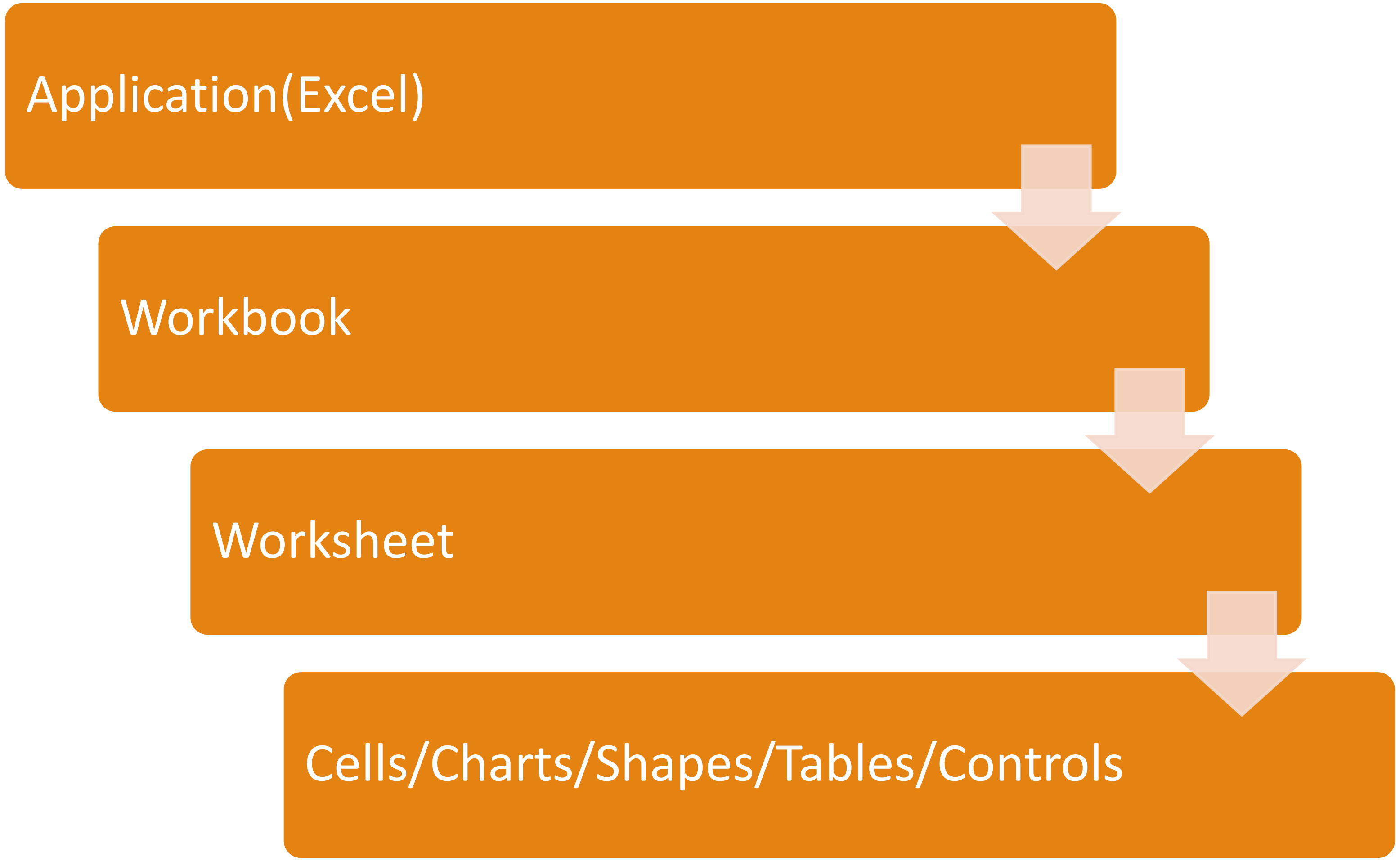
## Selecting Your Fav Car

Range(“A1”).Select

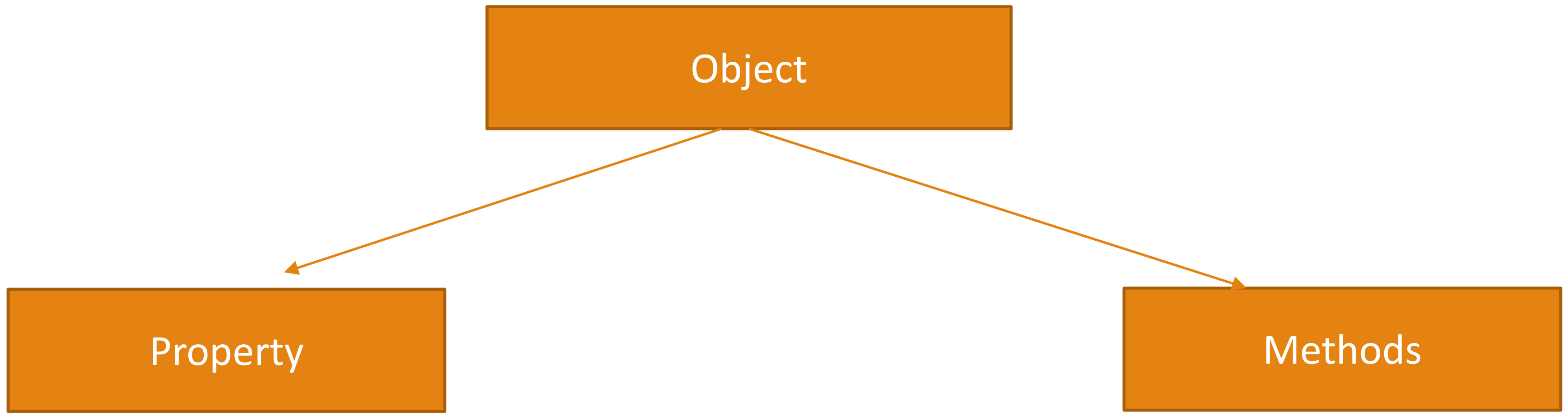
Sheets(“Sheet1”) .Range(“A1”).Select

Workbooks(“Book1.xlsm”).Sheets(“Sheet1”).Range(“A1”).Select

## Selecting the Range



## Object Model of Excel



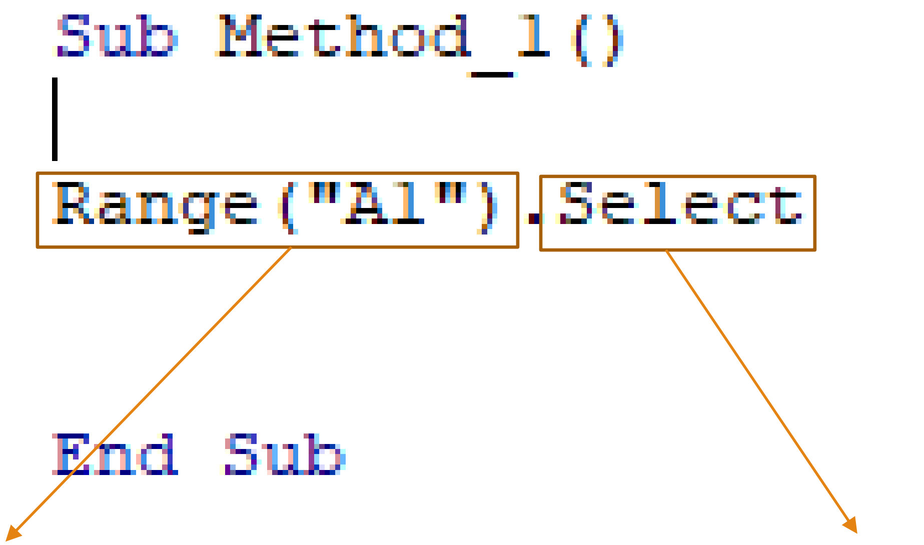
Object: Anything and everything in Excel is a Object. Ex: Workbook, Worksheet, Cell, Range, Chart, Shape, Pivot Table…

Method: A method is an **Action** you perform with an Object.
Ex: Sheets(“Sheet1”).Select

Property: Properties are the attributes that describe an object. An object’s Property determine how it looks, how it behaves, and even whether it is visible or hidden. Using VBA, you can do two things with an object’s Property
Ex: Sheets(“Sheet1”).Name = “Sales Info”
Using VBA, you can do two things with an object’s Property:

1. Read Property to take decisions
2. Change Property’s setting

Method: A method is an **Action** you perform with an Object.

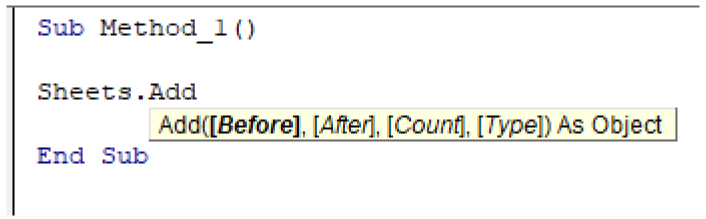

```
Sub Method_1()

Range("A1").Select

End Sub
```

Methods may have parameters, these parameters tell how to do the action.

Let's see "Add" method to Sheets object



```
Sub Method_1()

Sheets.Add
          Add([Before], [After], [Count], [Type]) As Object
End Sub
```

Below code insert 2 new sheets Before "Sheet1"

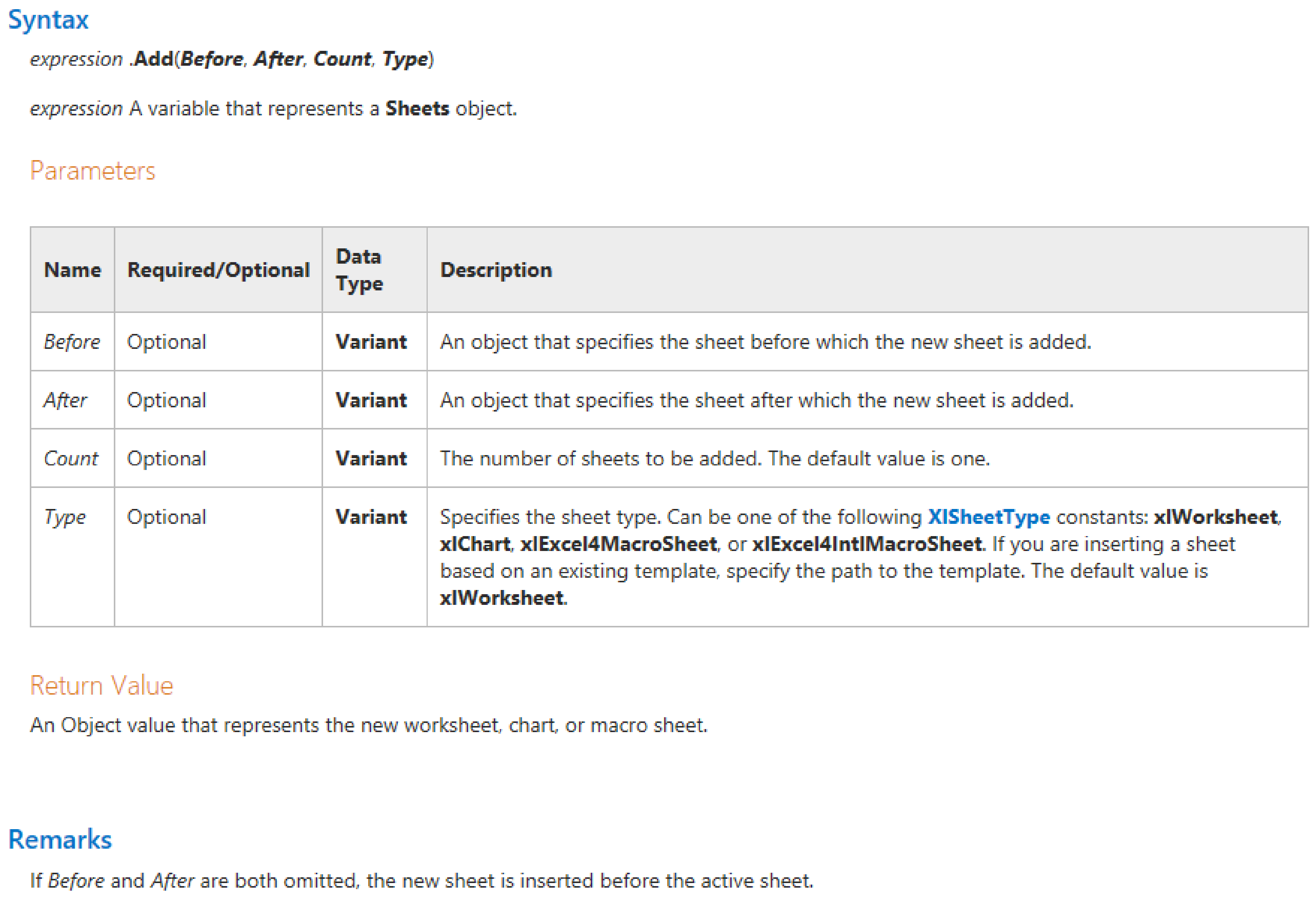
```
Sheets.Add Before:=Sheets("Sheet1"), Count:=2
```

Below code insert 3 new sheets After "Sheet1"

```
Sheets.Add After:=Sheets("Sheet3"), Count:=3
```

Below code insert 4 new sheets before to activesheet

```
Sheets.Add ,,4
```



## Syntax

*expression* .**Add**(***Before***, ***After***, ***Count***, ***Type***)

*expression* A variable that represents a **Sheets** object.

### Parameters

| Name | Required/Optional | Data Type | Description |
|---|---|---|---|
| *Before* | Optional | **Variant** | An object that specifies the sheet before which the new sheet is added. |
| *After* | Optional | **Variant** | An object that specifies the sheet after which the new sheet is added. |
| *Count* | Optional | **Variant** | The number of sheets to be added. The default value is one. |
| *Type* | Optional | **Variant** | Specifies the sheet type. Can be one of the following **XlSheetType** constants: **xlWorksheet**, **xlChart**, **xlExcel4MacroSheet**, or **xlExcel4IntlMacroSheet**. If you are inserting a sheet based on an existing template, specify the path to the template. The default value is **xlWorksheet**. |

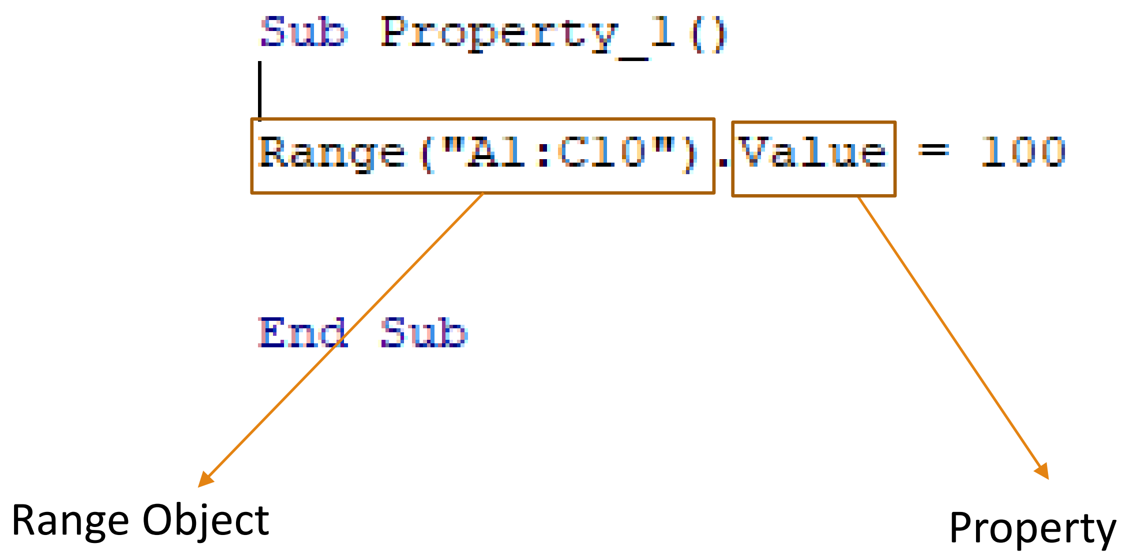
### Return Value

An Object value that represents the new worksheet, chart, or macro sheet.

## Remarks

If *Before* and *After* are both omitted, the new sheet is inserted before the active sheet.

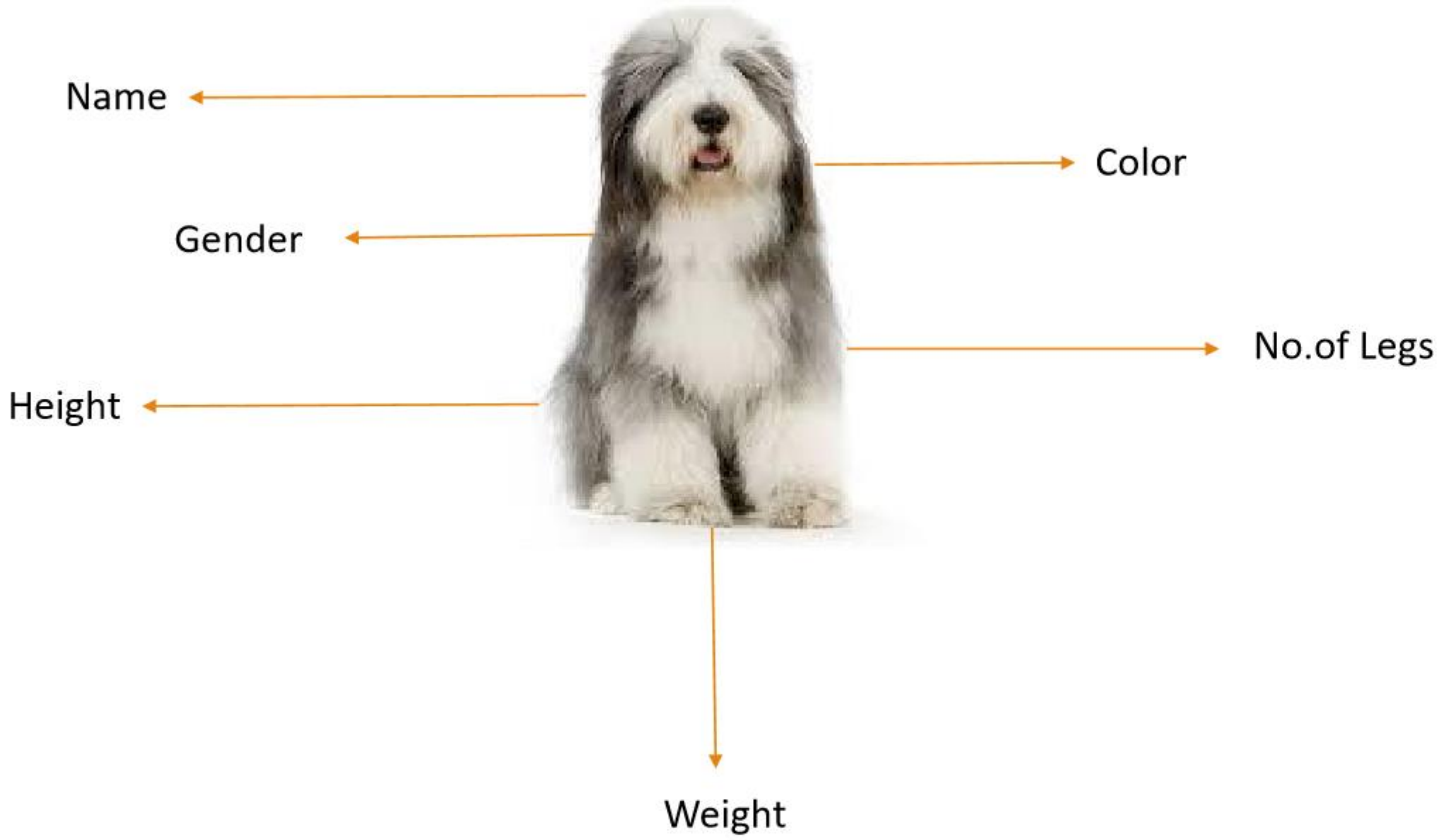
# Parameters of a Method



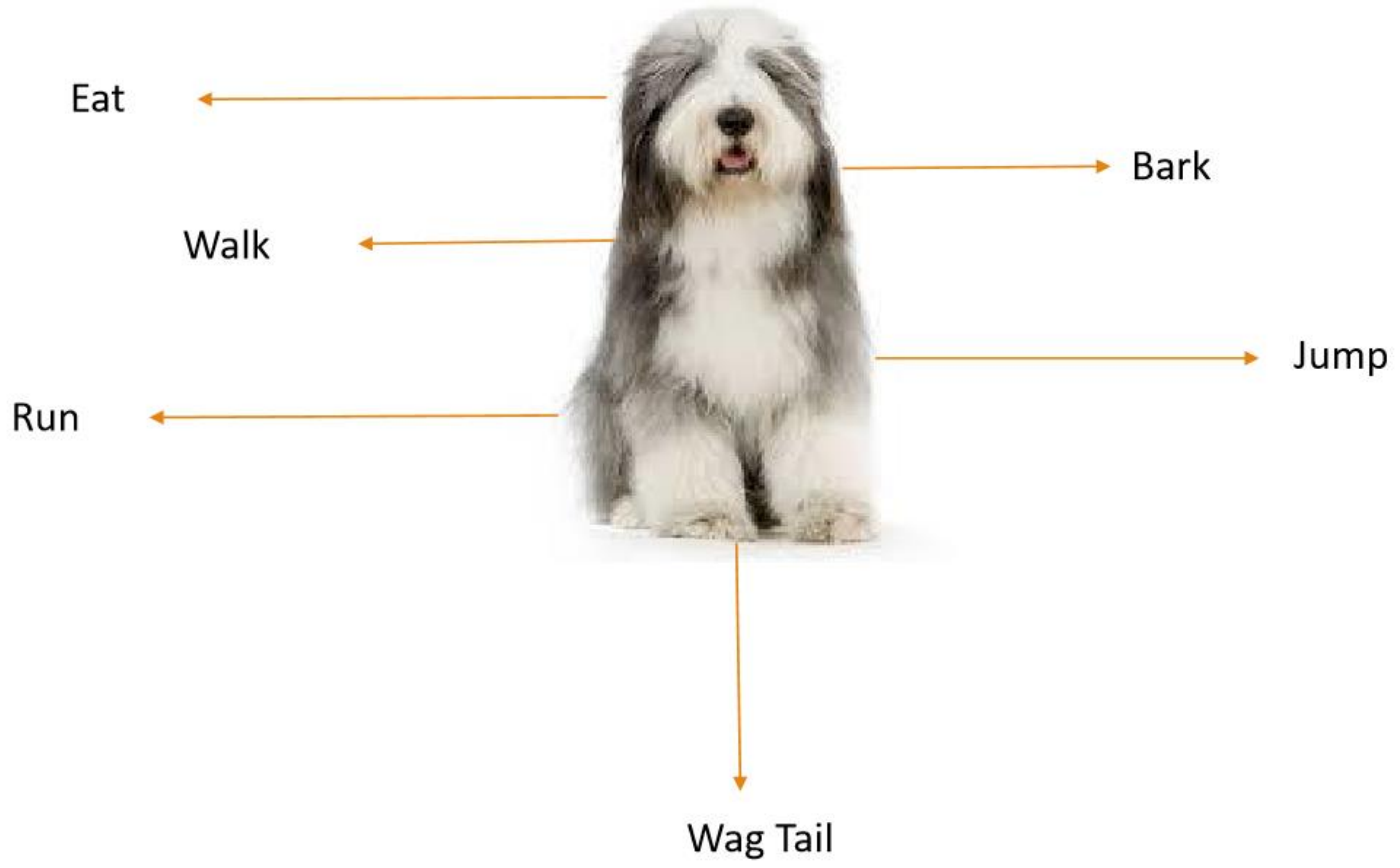
```
Sub Property_1()

Range("A1:C10").Value = 100

End Sub
```



## Properties of a Dog Object



## Methods of a Dog Object

```
Sub Range_Properties_Methods()
Range("a1").Activate
Range("a1").AddIndent = 2
Range("a1").HorizontalAlignment = xlCenter
Range("a1").Font.Name = "Wingdings"
Range("a1").Font.Bold = True
Range("a1").AddComment "VBA is Awesome"
Range("a1").Comment.Text "hi", 5
Range("a1").Font.Color = vbGreen
Range("a1").ClearContents
Range("a1").Font.Italic = True
Range("a1").ColumnWidth = 25
Range("a1").Copy
Range("b1").PasteSpecial
Range("a1").EntireColumn.Hidden = True
Range("a1").EntireRow.Hidden = False
Range("a1").Clear
Range("a1").Delete
Range("e14:g17").Merge
Range("D13:G13").FillRight
Range("j8").FlashFill
MsgBox Range("a1").Rows.Count
End Sub
```

## Find Properties and Methods of a Range

## Why Learning is Important?

1. Ride the Wave.

or

2. Get crushed by the Tide.

Remember that the best invest in this world is in yourself. So ensure you get trained/upskilled every year. This is needed so that you get paid more and respect more.