

**Assignment - 19**

1. What are the data types used in VBA?

Answer:

| **Data Type** | **Stored** | **Range of Values** |
| --- | --- | --- |
| Byte | 1 Byte | 0 to 255 |
| Integer | 2 Bytes | -32,768 to 32,767 |
| Single | 4 Bytes | -3.402823E38 to -1.401298E-45 for negative values, 1.401298E-45 to 3.402823E38 for positive values |
| Long | 4 Bytes | -2,147,483,648 to 2,147,483,648 |
| Double | 8 Bytes | -1.79769313486232e+308 to -4.94065645841247E-324 for negative values, 4.94065645841247E-324 to 1.79769313486232e+308 for positive values. |
| Decimal | 14 Bytes | +/-79,228,162,514,264,337,593,543,950,335 for no decimal points,+/-7.9228162514264337593543950335 for 28 places to the right of the decimal |
| Date | 8 Bytes | January 1, 100 to December 31, 9999 |
| Currency | 8 Bytes | -922,337,203,685,477.5808 to 922,337,203,685,477.5807 |
| String (variable length) | 10 bytes added to the string length | 0 to 2 billion characters |
| String (fixed length) | string length | 1 to approximately 65,400 |
| Variant (with numbers) | 16 bytes | Any numeric value up to the range of a Double |
| Variant (with characters) | 22 bytes + string length (24 bytes on 64-bit systems) | Same range as for variable-length String |
| Object | 4 Bytes | Object in VBA |
| Boolean | 2 Bytes | True or False |

2. What are variables and how do you declare them in VBA? What happens if you don’t declare a variable?

Answer: A variable is a location in your computer's memory that you define and then use to store values. This storage is temporary and the values are cleared when your macro ends.

You can name a variable something meaningful and specify the type of data that it will store. You can then access and change that variable's value as many times as you want in your Excel VBA code.

A constant is similar, however, once initialized, the value it stores cannot be changed in your VBA code.

## Declaring variables in Excel VBA

Creating variables in VBA is known as *declaring your variables*.

A variable declaration is made up of three parts:

1. The keyword Dim
2. The name of the variable
3. Its data type.

The declarations are the first lines of code you see in an Excel macro.

The code below declares four variables. Each variable is declared on a separate line.

Sub VariableExamples()

Dim companyID as String

Dim companyName as String

Dim numberOfProducts as Integer

Dim productPrice as Double

End Sub

When naming a variable, there are some rules to abide by (these are the same rules to take into account when naming macros).

* The variable name must not begin with a number.
* You cannot use special characters such as *%, &, !* or *@*.
* You cannot use spaces.
* A reserved keyword such as Dim, Public or Next cannot be used. These reserved words are important for other VBA operations.

It is good practice to define a data type for each of your variables. This specifies the type of data that the variable will store.

If a variable type is not defined then the Variant data type is used. This can handle any data type, but takes up more storage space.

There are a few common VBA variable types that you will see and use frequently. These are:

* String to store text values.
* Long and Integer to store whole numbers.
* Double to store numbers with decimals.
* Boolean to store TRUE and FALSE values.
* Object to store VBA objects such as worksheets and charts.

If you have multiple variables of the same data type, you can declare these on the same line.

In the code below, the two String variable types are declared on the same line.

Sub VariableExamples()

Dim companyID, companyName as String

Dim numberOfProducts as Integer

Dim productPrice as Double

End Sub

3. What is a range object in VBA? What is a worksheet object?

Answer:

Range is a property in VBA that helps specify a particular cell, a range of cells, a row, a column, or a three-dimensional range. In the context of the Excel worksheet, the VBA range object includes a single cell or multiple cells spread across various rows and columns.

For example, the range property in VBA is used to refer to specific rows or columns while writing the code. The code “Range(“A1:A5”).Value=2” returns the number 2 in the range A1:A5.

In VBA, macros are recorded

and executed to automate the Excel tasks. This helps perform the repetitive processes in a faster and more accurate way. For running the macros, VBA identifies the cells on which the called tasks are to be performed. It is here that the range object in VBA comes in use.

The Worksheet object is a member of the Worksheets collection. The Worksheets collection contains all the Worksheet objects in a workbook.

The Worksheet object is also a member of the Sheets collection. The Sheets collection contains all the sheets in the workbook (both chart sheets and worksheets).

## Example

Use [Worksheets](https://learn.microsoft.com/en-us/office/vba/api/excel.workbook.worksheets) (*index*), where *index* is the worksheet index number or name, to return a single Worksheet object. The following example hides worksheet one in the active workbook.

VB

Copy

Worksheets(1).Visible = False

The worksheet index number denotes the position of the worksheet on the workbook's tab bar. Worksheets(1) is the first (leftmost) worksheet in the workbook, and Worksheets(Worksheets.Count) is the last one. All worksheets are included in the index count, even if they are hidden.

The worksheet name is shown on the tab for the worksheet. Use the [Name](https://learn.microsoft.com/en-us/office/vba/api/excel.worksheet.name) property to set or return the worksheet name. The following example protects the scenarios on Sheet1.

VB

Copy

Dim strPassword As String

strPassword = InputBox ("Enter the password for the worksheet")

Worksheets("Sheet1").Protect password:=strPassword, scenarios:=True

When a worksheet is the active sheet, you can use the ActiveSheet property to refer to it. The following example uses the Activate method to activate Sheet1, sets the page orientation to landscape mode, and then prints the worksheet.

VB

Copy

Worksheets("Sheet1").Activate

ActiveSheet.PageSetup.Orientation = xlLandscape

ActiveSheet.PrintOut

This example uses the BeforeDoubleClick event to open a specified set of files in Notepad. To use this example, your worksheet must contain the following data:

* Cell A1 must contain the names of the files to open, each separated by a comma and a space.
* Cell D1 must contain the path to where the Notepad files are located.
* Cell D2 must contain the path to where the Notepad program is located.
* Cell D3 must contain the file extension, without the period, for the Notepad files (txt).

When you double-click cell A1, the files specified in cell A1 are opened in Notepad.

VB

Copy

Private Sub Worksheet\_BeforeDoubleClick(ByVal Target As Range, Cancel As Boolean)

'Define your variables.

Dim sFile As String, sPath As String, sTxt As String, sExe As String, sSfx As String

'If you did not double-click on A1, then exit the function.

If Target.Address <> "$A$1" Then Exit Sub

'If you did double-click on A1, then override the default double-click behavior with this function.

Cancel = True

'Set the path to the files, the path to Notepad, the file extension of the files, and the names of the files,

'based on the information on the worksheet.

sPath = Range("D1").Value

sExe = Range("D2").Value

sSfx = Range("D3").Value

sFile = Range("A1").Value

'Remove the spaces between the file names.

sFile = WorksheetFunction.Substitute(sFile, " ", "")

'Go through each file in the list (separated by commas) and

'create the path, call the executable, and move on to the next comma.

Do While InStr(sFile, ",")

sTxt = sPath & "\" & Left(sFile, InStr(sFile, ",") - 1) & "." & sSfx

If Dir(sTxt) <> "" Then Shell sExe & " " & sTxt, vbNormalFocus

sFile = Right(sFile, Len(sFile) - InStr(sFile, ","))

Loop

'Finish off the last file name in the list

sTxt = sPath & "\" & sFile & "." & sSfx

If Dir(sTxt) <> "" Then Shell sExe & " " & sTxt, vbNormalNoFocus

End Sub

4. What is the difference between worksheet and sheet in excel?

Answer:

The difference between Sheets and Worksheets

In essence, all Worksheets are Sheets, but not all Sheets are Worksheets. There are different types of Sheets:

* Worksheet – the sheet with the gridlines and cells
* Chart – the sheet which contains a single chart
* DialogSheet – an Excel 5 dialog sheet. These are effectively defunct as they have been replaced by VBA UserForms
* Macro sheets – A sheet containing [Excel 4 macros](https://exceloffthegrid.com/using-excel-4-macro-functions/). These were replaced by VBA in 1995.
* International Macro sheet – A sheet containing an internationally compatible Excel 4 macro (also replaced in 1995).

Since DialogSheets, and both forms of Macro sheets were replaced in the 90’s, we can pretty much ignore them. That leaves just two types of sheets we are likely to encounter: Charts and Worksheets.

5. What is the difference between A1 reference style and R1C1 Reference style? What are the advantages and disadvantages of using R1C1 reference style?

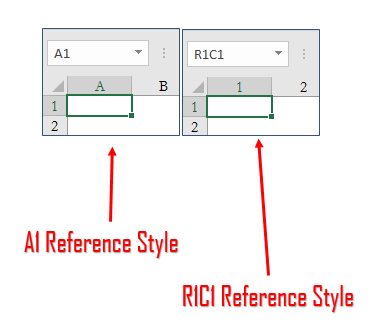
Answer:

## Difference Between A1 and R1C Reference Style

In the A1 reference style, you have the column name as an alphabet and the row name as a number and when you select the A1 cell that means you are in column A and row 1.

But in R1C1 both column and row are in numbers.

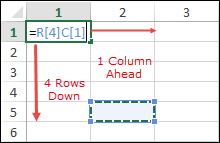
So, when you select cell A1 it shows you R1C1, which means row 1 and column 1, and if you go to A2 then it will be R2C1.



In the above two examples, you have the same active cell, but different cell addresses. **The real difference** comes when you write formulas and use a reference to other cells.

In R1C1, when you refer to a cell it creates the address of referred cell **using its distance from the active cell**.

For example, if you refer to cell B5 from cell A1 it will show the address of B5 as R[4]C[1].



Now, just think this way. Cell B5 is 4 rows down and 1 column ahead of cell A1, so that’s why its address is R[4]C[1].

But here’s the kicker. If you refer to the same cell from a different cell then its address will be different.

**The point to understand is**, In the R1C1 reference style, there is no permanent address for a cell (if you are using relative reference), so a cell’s address dependents ob from where you are referring to it.

Using the R1C1 reference is a realistic approach to working with cell references.

Ahead of this post, I will detail this approach. But first of all, let’s learn to activate it.

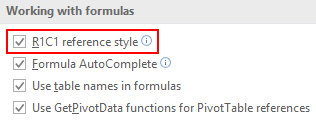
## How to Activate R1C1 Cell Reference in Excel – Simple Steps

To use R1C1, the first thing you need to do is to activate it and for this, you can use any of the below methods.

### From Excel Options

Please follow these simple steps to set the R1C1 reference as default.

1. Go to File Tab ➜ Option ➜ Formulas ➜ Working with formulas.
2. Tick mark “R1C1 Reference Style”.
3. Click OK.



### Using a VBA Code

If you are [macro savvy and want to save time](https://excelchamps.com/blog/useful-macro-codes-for-vba-newcomers/) then you can use the below macro code to toggle between cell reference styles.

Sub ChangeCellRef()

If Application.ReferenceStyle = xlA1 Then

Application.ReferenceStyle = xlR1C1

Else

Application.ReferenceStyle = xlA1

End If

End Sub

Once you make R1C1 your default reference style all the references in formulas in all the workbooks will change.

6. When is offset statement used for in VBA? Let’s suppose your current highlight cell is A1 in the below table. Using OFFSET statement, write a VBA code to highlight the cell with “Hello” written in it.

A B C

1. 25 354 362

2. 36 6897 962

3. 85 85 Hello

4. 96 365 56

5. 75 62 2662

Answer:

***V*BA Offset** function one may use to move or refer to a reference skipping a particular number of rows and columns. The arguments for this function in VBA are the same as those in the worksheet.

