**Assignment - 19**

***1. What are the data types used in VBA?***

**Below table shows various data type in VBA.**

|  |  |  |
| --- | --- | --- |
| Data Type | Stored | Data Type |
| 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?***

**What are variables ?**

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

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

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

**how to declare them in VBA?**

Creating variables in VBA is known as declaring our 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 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, these are the same rules to take into account when naming macros:

1. The variable name must not begin with a number.
2. we cannot use special characters such as %, &, ! or @.
3. we cannot use spaces.
4. 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:

1. String to store text values.
2. Long and Integer to store whole numbers.
3. Double to store numbers with decimals.
4. Boolean to store TRUE and FALSE values.
5. 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.

**When we don't declare the data type of the variable,** then VBA treats it as a Variant data type, and it is bound to accept any type of data integer like an integer, string, workbook, etc. In the same program, the variant variables can accept string values, integer values, and any other type as well.

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

* 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 VBA range property is similar to the worksheet property and has several applications.

**Worksheet object :**

In excel VBA, the worksheets object represents the collection of the worksheets which are open in the excel workbook. This is the reason we write it as plural- Worksheets Object, which means it contains more than one worksheet.

**By Name**

In excel, we can refer to a sheet using its name. By default, Excel creates sheets in natural number ordering. For example, Sheet1, Sheet2, etc. We can also give our own name to these sheets.

**By Number**

We can also use sheet numbers to refer to a sheet in excel. The two operations to refer to a sheet, Sheets() and Worksheets() work in different ways while accessing the sheets by number.

**By ActiveSheet Property**

In excel VBA, we can also refer to a sheet using the ActiveSheet property.

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).

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

All Worksheets are Sheets, but not all Sheets are Worksheets.

**Worksheets Object**

Worksheets Object is a collection of all the Worksheet objects in the specified or active workbook. Each Worksheet object represents a worksheet. 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).

**Sheets Object**

Sheets Object is a collection of all types of sheets in the specified or active workbook. The Sheets collection can contain Chart or Worksheet objects. Although today we only use 2 types of Sheets, Worksheets and Chart Sheets, there used to be 3 more types of Sheets, like Dialog Sheets or Macro Sheets. You may still have in your company old excel files that use them. In that case if you check the Sheets collection you'll seem them all there.

**Examples**

If a workbook has 4 worksheets and 1 chart sheet, in VBA:

Sheets.Count will include both types: 5

Worksheets.Count will include only worksheets: 4

**Note:** In VBA you should use Worksheets() instead of Sheets() when referencing a sheet from another sheet in a formula.

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

In Excel, there are two kinds of cell reference styles first is A1 and the second is R1C1.

1. In the A1 reference style, we have the column name as an alphabet and the row name as a number and when we select the A1 cell that means we are in column A and row 1. But in R1C1 both column and row are in numbers. So, when we select cell A1 it shows you R1C1, which means row 1 and column 1, and if we go to A2 then it will be R2C1.
2. In R1C1, when we refer to a cell it creates the address of referred cell using its distance from the active cell.

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

|  |
| --- |
| A1 Reference Style R1C1 Reference Style Reference Type |
| A1 R[-4]C[-1] Relative Reference |
| $A$1 R1C1 Absolute Reference |
| $A1 R[-4]C1 Relative Row and Absolute Column |
| A$1 R1C[-1] Absolute Row and Relative Column |

* One of the main advantages of R1C1 references is that they make it easy to use relative references in formulas and functions. By using relative references, you can create formulas and functions that can be easily copied and pasted to other cells without having to manually adjust the cell references.
* Another advantage of R1C1 references is that they provide a consistent way of referring to cells, regardless of their position on the worksheet. This can be particularly useful when working with large and complex worksheets, as it can help to reduce errors and make it easier to understand and maintain the formulas and functions.
* R1C1 references can also be useful when working with macros and other automated processes in Excel. By using R1C1 references in your code, you can create more flexible and dynamic macros that can adapt to changes in the worksheet structure and layout.
* While the A1 reference style is the default in Excel and is generally easier to read and understand, R1C1 references can be a powerful tool for working with formulas, functions, and macros in Excel

***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

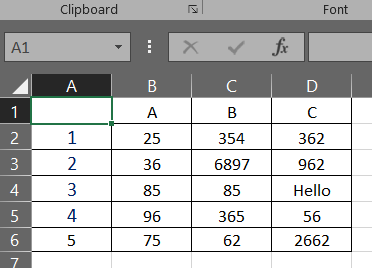
5 75 62 2662

**OFFSET –** It is a reference function in Excel. The OFFSET function returns a reference to a range that is a specific number of rows and columns from another range or cell. It is one of the most important notions in Excel.

To use OFFSET function in VBA, we have to use VBA RANGE object because OFFSET refers cells and from that RANGE object we can use OFFSET function. In Excel, RANGE refers to the range of cells.

**Above Example :-**

**1.** Suppose your current highlight cell is A1.



**A macro using OFFSET statement to highlight the cell with “Hello” written in it :-**

*Private Sub select\_hello()*

*Range("A1").Offset(3, 3).Select*

*End Sub*

