ADVANCE EXCEL ASSIGNMENT 19

1. What are the data types used in VBA?

The following table shows the supported data types, including storage sizes and ranges.

Data type	Storage size	Range
Boolean	2 bytes	True or False
Byte	1 byte	0 to 255
Collection	Unknown	Unknown
Currency (scaled integer)	8 bytes	-922,337,203,685,477.5808 to 922,337,203,685,477.5807
Date	8 bytes	-657,434 (January 1, 100), to 2,958,465 (December 31, 9999)
Decimal	14 bytes	+/- 79,228,162,514,264,337,593,543,950,335 with no decimal point +/-7.9228162514264337593543950335 with 28 places to the right of the decimal Smallest non-zero number is+/- 0.0000000000000000000000000001
Dictionary	Unknown	Unknown
Double (double-precision floating-point)	8 bytes	-1.79769313486231E308 to -4.94065645841247E-324 for negative values 4.94065645841247E-324 to 1.79769313486232E308 for positive values
Integer	2 bytes	-32,768 to 32,767
Long (Long integer)	4 bytes	-2,147,483,648 to 2,147,483,647
LongLong (LongLong integer)	8 bytes	-9,223,372,036,854,775,808 to 9,223,372,036,854,775,807 Valid on 64-bit platforms only.
LongPtr (Long integer on 32-bit systems, LongLong integer on 64-bit systems)	4 bytes on 32-bit systems 8 bytes on 64-bit systems	-2,147,483,648 to 2,147,483,647 on 32-bit systems -9,223,372,036,854,775,808 to 9,223,372,036,854,775,807 on 64-bit systems
Object	4 bytes	Any Object reference
Single (single-precision floating-point)	4 bytes	-3.402823E38 to -1.401298E-45 for negative values

Data type	Storage size	Range
		1.401298E-45 to 3.402823E38 for positive values
String (variable-length)	10 bytes + string length	0 to approximately 2 billion
String (fixed-length)	Length of string	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
User-defined (using Type)	Number required by elements	The range of each element is the same as the range of its data type.

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

The word "variable" defines variables as the name of memory in your location, which holds some value. You can pass a value in a code based on the variable type. The value will be used while executing the code and you will get the output.

To declare a variable in code, you should assign a name to that variable. You can assign any name to a variable. However, selecting a variable name that relates to data is advisable so that other users can understand it easily.

When declaring variables, you usually use a Dim statement. A declaration statement can be placed within a procedure to create a procedure-level variable. Or it may be placed at the top of a module, in the Declarations section, to create a module-level variable.

Dim strName As String

If this statement appears within a procedure, the variable strName can be used only in that procedure. If the statement appears in the Declarations section of the module, the variable strName is available to all procedures within the module, but not to procedures in other modules in the project.

Public strName As String

Variables can be declared as one of the following data types: Boolean, Byte, Integer, Long, Currency, Single, Double, Date, String (for variable-length strings), String * length (for fixed-length strings), Object, or Variant. If you don't specify a data type, the Variant data type is assigned by default. You can also create a user-defined type by using the Type statement.

You can declare several variables in one statement. To specify a data type, you must include the data type for each variable.

In the following statement, the variables intX, intY, and intZ are declared as type Integer.

Dim intX As Integer, intY As Integer, intZ As Integer

In the following statement, intX and intY are declared as type Variant, and only intZ is declared as type Integer.

Dim intX, intY, intZ As Integer

You don't have to supply the variable's data type in the declaration statement. If you omit the data type, the variable will be of type Variant. The shorthand to declare x and y as Integer in the statement above is:

Dim intX%, intY%, intZ as Integer

The shorthand for the types is: % -integer; & -long; @ -currency; # -double; ! -single; $ -string

You can implicitly declare a variable in Visual Basic simply by using it in an assignment statement. All variables that are implicitly declared are of type Variant. Variables of type Variant require more memory resources than most other variables. Your application will be more efficient if you declare variables explicitly and with a specific data type. Explicitly declaring all variables reduces the incidence of naming-conflict errors and spelling mistakes.

If you don't want Visual Basic to make implicit declarations, you can place the Option Explicit statement in a module before any procedures. This statement requires you to explicitly declare all variables within the module. If a module includes the Option Explicit statement, a compile-time error will occur when Visual Basic encounters a variable name that has not been previously declared, or that has been spelled incorrectly.

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.

In VBA, the worksheet object represents a single worksheet that is a part of the workbook’s worksheets (or sheets) collection. Using the worksheet object, you can refer to the worksheet in a VBA code, and refer to a worksheet you can also get access to the properties, methods, and events related to it.

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

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 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 see them all there.

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

R1C1 references are an alternative way of referring to cells in Excel. Instead of using the traditional A1 reference style, which refers to cells by their column letter and row number (e.g. A1, B2, C3), R1C1 references use the notation R[row]C[column], where [row] and [column] are the relative row and column numbers of the cell being referred to.

For example, the cell in the first row and first column of a worksheet would be referred to as R1C1, while the cell in the third row and fourth column would be referred to as R3C4. This notation can be particularly useful when working with formulas and functions that involve relative references, as it allows you to easily refer to cells relative to the current cell.

When you use an R1C1 reference in a formula or function, Excel will interpret the reference based on the relative position of the cell containing the formula or function. For example, if you enter the formula "=R[1]C[1]" in cell A1, Excel will interpret this as a reference to the cell in the second row and second column (i.e. cell B2). Similarly, if you enter the formula "=R[-1]C[-1]" in cell B2, Excel will interpret this as a reference to the cell in the first row and first column (i.e. cell A1).

Advantages of R1C1 reference:

Relative References: 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.

Consistency: 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.

Automation: 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.

Disadvantages of R1C1reference:

Unless you have a reason to use the R1C1 reference style notation, I would recommend sticking to the default A1 reference style. A1 style it’s easier to use, And since almost all the Excel users use it, it makes it easier for you to share your work with other people. One instance where you may want to use the R1C1 notation is when you want to debug formulas and identify errors.

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.

VBA 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. In VBA, we cannot directly enter the word OFFSET. Instead, we need to use the VBA RANGE object first. Then, from that range object, we can use the OFFSET property. In Excel, the range is nothing but a cell or range of the cell. Since OFFSET refers to cells, we need to use the object RANGE first, and then we can use the OFFSET method.

Syntax

expression.**Offset** (*RowOffset*, *ColumnOffset*)

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

Parameters

Name	Required/Optional	Data type	Description
RowOffset	Optional	**Variant**	The number of rows—positive, negative, or 0 (zero)—by which the range is to be offset. Positive values are offset downward, and negative values are offset upward. The default value is 0.
ColumnOffset	Optional	**Variant**	The number of columns—positive, negative, or 0 (zero)—by which the range is to be offset.

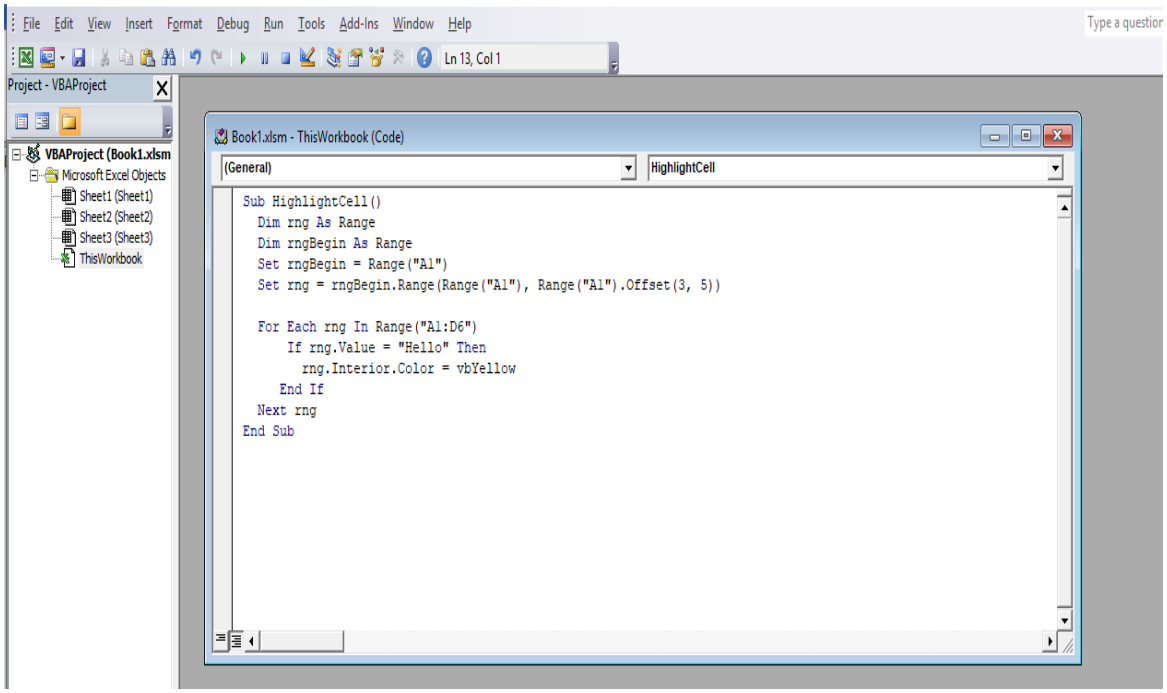
Name	Required/Optional	Data type	Description
			Positive values are offset to the right, and negative values are offset to the left. The default value is 0.

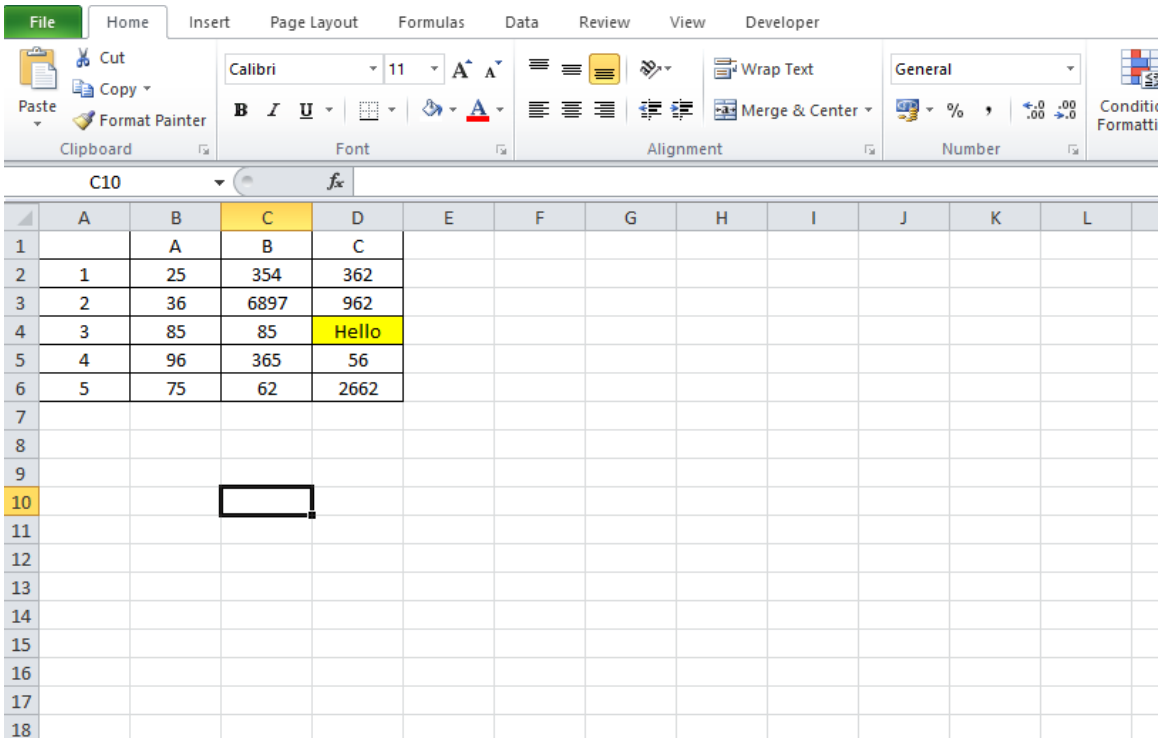
Using OFFSET statement, write a VBA code to highlight the cell with “Hello” written in it.

	A	B	C
1	25	354	362
2	36	6897	962
3	85	85	Hello
4	96	365	56
5	75	62	2662

Code is given below:

```
Sub HighlightCell()
  Dim rng As Range
  Dim rngBegin As Range
  Set rngBegin = Range("A1")
  Set rng = rngBegin.Range(Range("A1"), Range("A1").Offset(3, 5))
  For Each rng In Range("A1:D6")
     If rng.Value = "Hello" Then
       rng.Interior.Color = vbYellow
    End If
  Next rng
End Sub
```





	A	B	C	D	E	F	G	H	I	J	K	L
1		A	B	C								
2	1	25	354	362								
3	2	36	6897	962								
4	3	85	85	Hello								
5	4	96	365	56								
6	5	75	62	2662								
7												
8												
9												
10												
11												
12												
13												
14												
15												
16												
17												
18												