Q1. What are the data types used in VBA?

Ans: Below is a list of **data types used in VBA**

- Integer: Used to store number values that won't take on decimal form.
- Single: Used to store number values that may take on decimal form. Can also contain integers.
- Double: A longer form of the single variable. Takes up more space, but needed for larger numbers.
- Date: Stores date values.
- String: Stores text. Can contain numbers, but will store them as a text (calculations cannot be performed on numbers stored as a string)
- Boolean: Used to store binary results (True/False, 1/0)

Again, there are other data types, but these are the most commonly used for creating macros.

Q2. What are variables and how do you declare them in VBA? What happens if you don't declare a variable?

Ans: A variable is defined as storage in the computer memory that stores information to execute the VBA code.

The type of data stored in the variable depends on the type of data of the variable. For example, if a user wants to store integers in the variable, the data type will be an integer. A variable differs from a constant in that while the variable changes when the code is executed, the constant never changes.

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.

Undeclared variables are undefined because they simply do not exist. As described earlier, assigning a value to an undeclared variable does not cause an error; instead, it implicitly declares the variable in the global scope.

The second kind of undefined variable is one that has been declared but has never had a value assigned to it. If you read the value of one of these variables, you obtain its default value, undefined. This type of undefined variable might more usefully be called unassigned, to distinguish it from the more serious kind of undefined variable that has not even been declared and does not exist.

Q3. What is a range object in VBA? What is a worksheet object?

Ans: Excel's VBA Range is an object. Objects are what is manipulated by Visual Basic for Applications.

More precisely, you can **use** the Range object to represent a range within a worksheet. This means that, by using Excel's VBA Range object, you can refer to:

- A single cell.
- A row or a column of cells.
- A selection of cells, regardless of whether they're contiguous or not.
- A 3-D range.

As you can see from the above, the size of Excel's VBA Range objects can vary widely. At the most basic level, you can be making reference to a single (1) cell. On the other extreme, you have the possibility of referencing all of the cells in an Excel worksheet.

Despite this flexibility when referring to cells within a particular Excel worksheet, Excel's VBA Range object does have some limitations. The most relevant is that you can only use it to refer to a single Excel worksheet at a time. Therefore, in order to refer to ranges of cells in different worksheets, you must use separate references for each of the worksheets.

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.

Q4. What is the difference between worksheet and sheet in excel?

Ans: I use the terms *Sheet* and *Worksheet* interchangeably when talking about Excel, I think most users do. Google also appears to think they are the same thing; If I search for "How to loop through Sheets with Excel VBA", all the results returned on the first page refer to Worksheets.
Yet, Sheets and Worksheets from a VBA perspective are definitely not the same. Unfortunately, most of the VBA code I write doesn't really consider the differences

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. 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.

Q5. What is the difference between A1 reference style and R1C1 Reference style? What are the advantages and disadvantages of using R1C1 reference style?

Ans: The real difference comes when you want to write formulas. Let's take a simple example adding two columns together. I am displaying the formulas so the differences are clearer.

First, the familiar A1 style:

And now for R1C1.

At first this looks quite horrific and ten times more complicated that the A1 style. Before I break down how it works, what do you notice between the two styles (other than the obvious one)?

Each formula is different in A1 style: A2+B2, A3+B3…etc.

Whereas using R1C1, they are all identical. So this potentially means that wherever you write a formula in that column it will be same, no need to think about which row or column you are in. This is particularly helpful when you are writing VBA code.

R1C1

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.

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

1. Relative R1C1 Reference

Using relative reference in R1C1 is quite simple. In this reference style, when you refer to a cell, it creates the address of the referred cell using its distance from the active cell. If you want to go to a column on the right side of the active cell, the number will be positive, or if the left side then a negative number. And same for the row, a positive number for the below and negative for the above active row.

2. Absolute R1C1 Reference

I’m sure you have noticed this thing in the above example that row and column numbers have square brackets. Let me show you what happens when you don’t use square brackets in the cell reference.