Advance Excel Assignment 20

(iNeuron.ai)

By Manjunath Pai

1. Use the above data and write a VBA code using the following statements to display in the next column if the number is odd or even

a. IF ELSE statement

b. Select Case statement

c. For Next Statement

In VBA, the **Select Case Statement** is an alternative to the **If-Then statement**, allowing you to test if conditions are met, running specific code for each condition. The Select Statement is preferable to the If Statement when there are multiple conditions to process.

Excel VBA has the IF Then Else construct that you can use to analyse multiple conditions and execute codes based on these conditions.
Another similar construct that allows you to check for multiple conditions is the **SELECT CASE** statement.

VBA FOR NEXT is **a fixed loop that uses a counter to run iterations**. In simple words, you need to specify the number of times you want to run the loop, and once it reaches that count

loop, it will stop automatically. That's why it is a fixed loop and most popular among VBA developers.

2. What are the types of errors that you usually see in VBA?

Syntax Errors

Syntax errors are those that appear while you write code. If you're using Visual Studio, Visual Basic checks your code as you type it in the **Code Editor** window and alerts you if you make a mistake, such as misspelling a word or using a language element improperly. If you compile from the command line, Visual Basic displays a compiler error with information about the syntax error. Syntax errors are the most common type of errors. You can fix them easily in the coding environment as soon as they occur.

Run-Time Errors

Run-time errors are those that appear only after you compile and run your code. These involve code that may appear to be correct in that it has no syntax errors, but that will not execute. For example, you might correctly write a line of code to open a file. But if the file does not exist, the application cannot open the file, and it throws an exception. You can fix most run-time errors by rewriting the faulty code or by using exception handling, and then recompiling and rerunning it.

Logic Errors

Logic errors are those that appear once the application is in use. They are most often faulty assumptions made by the developer, or unwanted or unexpected results in response to user actions. For example, a mistyped key might provide incorrect information to a method, or you may assume that a valid value is always supplied to a method when that is not the case. Although logic errors can be handled by using exception handling (for example, by testing whether an argument is Nothing and throwing an Argument Null Exception), most commonly they should be addressed by correcting the error in logic and recompiling the application.

4. How do you handle Runtime errors in VBA?

Runtime errors occur as your macro runs, and typically result from specific conditions present at that time. For example, if you prompt the user for a host name, and attempt to connect to that host, but the host is not available, the Connect method fails and Visual Basic generates a runtime error.

You should always include some form of error handling in your macros to deal with runtime errors. Without any error handling, a runtime error causes a macro to stop immediately, and gives the user little information**.**

5. Write some good practices to be followed by VBA users for handling errors

VBA Error Handling refers to the process of anticipating, detecting, and resolving VBA Runtime Errors. The VBA Error

Handling process occurs when writing code, before any errors actually occur.

VBA Runtime Errors are errors that occur during code execution. Examples of runtime errors include:

- Referencing a non-existent workbook, worksheet, or other object (Run-time Error 1004)
- Invalid data ex. referencing an Excel cell containing an error (Type Mismatch – Run-time Error 13)
- Attempting to divide by zero

What is UDF? Why are UDF's used? Create a UDF to multiply 2 numbers in VBA

With VBA, you can create a custom Function (also called a User Defined Function) that can be used in the worksheets just like regular functions.

These are helpful when the existing Excel functions are not enough. In such cases, you can create your own custom User Defined Function (UDF) to cater to your specific needs.

In this tutorial, I'll cover everything about creating and using custom functions in VBA.

Let me create a simple user-defined function in VBA and show you how it works.

The below code creates a function that will extract the numeric parts from an alphanumeric string.

Below is how this function – *GetNumeric* – can be used in Excel

Now before I tell you how this function is created in VBA and how it works, there are a few things you should know:

- When you create a function in VBA, it becomes available in the entire workbook just like any other regular function.
- When you type the function name followed by the equal to sign, Excel will show you the name of the function in the list of matching functions. In the above example, when I entered =Get, Excel showed me a list that had my custom function.