How to automate Vlookup with Excel VBA

Vlookup is an essential function in Excel and has become an important part of data processing. It provides a number of functions that you can combine with a comprehensive database.

But if you want to automate this function, you can, especially when you know how to use the automation language available in Excel, VBA. Here's how you can automate **the vlookup function in Excel VBA** .

Vlookup function formula in Excel

```
=vlookup(lookup_value, lookup_array, column_index, exact_match/partial_match)
```

Meaning of the arguments in the formula:

1. `lookup_value`: The base value that you want to look up in the array.
2. `lookup_array`: Source data, lookup_value will be used as reference.
3. `column_index`: The column returns the value.
4. `exact_match/partial_match`: Use 0 or 1 to specify the type of association.

Vlookup function of VBA in Excel

The Vlookup function has not much different usage when you use it directly in Excel or via VBA. Whether you're a data analyst dealing with rows of information or a warehouse manager who is constantly working with new products, the vlookup function is useful.

When working with a dynamic lookup range, automating formulas is best, to avoid having to manipulate a formula multiple times in Excel. By automating the Vlookup function in VBA, you can perform multi-column calculations with one click.

Illustration

This dataset has two parts: a lookup table and destination data points. The lookup table contains 3 columns, with data in the two fields Sub-Category and Product Name:

Picture 1 of How to automate Vlookup with Excel VBA

The destination table has the right columns, no subdirectories or product name data. Note that the values ??in the **Order ID** column also appear in the same column in the lookup table:

Picture 2 of How to automate Vlookup with Excel VBA

1. Make the necessary settings

Start by opening the code editor (press **Alt + F11** or navigate to the **Developer** tab) in a new Excel workbook and add the module to start coding. In the code editor, add the following lines at the top of the programming window to create a sub-routine:

```
Sub vlookup_fn1()End Sub
```

sub-routine is a container for VBA code and is necessary to run it successfully. Another alternative is to create a VBA userform to make the UI more interactive.

2. Declare variables and create scopes of references

First, you need to declare the data types of the variable using the Dim command:

```
Dim i as integer, lastrow as long
```

Next, create a variable **lastrow** , to store the value of the last populated row in the lookup range. **The lastrow** function uses the **end(xldown)** function to compute the last row reference in the specified range.

In this example, the **lastrow** variable contains the value of the last filled row in the lookup range, specifically 17 here.

```
lastrow = Sheets("Sheet2").Range("F3").End(xlDown).Row
```

In the above example, the destination table extends from A2:C17, and the source table extends from F2:H17. The vlookup formula will iterate through each value in column A, look it up in the source table, and paste the combined entries in columns B and C.

However, before entering the loop, you must store the range values ??in a new variable (**my_range**) as follows:

```
my_range = Sheets("Sheet2").Range("F3:H" & lastrow)
```

You must explicitly specify the starting cell reference (F3) and the ending column reference (H). VBA automatically adds row values ??to complete the lookup array.

Picture 3 of How to automate Vlookup with Excel VBA

3. Write a loop to cycle through each lookup value

Before writing the loop, specify the starting row value, 2. Specifying a start value for the loop is necessary when you are dealing with dynamic components. In the destination table, row 2 is the first row of data. Change this value if the data starts in another row.

```
i = 2
```

You can use a **do-while** loop to process each row, starting at row 2 and ending at row 17. Like the rest of the code, this aspect is flexible; The loop will run until the condition is false. Use a condition to check for a non-

blank value in the first cell of the current row.

```
Do While len(Sheets("Sheet2").Cells(i, 1).value) > 0
```

Inside the loop, you can call the Vlookup function to populate the cells:

```
Sheets("Sheet2").Cells(i, 2).Value = _ Application.WorksheetFunction.VLookup(She
```

These commands set the values ??of cells in the current row, in columns 2 and 3, respectively. They use the WorksheetFunction object to call the Vlookup function, passing the corresponding value from column 1 to search. They also pass the range you initially defined and the index of the associated column to fetch the final value.

The vlookup code is ready, but you still need to increment the row variable each time in the loop:

```
i = i + 1
```

Each time the loop runs, it increments the **i** value by **1** . Remember, the starting row value is 2, the increment occurs each time the loop runs. **Use the loop** keyword to set the end of this block of code.

When running the entire code, it populates the results in both columns, based on the values ??in the source table.

Picture 4 of How to automate Vlookup with Excel VBA

Here is the entire reference code:

```
Sub vlookup_fn1() Dim i As Integer lastrow = Sheets("Sheet2").Range("F3").End(x
```

4. Create a button to run the code

You can create a button in an Excel sheet to run it with one click. Insert the desired shape on the blank Excel sheet, right click on it and click the **Assign Macro** option .

Picture 5 of How to automate Vlookup with Excel VBA

In the dialog box, select the name of the sub-routine and click **OK** .

Picture 6 of How to automate Vlookup with Excel VBA

When you want to run the code, click the button to see how the data is populated immediately.

Above is **how to use Excel VBA to automate lookup functions** . Hope the article is useful to you.

You finished reading the article "**How to automate Vlookup with Excel VBA**" edited by the **TipsMake** team. We hope this article has provided you with many useful tech tips and tricks. You can search for similar articles on tips and guides. Thank you for reading and for following us regularly.