How to Remove the First Character from a String in VBA: A Simple Guide

Authored by

stats writer

November 20, 2025

When working with data processing or automation in Microsoft Excel, it is frequently necessary to manipulate text, or string, values. A common requirement is the removal of the initial character from a given string. In VBA (Visual Basic for Applications), this task is efficiently handled using intrinsic functions designed specifically for text operations.

While the initial concept might suggest using the Mid function, which can extract a section of a string based on starting position and length, the most straightforward method often involves combining the Right function and the Len function. The fundamental principle is to calculate the total length of the original string, subtract one (to exclude the first character), and then extract the resulting characters from the right side of the string.

Understanding String Manipulation in VBA

String manipulation is a cornerstone of effective automation using VBA. Data retrieved from external sources, databases, or user input often contains unwanted leading characters--such as identifiers, formatting codes, or redundant markers--that must be removed before the data can be analyzed or presented. VBA provides a powerful suite of functions specifically designed to manage these text operations efficiently, ensuring data integrity and cleaning processes are robust and repeatable.

For the specific task of removing the very first character of a text sequence, we require a method that can calculate the exact number of remaining characters and then isolate them. This approach avoids hardcoding lengths and ensures the solution works dynamically regardless of the length of the source string. The combined usage of the Right function and the Len function is the standard, most reliable technique for this operation, especially when dealing with cell ranges in an Excel spreadsheet.

The core concept relies on determining the total length of the string using Len, subtracting one (to account for the character we wish to discard), and then feeding that calculated length into the Right function. This allows the macro to extract every character except the first one. This dynamic calculation ensures that if the input string is 5 characters long, the result is the rightmost 4 characters, or if it is 50 characters long, the result is the rightmost 49 characters, maintaining precision across varied input data.

The Primary Method: Utilizing the VBA Right Function

The most effective and commonly used syntax to remove the first character from a string using VBA involves the combination of the Right function and the Len function. This method is highly recommended due to its clarity and efficiency in bulk data processing within Microsoft Excel environments.

```
Sub RemoveFirstChar()

Dim i As Integer
Dim myString As String

For i = 2 To 11
myString = Range("A" & i)
Range("B" & i) = Right(myString, Len(myString) - 1)
Next i

End Sub
```

This specific block of code is designed to iterate through a defined range within an Excel sheet. It systematically processes each cell, removing the leading character. Specifically, this procedure targets text strings found in the range **A2:A11** and deposits the resulting modified strings into the corresponding cells in the range **B2:B11**. This setup is ideal for scenarios where the source data must be kept intact while the processed results are stored elsewhere for verification or further steps.

Step-by-Step Breakdown of the Core Code

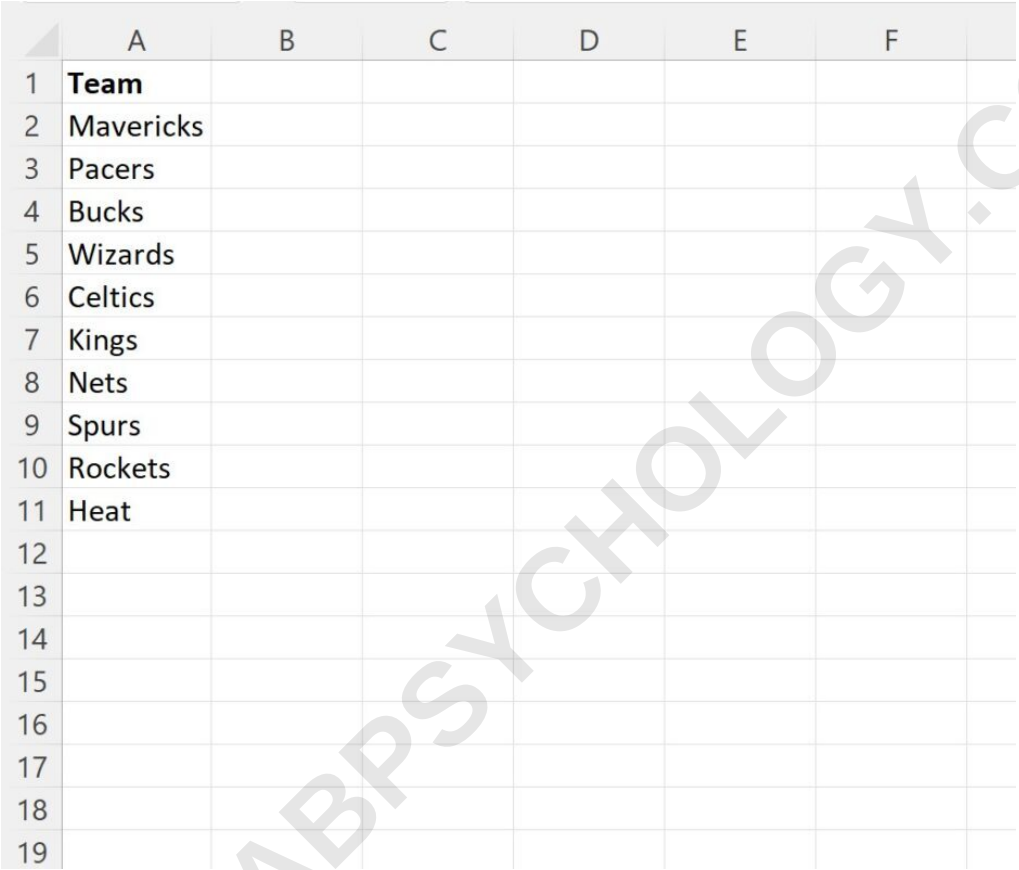
To fully understand the automation process, it is essential to analyze the components of the core manipulation line: **Range("B" & i) = Right(myString, Len(myString) - 1)**. This line executes the entire operation during each loop iteration. The process unfolds in a specific sequence, demonstrating the power of nested functions in VBA programming.

The **Len(myString)** function is executed first. It meticulously calculates the total number of characters present in the current string stored in the variable **myString**. If the string is "Example", the length is 7.
Next, **Len(myString) - 1** is calculated. This step creates the crucial length parameter needed for the extraction, subtracting one from the total length. Using our example "Example", the resulting calculation is 7 - 1, yielding 6.
Finally, the Right function is called: **Right(myString, 6)**. This function extracts the specified number of characters (6) starting from the right end of the original string. Since we excluded the first character, the output is "xample".

This dynamic calculation ensures that regardless of whether the cell contains a short word or a long sentence, the first character is accurately removed, providing a reliable and robust solution for data cleaning tasks within your automated routines. Utilizing a loop structure (**For i = 2 To 11**) allows this operation to be scaled easily across thousands of rows if necessary, simply by adjusting the loop boundaries.

Practical Implementation Example: Removing the First Character from an Excel Range

To illustrate this concept in a tangible setting, let us consider a dataset where we have inadvertently added an index number or a placeholder character at the start of meaningful data entries. Suppose we have the following list of basketball team names in Excel, each prefixed by an unwanted numeric digit:



	A	B	C	D	E	F
1	**Team**					
2	Mavericks					
3	Pacers					
4	Bucks					
5	Wizards					
6	Celtics					
7	Kings					
8	Nets					
9	Spurs					
10	Rockets					
11	Heat					
12						
13						
14						
15						
16						
17						
18						
19						

Our objective is to execute a cleaning operation that removes this leading numeric character from each team name. This is a very common scenario encountered in data imports where initial formatting needs rectification. Since we want to process a list of items rather than a single string, we must employ a VBA macro utilizing a loop structure to apply the operation across the entire specified column range.

We can create the following macro, which mirrors the structure discussed previously, designed specifically to iterate from row 2 to row 11 in column A, performing the desired string truncation:

Sub RemoveFirstChar()

```
Dim i As Integer
Dim myString As String

For i = 2 To 11
myString = Range("A" & i)
Range("B" & i) = Right(myString, Len(myString) - 1)
Next i

End Sub
```

Visualizing the Results of the VBA Macro

Once the macro **RemoveFirstChar** is executed within the VBA editor, the script will rapidly loop through all ten cells in column A. It extracts the necessary substring and writes the cleaned data to column B. This procedure is instantaneous for small datasets, but the speed advantage of VBA becomes particularly noticeable when processing hundreds or thousands of rows compared to using manual formulas or external text manipulation tools.

When we run this macro, we receive the following output in our worksheet. Notice how the leading numeric digit has been successfully removed from every team name, resulting in clean, usable text data in the new column:

	A	B	C	D	E	F
1	**Team**					
2	Mavericks	avericks				
3	Pacers	acers				
4	Bucks	ucks				
5	Wizards	izards				
6	Celtics	eltics				
7	Kings	ings				
8	Nets	ets				
9	Spurs	purs				
10	Rockets	ockets				
11	Heat	eat				
12						
13						
14						
15						
16						
17						
18						

As clearly demonstrated, Column B displays each of the strings originally held in column A, now accurately cleaned with the first character permanently removed. This successful transformation validates the use of the combined Right and Len functions for batch string processing.

Adapting the Code to Remove 'N' Starting Characters

The flexibility of this string manipulation technique extends beyond merely removing a single character. If your data requires the removal of the first 'N' characters--for instance, if every entry starts with a five-digit code that needs to be truncated--the VBA code can be trivially adjusted. The critical element to modify is the subtraction value within the Len calculation.

If you would instead like to remove the first N characters from a string, you simply change the **1** in the **Len(myString) - 1** expression to the desired number of leading characters you wish to discard. For example, to remove the first **2** characters from a string, we adjust the subtraction factor accordingly in the macro below:

```
Sub RemoveFirstTwoChar()

Dim i As Integer
Dim myString As String
```

```
For i = 2 To 11
myString = Range("A" & i)
Range("B" & i) = Right(myString, Len(myString) - 2)
Next i

End Sub
```

By changing the parameter from **- 1** to **- 2**, the resulting length passed to the Right function is now two characters shorter than the original. This means the Right function retrieves the substring starting from the third character onward. If we run this revised macro on the same dataset, the output confirms the removal of the first two characters (the number and the subsequent space, assuming one existed, or the number and the first letter):

	A	B	C	D	E	F
1	**Team**					
2	Mavericks	vericks				
3	Pacers	cers				
4	Bucks	cks				
5	Wizards	zards				
6	Celtics	ltics				
7	Kings	ngs				
8	Nets	ts				
9	Spurs	urs				
10	Rockets	ckets				
11	Heat	at				
12						
13						
14						
15						
16						
17						
18						

As observed in the visual result, Column B displays each of the strings in column A with the first two characters removed, showcasing the scalability of this method for comprehensive data preparation.

Alternative Approach: Leveraging the Mid Function

While the combination of Right and Len is often the cleanest technique, it is important to recognize

that the Mid function can also achieve the same result. The Mid function is designed to return a specified number of characters from a string, starting at a designated position. To remove the first character, we instruct Mid to start extraction at the second character.

The syntax for using Mid to remove the first character is: **Mid(SourceString, 2)**. When the optional third parameter (length) is omitted, Mid returns all characters from the specified starting position (2) to the end of the string. This makes it an incredibly concise and elegant solution for this specific problem.

In the context of our previous macro, the key line would be modified to: **Range("B" & i) = Mid(myString, 2)**. Although both Right/Len and Mid achieve the same outcome, many developers find the explicit **Mid(..., 2)** structure slightly more readable and faster to implement when only the leading character needs to be stripped.

Best Practices for VBA String Handling

When implementing string manipulation routines in VBA, adhering to best practices ensures your code is robust, performant, and easily maintainable:

Error Handling: Always account for potential errors. For instance, if the input string is empty or contains fewer characters than you are attempting to remove (e.g., trying to remove 2 characters from a 1-character string), the code might throw an error or return unexpected results. Using conditional logic (like **If Len(myString) > N Then ...**) can prevent runtime failures.
Variable Declaration: Utilize **Option Explicit** at the top of every module and declare all variables explicitly (e.g., **Dim myString As String**). This practice reduces the chance of typographical errors leading to bugs, particularly when dealing with long strings.
Performance Optimization: When processing large datasets (thousands of rows), reading all data into an array first, processing the strings within the array, and then writing the final result back to the sheet is far faster than accessing the **Range** object within every loop iteration.

The following tutorials explain how to perform other common tasks using VBA: