How to Sort by Length in Google Sheets: A Step-by-Step Guide

Authored by

stats writer

January 14, 2026

In Google Sheets, mastering data manipulation often requires creative use of built-in functions. One common requirement is to sort data not based on alphabetical or numerical value, but based on the sheer length of the text strings contained within the cells. This capability is surprisingly powerful for tasks ranging from database cleanup to linguistic analysis. While Sheets does not offer a direct 'Sort by Length' button, the process is straightforward when utilizing the powerful combination of helper columns and sorting tools, or advanced formulas like **ARRAYFORMULA**. This detailed guide will walk you through the standard method using the SORT function and the LEN function, ensuring you can efficiently organize your textual data.

Often, you need to reorder the textual entries--the strings--within a column of your spreadsheet based entirely on the count of characters they contain. This specific type of sorting is invaluable when performing data normalization, analyzing database field sizes, or simply ensuring visual consistency in reports. Successfully achieving this requires combining the LEN function, which calculates string length, with Sheets' robust sorting capabilities.

Fortunately, executing this operation is remarkably easy when you leverage the LEN function to create a temporary metric. This metric--the calculated length--then serves as the primary key for the subsequent sorting procedure. The following step-by-step example demonstrates the beginner-friendly method involving a helper column, followed by a discussion of more advanced, formula-based techniques.

Understanding the Core Functions: LEN and SORT

Before diving into the implementation, it is essential to understand the roles of the two primary functions we will employ. The **LEN function** is a fundamental text function in Google Sheets, designed specifically to return the numerical count of characters in a specified string. This includes letters, numbers, spaces, and punctuation. By applying `LEN` to a range of cells, we generate an array of lengths corresponding to the original data.

The **SORT function**, on the other hand, is designed to sort an entire range of data based on the values in one or more specified columns. It takes three primary arguments: the range to be sorted, the index of the column that contains the sort keys, and a boolean value (`TRUE` for ascending order, `FALSE` for descending order) indicating the sort direction. When we use the character length calculated by `LEN` as our sort key, the **SORT function** meticulously reorders the original data based on these numerical length values.

In essence, sorting by length transforms the textual sorting challenge into a straightforward numerical sorting problem. Whether you use a helper column or embed the `LEN` calculation directly within the `SORT` formula, the principle remains the same: quantify the length first, then sort the original data based on that quantification. This methodology ensures accurate and rapid reordering of large datasets, facilitating better data management and analysis.

Step 1: Setting Up the Dataset

To illustrate this technique, we begin by populating our spreadsheet with the source data. For this example, we will use a list of various professional basketball teams. This setup simulates a real-world scenario where textual data needs organization based on its character size.

We will place these names starting in cell **A2**. It is generally considered best practice to leave the first row for clear column headers, which aids in data management and readability, especially when performing complex sorting operations later. Ensure your data is clean; for instance, avoid unintended leading or trailing spaces, as the **LEN function** counts those spaces as characters, which could inadvertently skew your sorting results and produce inaccurate organization.

The visual representation below shows our starting data, ready for the length calculation phase. This initial step is critical as the accuracy of the subsequent steps relies entirely on the integrity and organization of the input data range.

	A	B	C
1	**Team**		
2	Mavs		
3	Warriors		
4	Hawks		
5	Kings		
6	Lakers		
7	Nets		
8	Magic		
9	Timberwolves		
10	Clippers		
11	Pacers		
12			
13			
14			
15			
16			

Step 2: Calculating the Length of Each String (Helper Column Method)

The next critical step involves quantifying the character length of every string in Column A. We achieve this by introducing a temporary column, commonly referred to as a **helper column**, which

will store the numerical output of the LEN function. This method is highly recommended for beginners as it provides visual verification of the calculated lengths before the final sorting procedure.

In cell **B1**, label this new column "Length" for immediate clarity regarding its purpose. Then, starting in cell **B2**, we input the simple formula designed to reference the corresponding cell in the data column. Since our first team name is in **A2**, the formula calculates its precise character length:

=LEN(A2)

After entering the formula in **B2**, we must apply it efficiently to the entire dataset. Instead of manually retyping the formula for each row, we utilize the fill handle functionality: click on the bottom-right corner of cell **B2** and drag the formula down to the last row containing data (in our example, down to **B11**). This swift action automatically adjusts the cell reference (e.g., to **A3**, **A4**, and so on) for each subsequent row, quickly calculating the character count for every team name.

B2 =LEN(A2)

	A	B	C
1	**Team**	**Length**	
2	Mavs	4	
3	Warriors	8	
4	Hawks	5	
5	Kings	5	
6	Lakers	6	
7	Nets	4	
8	Magic	5	
9	Timberwolves	12	
10	Clippers	8	
11	Pacers	6	
12			
13			
14			
15			

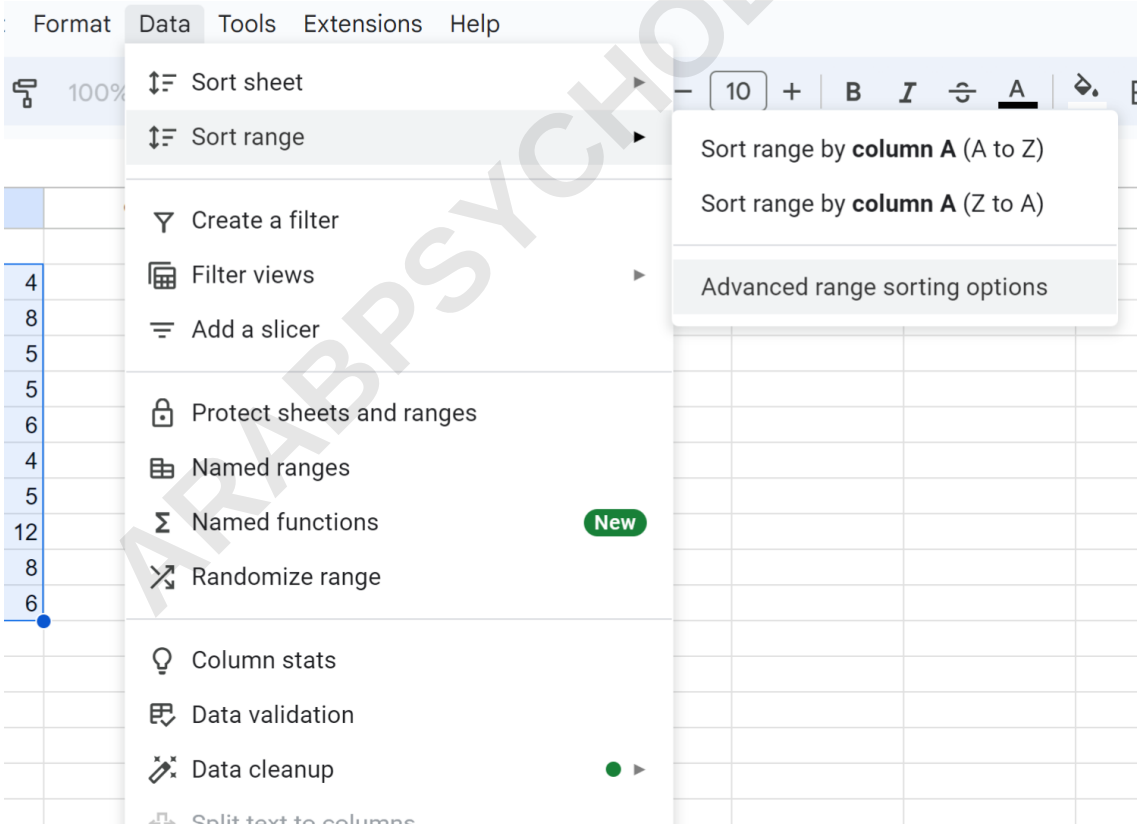
Upon completion, the newly populated **Length** column (Column B) provides the precise numerical length--the character count--for each respective string in the **Team** column (Column A). It is these numerical values that we will use as the primary key for our final sorting operation, demonstrating how the LEN function effectively transforms text data into numerical data suitable for sorting.

Step 3: Executing the Sort Operation via Data Menu

With the length calculations complete in the helper column, we are now ready to perform the actual physical sorting of the rows. This process involves using the powerful built-in sorting tools within Google Sheets to rearrange the entire dataset based on the numerical values generated in our 'Length' column.

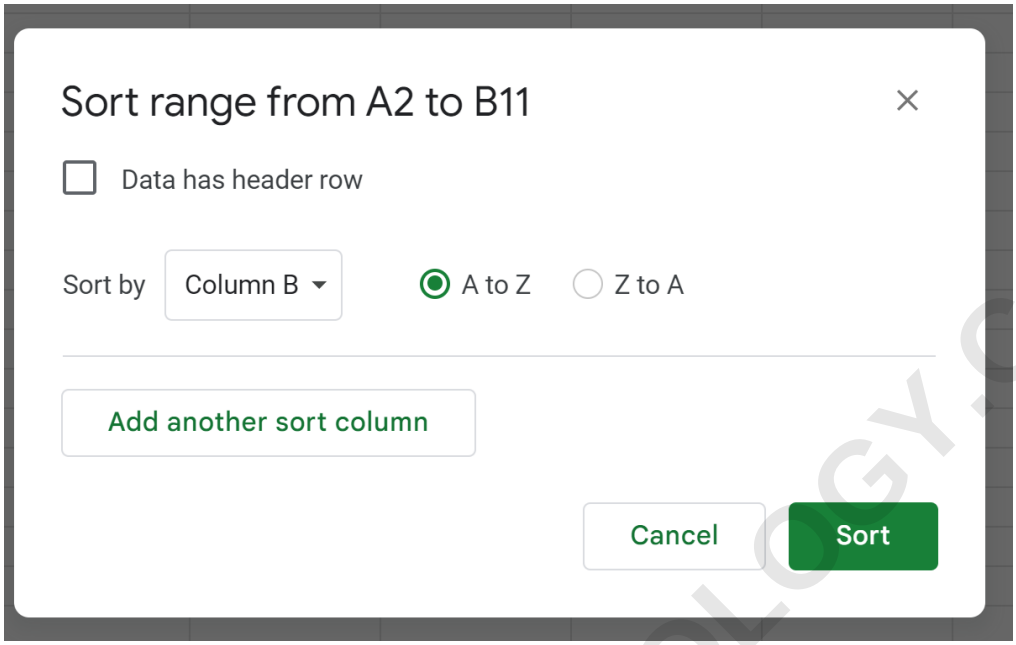
The first step is to accurately select the range that needs sorting. It is absolutely crucial to highlight the entire dataset, encompassing both the original strings and their corresponding calculated lengths. In our example, we meticulously highlight the cell range **A2:B11**. A common error is failing to include both columns; this error would result in the team names becoming detached from their correct lengths, thereby corrupting the dataset integrity and rendering the sort meaningless.

Once the full range is selected, navigate to the main menu ribbon: click the **Data** tab, select **Sort range**, and then choose the **Advanced range sorting options**. This specific sequence ensures that we have full, granular control over which column dictates the sorting order, preventing accidental sorting based on Column A's alphabetical value instead of Column B's numerical value.



In the resulting configuration window, you must specify the correct numerical sorting key. We instruct Sheets for the data to be sorted by the values in Column B. Select 'Column B' under the

Sort by dropdown menu. To sort from the shortest string to the longest string, choose the Ascending order option, typically labeled "A to Z" (which corresponds to sorting numbers from smallest to largest). Finally, confirm the action by clicking the **Sort** button.



The spreadsheet will instantly reorder all rows based on the calculated length metric. The team names in Column A are now organized based on their corresponding character counts in Column B, resulting in the desired output where the shortest names appear sequentially before the longest. The subsequent image confirms the successful completion of the length-based sorting procedure, moving precisely from the smallest numerical length to the largest.

	A	B	C	D
1	**Team**	**Length**		
2	Mavs	4		
3	Nets	4		
4	Hawks	5		
5	Kings	5		
6	Magic	5		
7	Lakers	6		
8	Pacers	6		
9	Warriors	8		
10	Clippers	8		
11	Timberwolves	12		
12				
13				
14				
15				
16				

Sorting Options and Post-Processing Notes

The standard sort method, as demonstrated above, inherently places the shortest strings first (ascending order). However, flexibility in data organization is paramount for various analytical needs. If your specific goal is to immediately identify the longest strings, you must reverse the sorting direction during the configuration in Step 3. Instead of selecting "A to Z" (Ascending), you must choose **Sort by Z to A** (Descending order).

This Descending sort ensures that the strings with the greatest character counts rise to the top of the dataset. This variation is particularly useful in specialized fields such as search engine optimization (SEO), where identifying maximum-length page titles or lengthy meta descriptions is necessary, or in database administration where constraint checks on input length are critical to prevent data overflow errors.

Furthermore, it is important to emphasize that the helper column created in Step 2 is merely a computational tool used to facilitate the initial sorting process. Once the data has been correctly sorted based on string length, the **Length** column often becomes redundant relative to the final organized data presentation. Therefore, you should feel free to delete Column B entirely to maintain a clean and professional presentation of your final dataset, unless you explicitly require the length data for subsequent complex statistical analysis.

Alternative Method: Dynamic Sorting with the SORT Function

For users seeking a more concise and entirely dynamic solution that eliminates the necessity of a visible, persistent helper column, the SORT function can be utilized directly within a single formula. This method is highly efficient because it performs the length calculation and the sorting operation simultaneously, outputting the entire sorted array immediately to a specified location.

The general syntax for this dynamic approach involves embedding the LEN function within the sort index argument of the **SORT function**. Since `SORT` requires an array or column of numerical values to sort by, we generate that array of lengths on the fly, making it act as the sorting key. If your original data is contained within the range `A2:A10`, the complete formula for sorting the data by length in ascending order is:

=SORT(A2:A10, LEN(A2:A10), TRUE)

In this powerful formula, `A2:A10` serves two purposes: it defines the data range being returned as the sorted result, and it defines the range used as the input for calculating the sorting index. The expression `LEN(A2:A10)` dynamically generates the necessary numerical array of lengths. Finally, `TRUE` specifies the sort direction (shortest to longest). This dynamic method automatically spills the results into the adjacent cells, making it the preferred method for maintaining the original, unsorted data structure while creating a constantly updated sorted view elsewhere in the sheet.

Practical Applications of Length-Based Sorting

Sorting data by string length extends far beyond simple alphabetical organization and provides significant utility across numerous data management and analysis fields. Understanding these applications helps leverage this technique effectively:

Data Quality and Validation: Length sorting is invaluable for identifying anomalies in data entry. For example, if a column is standardized to contain unique identification codes that must be exactly 12 characters long, sorting by length instantly isolates codes that are 6 characters (too short) or 15 characters (too long), flagging them for immediate audit and correction, thereby enforcing data integrity standards.

Linguistic and Text Analysis: When engaged in text mining or preliminary natural language processing (NLP), sorting sentences, phrases, or individual words by length offers crucial insights. It helps analysts study writing patterns, differentiating common short, simple phrases from complex, lengthy descriptive constructs, allowing for detailed analysis of textual complexity and lexical variation within a corpus.

Report Design and User Interface Aesthetics: In the creation of professional visualizations or

reports, the order in which labels, categories, or names are displayed significantly impacts readability. Sorting labels from shortest to longest often improves the visual flow and prevents misalignment in charts, legends, and tables, contributing to a cleaner, more intuitive user experience.

Database Migration Preparation: Preparing textual data for transfer to a relational database often involves checking field data against strict maximum character constraints. By sorting the source data from the longest strings to the shortest, users can quickly and proactively identify any strings that might exceed the predetermined character limits of the target database columns, preventing critical truncation errors or import failures during the migration process.

This feature, although straightforward in its execution using the LEN function, provides a powerful and indispensable organizational tool for anyone working extensively with complex or high-volume textual data within Google Sheets.