How to Google Sheets: Concatenate Cells with Line Break

Authored by

stats writer

November 19, 2025

Effective data management often requires combining disparate pieces of information into a single, comprehensive field. When working in a powerful spreadsheet environment like **Google Sheets**, the need to **concatenate** data from multiple cells while ensuring readability through proper formatting is paramount. This technique is especially useful for merging addresses, full names, or multi-line descriptions.

This comprehensive guide details the precise methods available within Google Sheets to merge cell content and insert a distinct **line break** between the concatenated values. Achieving this structure allows for neat, organized, and professional presentation of your data.

We will explore two primary, yet distinct, functional approaches: using the classic **CONCATENATE** function and utilizing the more flexible **TEXTJOIN** function. Mastering these methods will significantly enhance your data manipulation capabilities within the platform.

The Importance of the Line Feed Character: CHAR(10)

Before diving into the specific functions, it is essential to understand how Google Sheets interprets a line break within a cell's value. Unlike pressing 'Alt + Enter' within the cell itself, standard concatenation functions require a specific character code to signal a new line.

In the context of spreadsheet applications, including Google Sheets, the non-printable character known as the Line Feed is used for this purpose. This character is universally represented by the numeric code 10 in the **CHAR** function. Therefore, the expression **CHAR(10)** becomes the crucial delimiter we use to introduce a line break between concatenated text strings.

It is important to note that while **CHAR(10)** inserts the line break, the cell containing the output formula must have 'Text Wrapping' enabled to display the break correctly. If wrapping is not enabled, the content will simply appear concatenated on a single, extended line, even though the character code is present.

Overview of Concatenation Methods

Google Sheets offers two powerful ways to handle concatenation, both of which can incorporate the **CHAR(10)** delimiter. The choice between them often depends on the complexity of the data range and whether you need to handle potential empty cells automatically.

The first method employs the established **CONCATENATE()** function, which explicitly links each text string and delimiter individually. The second method uses the more modern and efficient **TEXTJOIN()** function, which is designed to handle delimiters across ranges of data effortlessly.

Below are the basic structures for merging the contents of two cells, **A1** and **A2**, into a single cell using a line break:

Method 1: Using CONCATENATE()

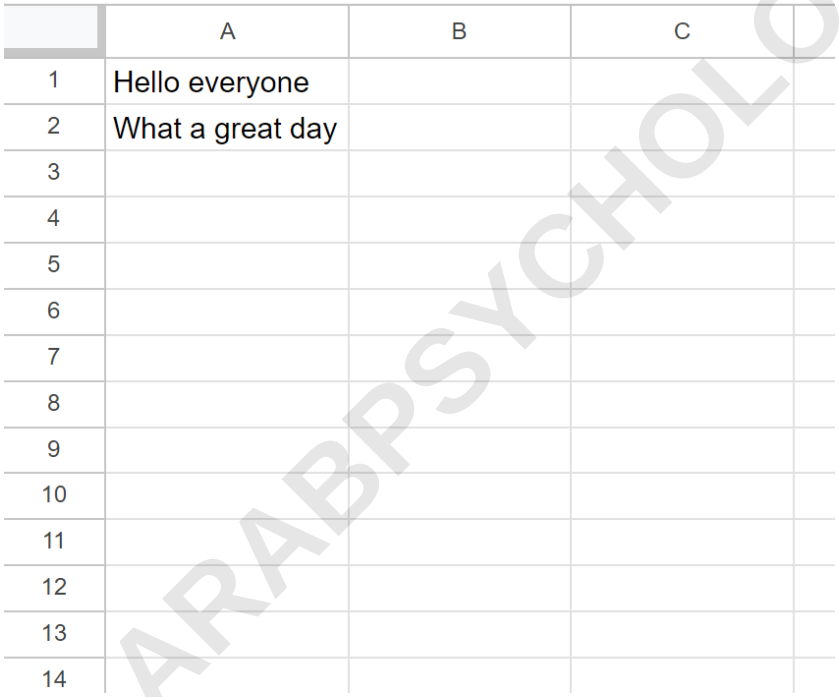
=CONCATENATE(A1, CHAR(10), A2)

Method 2: Using TEXTJOIN()

=TEXTJOIN(CHAR(10), TRUE, A1:A2)

Both formulas achieve the desired outcome: merging the values found in cells **A1** and **A2**, separated by a definitive line break.

We will now use a consistent dataset to demonstrate how each approach operates in a live environment. Consider the following data setup, where the content of **A1** is 'Name:' and the content of **A2** is 'John Doe':



	A	B	C
1	Hello everyone		
2	What a great day		
3			
4			
5			
6			
7			
8			
9			
10			
11			
12			
13			
14			

Method 1: Utilizing the CONCATENATE Function for Line Breaks

The **CONCATENATE()** function is one of the oldest and most direct methods for combining multiple text strings or cell references in Google Sheets. This function accepts a series of arguments, merging them sequentially into a single output string. To insert a line break, we must explicitly include **CHAR(10)** as an individual argument between the cells we wish to separate.

The structure requires careful placement of the arguments. You must specify the first cell, followed

by the delimiter (**CHAR(10)**), and then the subsequent cell. If you were merging five cells, you would need four instances of **CHAR(10)** interspersed throughout the **formula**. This explicit approach ensures absolute control over the placement of every character, including the line break.

While effective, the manual nature of listing every cell and every delimiter makes **CONCATENATE()** less efficient for handling large, continuous ranges of data. It is best reserved for combining a small, fixed number of **cells** or mixing static text strings with cell references.

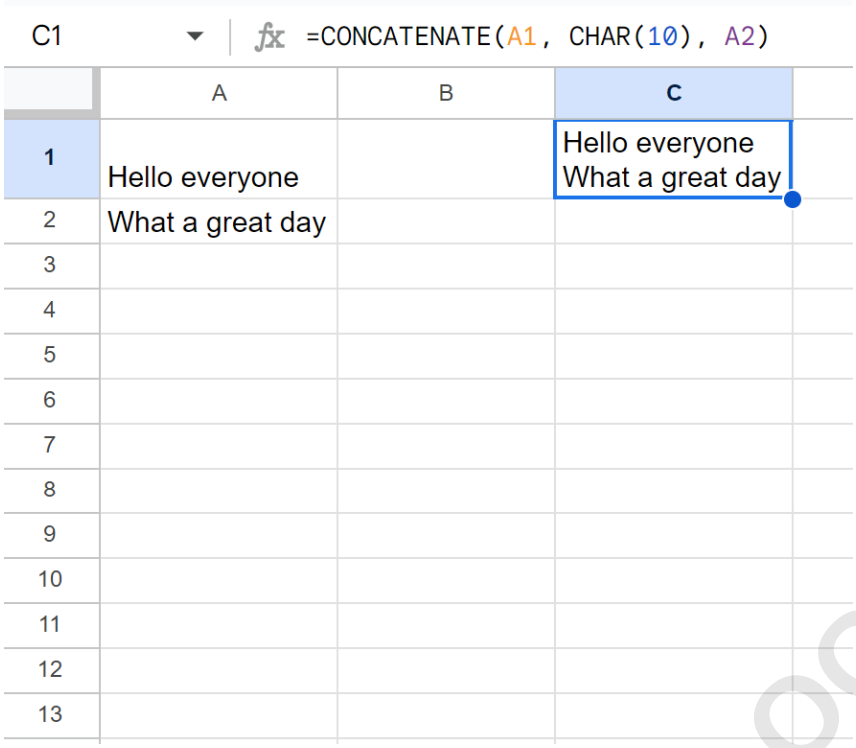
Practical Application of CONCATENATE()

To see this method in action, we will input the formula into cell **C1**, referencing the values established in **A1** ('Name:') and **A2** ('John Doe'). The goal is to display 'Name:' on the first line and 'John Doe' on the second line within cell **C1**. We construct the formula as follows:

=CONCATENATE(A1, CHAR(10), A2)

Upon execution, the contents of the two source cells are successfully joined, with **CHAR(10)** serving as the invisible bridge that forces the second string onto a new line. The resulting output demonstrates perfect alignment and formatting, assuming 'Text Wrapping' is active in cell **C1**.

The resulting visual representation confirms that the strings from **A1** and **A2** have been correctly concatenated and separated by a line break, illustrating the power and specificity of the **CONCATENATE()** function combined with **CHAR(10)**:



Method 2: Leveraging the TEXTJOIN Function

The **TEXTJOIN()** function offers a significant advantage over **CONCATENATE()**, particularly when dealing with ranges or conditional concatenation. **TEXTJOIN()** requires three main arguments: the delimiter, a boolean value determining whether to ignore empty cells, and the range or sequence of text strings to join.

For our purpose--inserting a line break--the delimiter argument is specified as **CHAR(10)**. The second argument, the boolean value, is critical for clean data. Setting it to **TRUE** ensures that if any cell within the specified range is empty, that cell is skipped entirely, preventing unnecessary blank lines in your output. Setting it to **FALSE** means empty cells will still trigger the delimiter, potentially resulting in blank lines.

Due to its ability to handle data ranges seamlessly and its built-in mechanism for ignoring blanks, **TEXTJOIN()** is the preferred method for complex data merging tasks, especially when dealing with lists that might have sporadic empty entries.

Practical Application of TEXTJOIN()

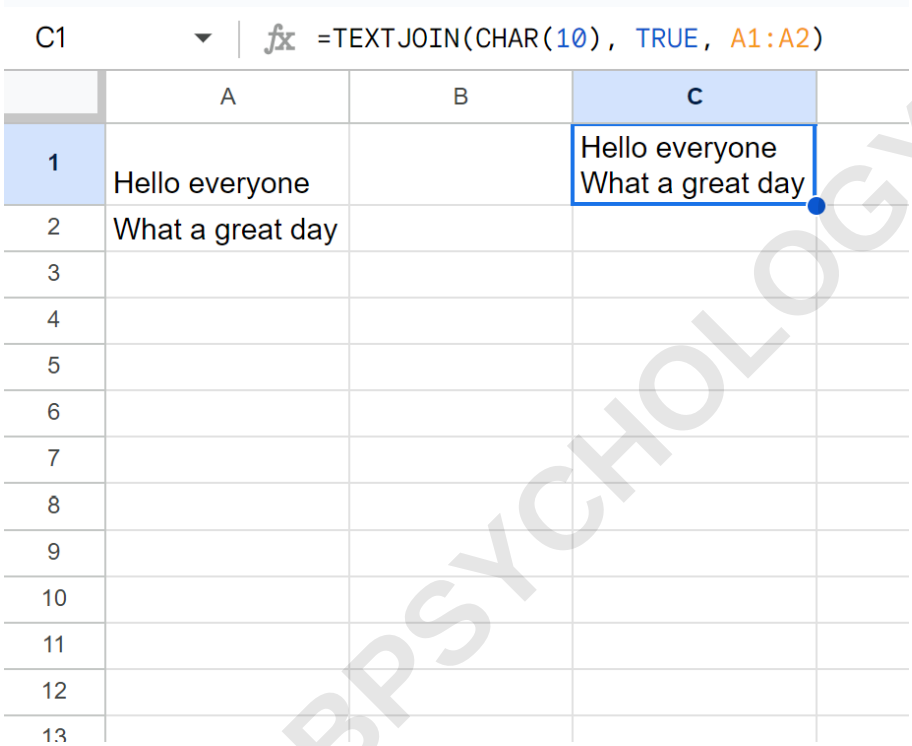
Using the same example data (**A1**: 'Name:', **A2**: 'John Doe'), we apply the **TEXTJOIN()** function in cell **C1**. Since we are dealing with only two adjacent cells, we can define the source data using a range reference (**A1:A2**). We use **CHAR(10)** as our separator and set the ignore_empty argument

to **TRUE**.

The resulting formula structure is considerably cleaner than **CONCATENATE()** for ranges:

=TEXTJOIN(CHAR(10), TRUE, A1:A2)

As illustrated in the corresponding screenshot, the function successfully merges the strings from the defined range, inserting the **CHAR(10)** line break between the two elements. The output is identical to the result achieved using **CONCATENATE()** in this simple, two-cell scenario.



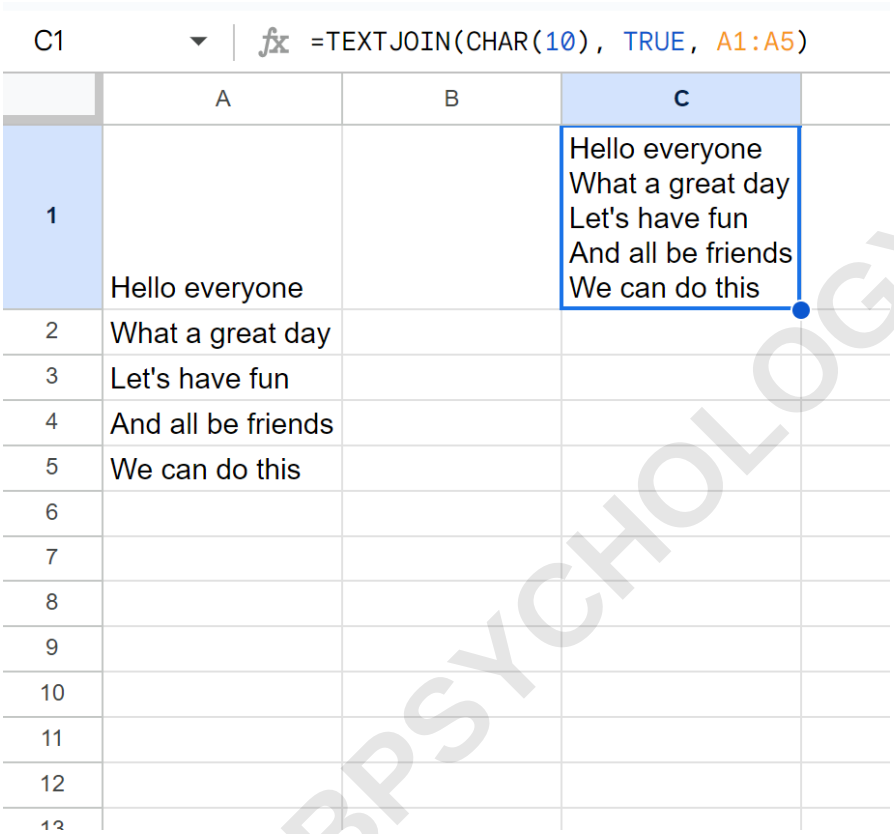
Advanced Range Concatenation Using TEXTJOIN()

The true power of **TEXTJOIN()** emerges when combining extensive data ranges. Unlike **CONCATENATE()**, which would require listing every cell reference individually, **TEXTJOIN()** accepts a single range reference, applying the specified delimiter efficiently across all included cells.

Consider a situation where you need to combine five distinct data points located in cells **A1** through **A5** into one output cell, with each entry appearing on a new line. Using **TEXTJOIN()** simplifies this task immensely by defining the range **A1:A5** as the final argument. The function automatically inserts **CHAR(10)** between every non-empty cell within that defined range.

Furthermore, setting the 'ignore_empty' argument to **TRUE** (as seen in the formula below) is crucial for maintaining clean output integrity. If, for instance, cell **A3** were blank, **TEXTJOIN()** would seamlessly skip it, ensuring that the content of **A2** is followed immediately by a line break and then the content of **A4**, preventing redundant blank lines.

The following example demonstrates how **TEXTJOIN()** concatenates the entire range **A1:A5** using line breaks:



The screenshot clearly shows that the contents of all five cells have been successfully consolidated into a single cell, with each original entry starting on a new line, confirming the efficiency of **TEXTJOIN()** for bulk data merging.

Choosing the Right Function: CONCATENATE vs. TEXTJOIN

When implementing line-break concatenation, the choice between **CONCATENATE()** and **TEXTJOIN()** depends primarily on the scope and variability of your source data. Both functions are highly effective when working with a fixed number of two or three adjacent cells, offering similar readability and performance.

However, for projects involving dynamic data ranges, potentially large numbers of cells, or data

sets prone to containing empty fields, **TEXTJOIN()** is the superior choice. Its ability to accept range inputs and its native handling of null values dramatically reduces the complexity and length of the required formula, making it easier to audit and maintain.

In contrast, **CONCATENATE()** remains valuable when you need meticulous, non-sequential merging--for example, combining A1, then a static phrase, then D5, then a line break, then B2. The explicit nature of its arguments grants precise control over every element in the output string.

Regardless of the function chosen, the core element for successful multi-line formatting remains the same: the careful integration of the **CHAR(10)** character, which is the universal signal for a **line break** within the Google Sheets environment.