Excel: Split Text and Get Last Item

Authored by

stats writer

November 17, 2025

1. Introduction: Efficient Text Parsing in Modern Excel

The ability to efficiently parse and manipulate textual data is fundamental to advanced data analysis in Excel. Prior to the introduction of dynamic array functions, extracting specific elements, such as the last item from a delimited string, often required complex combinations of functions like **MID**, **FIND**, **LEN**, and **SEARCH**. This traditional process was cumbersome, difficult to scale, and prone to errors due to nested logic. Fortunately, modern Excel provides powerful, streamlined tools that simplify this process dramatically, offering both clarity and robustness.

Chief among these tools is the **TEXTSPLIT function**, which allows users to easily fragment a single text string based on a specified character or sequence--known as a delimiter. The output of **TEXTSPLIT** is a dynamic array containing all the resulting pieces of text. However, to achieve our specific goal--isolating only the last piece of the fragmented data--we must combine **TEXTSPLIT** with another essential dynamic array utility: the **CHOOSECOLS function**.

This guide will demonstrate the precise and elegant formula used to split text based on a delimiter and subsequently retrieve the final resulting item. This technique is indispensable for common data cleaning operations, particularly when dealing with concatenated data such as full names, composite codes, or hierarchical addresses where only the trailing component holds the required analytical value. We will explore how these two functions seamlessly integrate to deliver a simple yet powerful solution.

2. Mastering the Core Formula Syntax

To achieve the efficient extraction of the last text segment, we utilize a combination of **TEXTSPLIT** to break the string apart and **CHOOSECOLS** to select the desired output column. The combined formula is exceptionally concise and leverages the inherent capabilities of dynamic arrays, resulting in high efficiency compared to traditional methods.

The specific syntax used to split the text in cell **A2** using a standard space character as the delimiter and retrieve the last element is as follows:

=CHOOSECOLS(TEXTSPLIT(A2, " "), -1)

In this powerful one-line command, the inner function, **TEXTSPLIT(A2, " ")**, executes first, performing the initial segmentation and creating a temporary horizontal array in memory. The argument `" "` specifies that the space character is the separator. If the source string in cell **A2** is "Chad Mike Douglas", **TEXTSPLIT** generates a three-column array: `{"Chad", "Mike", "Douglas"}`. This temporary array is then immediately passed as the primary argument to the outer **CHOOSECOLS** function.

The truly innovative component that isolates the last item is the second argument of **CHOOSECOLS**: `-1`. The use of a negative index in the **CHOOSECOLS function** instructs Excel to count columns backward from the end of the array rather than forward from the beginning. A value of `-1` guarantees the selection of the final column, regardless of the total number of segments produced by the split. Consequently, for the input "Chad Mike Douglas", the formula extracts and returns only the desired value: **Douglas**.

3. Deep Dive into the TEXTSPLIT Function

The **TEXTSPLIT function** is a cornerstone of modern Excel text manipulation, built specifically to overcome the constraints of older string processing formulas. Its primary role is to dissect a text string based on user-defined criteria, generating a multi-column or multi-row array of resulting substrings. This output is inherently dynamic, meaning the size of the array adapts automatically based on the input string's content.

The full syntax for **TEXTSPLIT** is highly versatile: `TEXTSPLIT(text, col_delimiter, , , , )`. While our specific goal requires only the first two arguments, a comprehensive understanding of the optional parameters is valuable for handling diverse data structures.

Text (Required): This is the cell reference or text string containing the data to be split (e.g., **A2**).
Col_delimiter (Required): This specifies the character or sequence of characters that signals where the text should be fragmented into new columns (e.g., `" "` for space, `", "` for comma and space).
Row_delimiter (Optional): This allows for splitting the text into new rows, useful if the input contains line breaks or other row separation characters within a single cell.
Ignore_empty (Optional): This Boolean argument (**TRUE** or **FALSE**, or 1 or 0) controls how consecutive delimiters are handled. Setting this to **TRUE** (1) is recommended when dealing with potentially messy source data, as it prevents the generation of empty strings in the resulting array.

In the context of extracting the last element, the **TEXTSPLIT** function is responsible for the crucial initial step: quantifying and separating all components. If the string contains *N* separate elements, **TEXTSPLIT** flawlessly produces an array with *N* columns, providing a complete, intermediate result that is perfectly structured for the subsequent selection process.

4. Leveraging CHOOSECOLS for Specific Array Extraction

The **CHOOSECOLS function** serves as the selector in our combined formula. While **TEXTSPLIT** generates the full array of split elements, **CHOOSECOLS** is utilized to filter this array, returning only the specific column required. This function is a dedicated tool for array manipulation, offering a clean way to manage the structure and dimensionality of dynamic array results.

The primary benefit of using **CHOOSECOLS** in this context is its native support for negative indexing, which eliminates the need for calculating the total count of elements using a separate function like **COLUMNS** or **COUNTA**, which was necessary in older array-based formulas.

Negative Indexing Explained: When you specify a column number as a negative integer (e.g., -1, -2, -3), Excel interprets this as counting backward from the last column in the input array.
The Power of -1: By using `-1` as the column number argument, we instruct the function to retrieve the final column. This methodology is supremely robust because it handles strings of any length--whether the input string has two words or ten words, the final word will always occupy the `-1` column position.

Contrast this approach with retrieving the first item, which would simply use `1` as the column index, or the second-to-last item, which would use `-2`. The consistency provided by negative indexing ensures that the formula remains invariant to the internal data structure generated by the preceding **TEXTSPLIT** operation, making it highly reliable for batch processing large datasets with variable string lengths.

5. Practical Application: Step-by-Step Example

To illustrate the practical utility of this formula, let us consider a typical scenario involving data normalization. We have a column containing full names, and the objective is to isolate and extract only the last name for standardizing records or integrating with other data tables. Our goal is to calculate the result in column B based on the input data in column A.

Suppose we have the following column of names in Excel:

	A	B	C	D
1	**Name**			
2	Andy Bernard			
3	Chad Mike Douglas			
4	Eric Ferdinand			
5	Josh Mann			
6	Arnold Smith Simmons			
7	Jake Johnson			
8	Tyson Reed			
9	Brett Handzel			
10	Mike Scott			
11				
12				
13				
14				

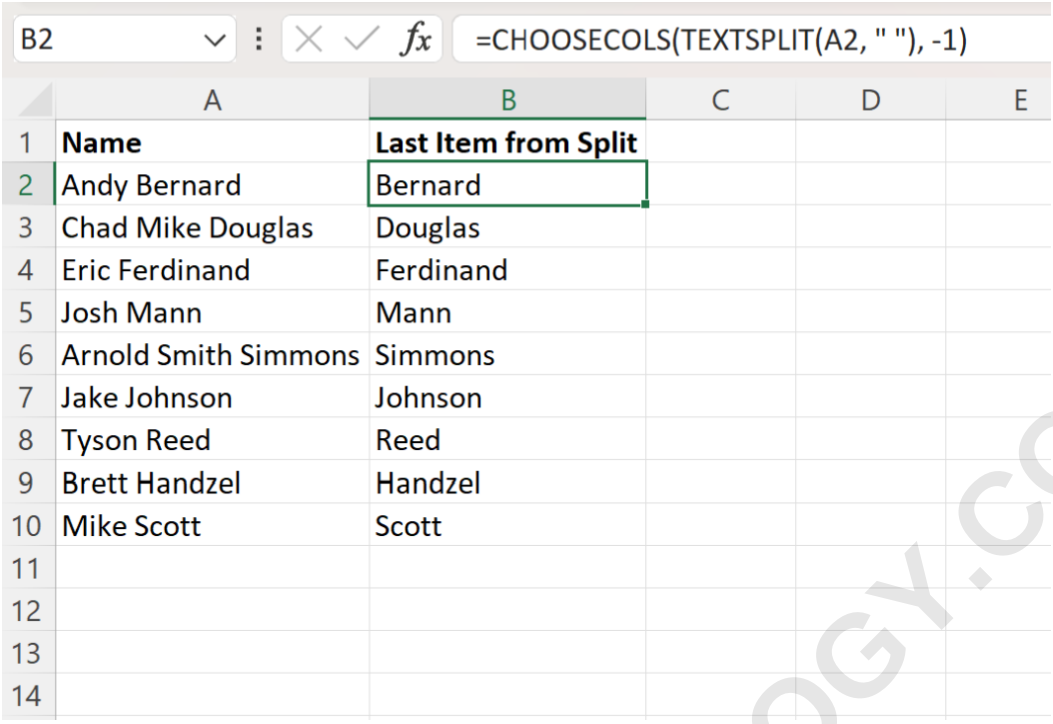
We seek to split these names based on the space delimiter and then retrieve only the final segment, corresponding to the last name. We must input our complete, nested formula into the first corresponding data cell, which is **B2**.

We can type the following formula into cell **B2** to initiate the text parsing and extraction process:

=CHOOSECOLS(TEXTSPLIT(A2, " "), -1)

After entering the formula into **B2**, the result for the first row will appear instantly. Due to the dynamic nature of these functions, if you are using a version of Excel that fully supports dynamic arrays, the result might automatically "spill" down to adjacent cells in column B, calculating the result for all corresponding rows in column A simultaneously. If automatic spilling does not occur, the traditional method of applying the formula to the remaining cells must be used.

We can then click and drag this formula down to apply it to each remaining cell in column B:



	A	B	C	D	E
1	**Name**	**Last Item from Split**			
2	Andy Bernard	Bernard			
3	Chad Mike Douglas	Douglas			
4	Eric Ferdinand	Ferdinand			
5	Josh Mann	Mann			
6	Arnold Smith Simmons	Simmons			
7	Jake Johnson	Johnson			
8	Tyson Reed	Reed			
9	Brett Handzel	Handzel			
10	Mike Scott	Scott			
11					
12					
13					
14					

The resulting output confirms that the formula successfully splits the text in column A based on the occurrence of the space character and accurately returns the last item from the split array for every row. In this highly common scenario, the formula effectively isolates the last name, providing clean, structured data ready for further manipulation or reporting.

6. Analyzing the Formula's Internal Logic

A detailed examination of how the formula `=CHOOSECOLS(TEXTSPLIT(A2, " "), -1)` processes data internally is vital for mastering its application and adapting it for variations in input data. The execution begins with the innermost function and works its way outward.

=CHOOSECOLS(TEXTSPLIT(A2, " "), -1)

The process initiates with **TEXTSPLIT(A2, " ")**. For a cell containing the full name "Andy Bernard," the function recognizes the space delimiter and partitions the string. This immediate action creates a temporary array in Excel's memory structured as `{"Andy", "Bernard"}`. This array is the complete, raw result of the split and is seamlessly handed off to the containing function.

Next, the **CHOOSECOLS function** takes this temporary array `{"Andy", "Bernard"}` as its input. The second parameter, `-1`, dictates the column selection strategy. Because it is a negative index, **CHOOSECOLS** selects the column by counting backward. Since "Bernard" occupies the last column in the array, the function successfully returns **Bernard** as the final result of the entire

expression.

This nested structure exemplifies the efficiency of modern Excel array processing, avoiding the resource-intensive string length calculations and character searching that were previously mandatory. By relying on two specialized functions, the resulting formula is not only shorter but significantly more transparent in its intended operation: first split the text, then choose the last column.

7. Considerations for Different Delimiters and Edge Cases

The utility of this combined formula extends far beyond simple name splitting. Its core strength lies in its adaptability to different data structures simply by changing the delimiter argument within the **TEXTSPLIT** function. For instance, if you were parsing item codes delimited by a hyphen, the formula would be adjusted to `=CHOOSECOLS(TEXTSPLIT(A2, "-"), -1)`. Furthermore, **TEXTSPLIT** supports arrays of delimiters, allowing you to split based on multiple possible separators (e.g., commas or semicolons) simultaneously: `=CHOOSECOLS(TEXTSPLIT(A2, {",", ";"}), -1)`.

To ensure maximum reliability across varied datasets, it is imperative to address common formatting issues, or edge cases, primarily using **TEXTSPLIT's** optional arguments:

Handling Multiple Consecutive Delimiters: If your source data contains redundant spaces (e.g., "Item 1 Item 2"), **TEXTSPLIT** will, by default, generate an empty string in the array for the extra space. To clean this up and consolidate the results, utilize the optional `ignore_empty` argument by setting it to **TRUE** (or 1): `=CHOOSECOLS(TEXTSPLIT(A2, " ", , TRUE), -1)`. Note the necessary comma placeholder used to skip the optional `row_delimiter` argument.
Addressing Leading or Trailing Whitespace: If cell **A2** contains unnecessary spaces before the first word or after the last word, these extraneous characters can sometimes interfere with downstream processing. The best practice is to preprocess the input text using the **TRIM** function, which removes leading and trailing spaces and reduces internal spaces to a single character: `=CHOOSECOLS(TEXTSPLIT(TRIM(A2), " "), -1)`. The **TRIM** function ensures that the **TEXTSPLIT** operation receives perfectly clean data.

By integrating these robust features and handling optional arguments correctly, users can ensure that the combined **CHOOSECOLS** and **TEXTSPLIT** methodology remains the most reliable and efficient technique for consistently extracting the last element from any structured text string in modern Excel.

Note: You can find the complete documentation for the **CHOOSECOLS function** in Excel online documentation, which provides further detail on its negative indexing and array manipulation capabilities.