

# How to extract text between two characters in Excel?

Authored by  
**stats writer**

November 18, 2025

## RECOMMENDED CITATION

stats writer (2025). *How to extract text between two characters in Excel?*.

PSYCHOLOGICAL SCALES. Retrieved from <https://scales.arabpsychology.com/?p=96429>

Analyzing and manipulating text data is a cornerstone of advanced data processing within Excel. A frequent requirement is the ability to isolate specific substrings--that is, characters located precisely between two defined boundary markers, often referred to as delimiters. Historically, accomplishing this task required complex nested formulas utilizing functions like **MID**, **SEARCH**, and **FIND**. While effective, these legacy approaches were prone to error and difficult for the average user to maintain.

Fortunately, modern versions of Excel have introduced powerful, intuitive text manipulation tools that dramatically simplify this process. This comprehensive guide details how to leverage the specialized **TEXTBEFORE** and **TEXTAFTER** functions to cleanly and efficiently extract text segments based on specified delimiters, thereby streamlining your data workflow. We will walk through the exact methodology and demonstrate practical applications using various character types.

Mastering this technique allows for rapid parsing of standardized datasets, such as log files, formatted reports, or lists containing identifiers enclosed by specific strings or characters. By the end of this tutorial, you will possess the expertise required to implement this solution across diverse data environments, minimizing manual data cleaning efforts and maximizing accuracy.

## Introducing the Modern Solution: **TEXTBEFORE** and **TEXTAFTER**

The introduction of the TEXTBEFORE and TEXTAFTER functions revolutionized string handling in Excel. These functions are designed specifically to return the text segment either before or after a designated delimiter. When extracting content situated between two delimiters, we must logically combine both functions into a single, cohesive function. This combination creates a robust method for precise substring extraction without the cumbersome calculations previously necessary.

The core principle involves a two-step operation. First, the TEXTAFTER function is executed to eliminate all content up to and including the starting delimiter, isolating the desired segment plus the remaining string. Second, the result of this operation is fed directly into the TEXTBEFORE function. This subsequent step isolates the content before the ending delimiter, thereby achieving the exact text extraction required.

It is paramount to recognize that both the starting and ending delimiters used in the formula are defined by the user. These delimiters can be single characters, like a parenthesis or asterisk, or they can be entire strings of text. This flexibility makes the formula highly adaptable to nearly any structured dataset where boundaries are clearly defined.

## Understanding the Core Syntax and Logic

To successfully extract text situated between two specific characters or strings within a single cell,

you must nest the **TEXTAFTER** function inside the **TEXTBEFORE** function. This approach ensures that the output of the first operation serves as the input for the second, guaranteeing the removal of both boundary markers.

The standard syntax for this combined operation is as follows:

```
=TEXTBEFORE(TEXTAFTER(A2, "this"), "that")
```

In this specific, pedagogical example, the formula is designed to perform a sequential extraction. The **TEXTAFTER** component first analyzes the content of cell A2, returning everything that follows the string "this." Subsequently, the **TEXTBEFORE** component takes that result and returns everything that precedes the string "that." The net result of this powerful combination is the accurate extraction of all the text located exclusively between the starting character "this" and the ending character "that" within the specified source cell A2.

The following detailed examples illustrate common scenarios and provide the practical formulas necessary to implement this highly efficient text extraction method in your worksheets.

### Example 1: Extracting Text Between Specific Strings

One of the most frequent applications of this technique is extracting numerical data or identifiers that are surrounded by descriptive text strings. Consider a dataset where the distance covered is embedded between the words "ran" and "miles." To isolate only the numerical value, we treat these words as our defined delimiters.

We can formulate the extraction by typing the following combined function into the designated result cell, typically **B2**, if our source data resides in column A starting at **A2**. The goal here is to extract the text in cell **A2** that is positioned exactly between the string "ran" and the string "miles," thereby isolating the quantitative information.

```
=TEXTBEFORE(TEXTAFTER(A2, "ran"), "miles")
```

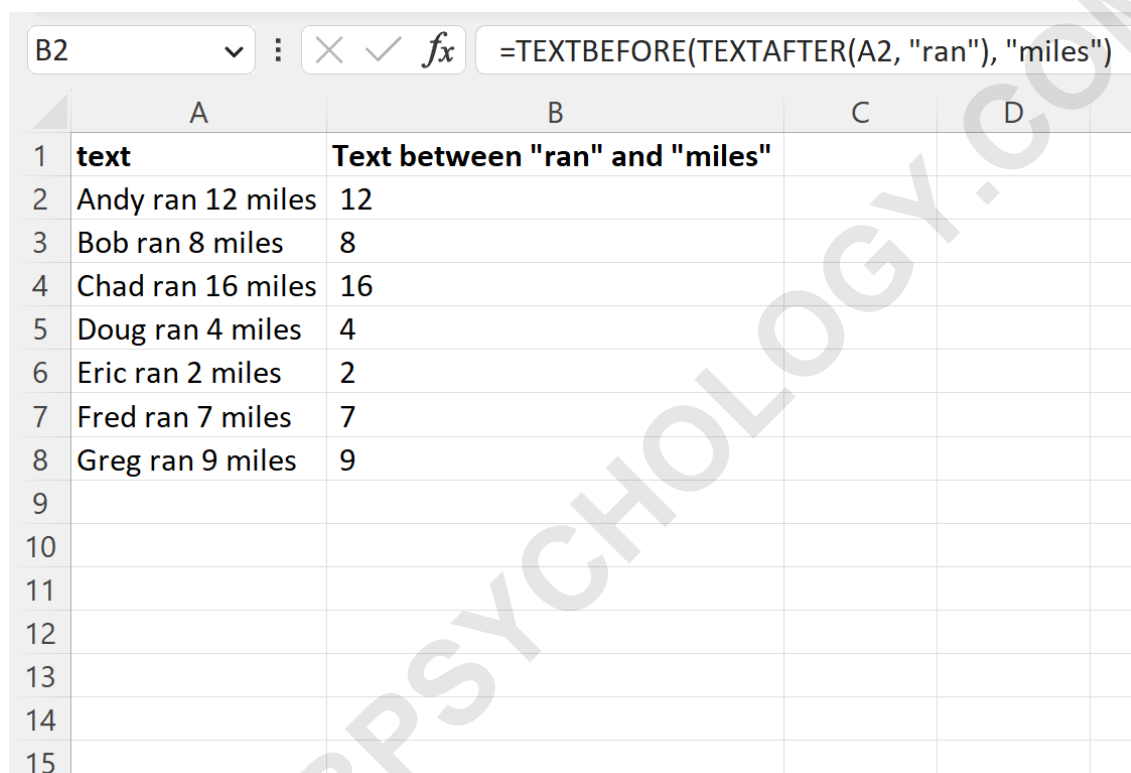
This formula first strips the string up to and including "ran" using **TEXTAFTER**, and then, from the remaining string, it retains everything before the string "miles" using **TEXTBEFORE**. This highly specific instruction ensures only the critical data point remains, ready for further mathematical analysis or reporting.

### Implementation Steps for String Extraction

Once the formula is correctly entered into the initial result cell (e.g., **B2**), the process becomes exceptionally rapid across the entire dataset. Unlike complex array formulas, this nested structure

allows for straightforward propagation to the subsequent rows. Users can easily click and drag the fill handle--the small square at the bottom-right corner of the selected cell--down the length of column B.

Executing this click-and-drag action ensures that the relative cell reference **A2** automatically adjusts to **A3**, **A4**, and so forth, applying the identical extraction logic to every corresponding row in the source column A. This efficiency is critical when handling large volumes of data, eliminating the need to manually adapt the formula for each entry.



	A	B	C	D
1	<b>text</b>	<b>Text between "ran" and "miles"</b>		
2	Andy ran 12 miles	12		
3	Bob ran 8 miles	8		
4	Chad ran 16 miles	16		
5	Doug ran 4 miles	4		
6	Eric ran 2 miles	2		
7	Fred ran 7 miles	7		
8	Greg ran 9 miles	9		
9				
10				
11				
12				
13				
14				
15				

Upon completion of the drag operation, column B will be populated exclusively with the extracted text segments. Specifically, column B now contains the text found between the strings "ran" and "miles" for each corresponding data entry present in column A, demonstrating a successful, automated extraction across the dataset.

## Example 2: Extracting Text Enclosed by Parentheses

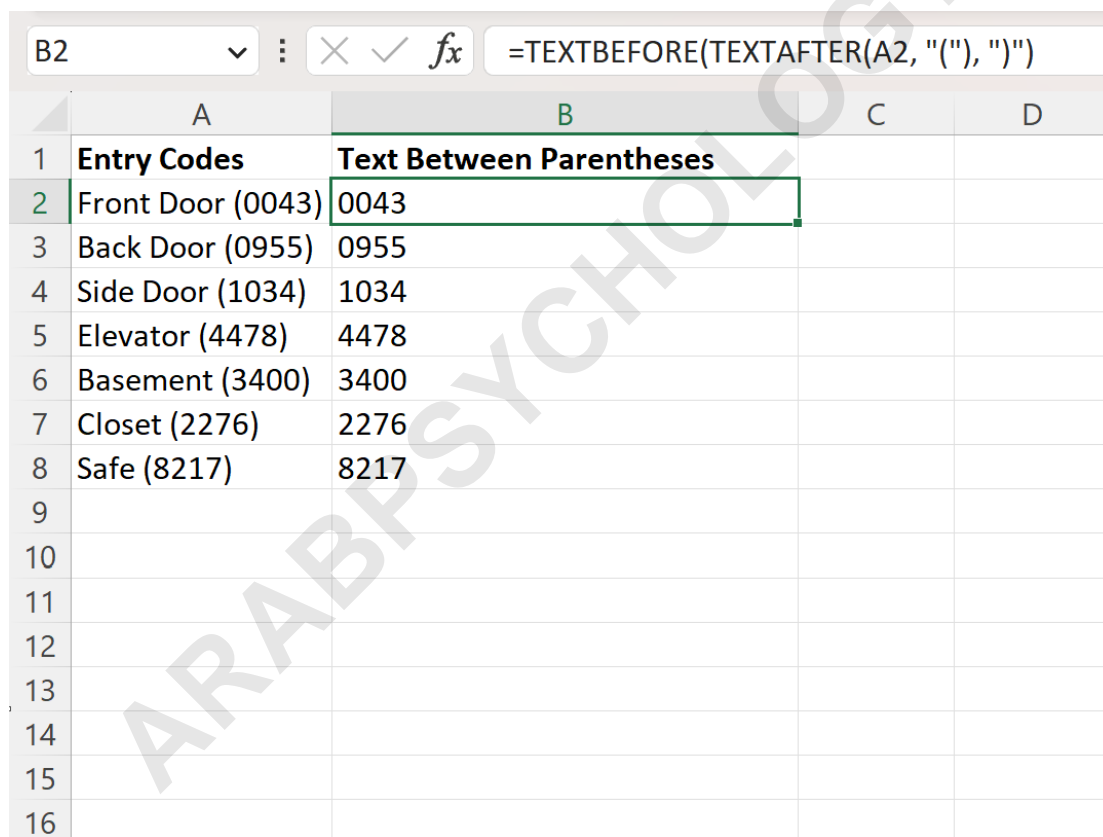
Another common requirement is the need to extract notes, identifiers, or codes that are typically encapsulated within standard punctuation, such as parentheses. Parentheses, round brackets, or curvature brackets, are frequently used to provide supplementary information. Isolating this supplementary data is a perfect use case for the nested TEXTBEFORE and TEXTAFTER functions.

To target the text contained within parentheses, the opening parenthesis "(" acts as the starting delimiter, and the closing parenthesis ")" serves as the ending delimiter. Note that when dealing with single characters that serve as delimiters, they must still be enclosed in double quotation marks within the formula structure, just like full text strings.

The formula to extract the text in source cell **A2** that is positioned between these delimiters is:

**=TEXTBEFORE(TEXTAFTER(A2, "("), ")")**

Once this formula is placed in the initial result cell (e.g., **B2**), the process for applying it to the entire column is identical to the previous example. We simply utilize the fill handle to drag the formula down to cover all relevant rows in column B. This ensures uniformity and eliminates repetitive manual input across the dataset.



	A	B	C	D
1	<b>Entry Codes</b>	<b>Text Between Parentheses</b>		
2	Front Door (0043)	0043		
3	Back Door (0955)	0955		
4	Side Door (1034)	1034		
5	Elevator (4478)	4478		
6	Basement (3400)	3400		
7	Closet (2276)	2276		
8	Safe (8217)	8217		
9				
10				
11				
12				
13				
14				
15				
16				

The resulting column B will then display only the content that was originally enclosed within the parentheses for each corresponding entry in column A. This demonstrates the formula's ability to handle special characters as delimiters effectively, yielding clean and isolated data segments.

### Example 3: Extracting Text Delimited by Asterisks

Beyond common punctuation, structured data sometimes employs less conventional characters, such as the asterisk (\*), as a separator or marker to highlight specific information. This is frequently seen in data cleansing scenarios or when importing raw data that uses proprietary formatting schemes. The nested formula approach remains entirely valid and robust for handling these special characters.

To extract content bracketed by asterisks, we set both the starting and ending delimiters to the asterisk character, "\*". This signals to the **TEXTAFTER** [function](#) where the extraction should begin, and to the **TEXTBEFORE** [function](#) where it must terminate. This dual-application is crucial when the delimiters are identical.

The complete formula to extract the text in source [cell A2](#) that is located between the asterisks is constructed as follows:

```
=TEXTBEFORE(TEXTAFTER(A2, "*"), "*")
```

This formula first locates the initial asterisk in [cell A2](#) and returns all text following it. It then takes that resulting string and extracts everything before the next occurrence of the asterisk. It is essential to ensure that the source string contains at least two instances of the delimiter for this formula to function correctly; otherwise, an error or unexpected result may occur.

	A	B	C	D
1	<b>Entry Codes</b>	<b>Text Between Asterisks</b>		
2	Front Door *0043*	0043		
3	Back Door*0955*	0955		
4	Side Door *1034*	1034		
5	Elevator *4478*	4478		
6	Basement *3400*	3400		
7	Closet *2276*	2276		
8	Safe *8217*	8217		
9				
10				
11				
12				
13				
14				
15				

Following the application of the formula and the necessary drag operation across the dataset, column B will accurately contain only the textual data that was enclosed between the asterisks for each entry in column A. This successfully demonstrates the versatility of the nested **TEXTBEFORE** and **TEXTAFTER** functions when dealing with specialized or repetitive delimiters.

## Refining Extraction Logic: Handling Multiple Occurrences

While the examples above assume a single pair of delimiters, the **TEXTBEFORE** and **TEXTAFTER** functions offer additional optional arguments that enhance precision, particularly when the delimiter appears multiple times within the source string. These arguments allow the user to specify which instance of the delimiter should be used for the boundary definition.

For instance, if a string contains data structured as "ID1-DataA-DataB-ID2," and you wish to extract "DataB," you would need to specify the instance count. In the **TEXTAFTER** function (to find the beginning of "DataB"), you would search after the second hyphen. In the **TEXTBEFORE** function (to find the end of "DataB"), you would search before the third hyphen. This level of control ensures accurate data parsing even in highly complex data structures.

Understanding the optional arguments, such as **instance\_num** and **match\_mode**, is crucial for advanced data manipulation. The **instance\_num** parameter dictates which occurrence of the delimiter to use--positive numbers count from the start, and negative numbers count from the end of the string. Utilizing these features transforms the basic nested formula into a powerful tool

capable of handling virtually any delimited data scenario within Excel.

## Conclusion and Best Practices for Text Extraction

The combined use of the TEXTAFTER and TEXTBEFORE functions provides a modern, clean, and highly efficient methodology for extracting text segments bracketed by two specific characters or strings in Excel. This approach stands in stark contrast to the older, more cumbersome reliance on complex nested structures involving **MID** and **SEARCH** functions, which often led to difficult troubleshooting and poor readability.

To ensure the highest reliability when applying these techniques, always adhere to the following best practices. First, confirm that your delimiters are accurately specified within double quotes, even if they are single characters. Second, test the formula on a subset of data to verify that it correctly accounts for edge cases, such as missing delimiters or the presence of multiple delimiter pairs. Finally, utilize relative cell referencing and the drag-and-fill feature to efficiently scale the solution across large datasets, maximizing productivity.

By integrating these powerful textual functions into your data analysis toolkit, you gain the ability to parse complex strings with precision and speed, making data preparation tasks significantly simpler and more robust.