Learn to Extract Text Between Two Characters in Excel

Authored by

Mohammed looti

November 14, 2025

In the demanding world of **Microsoft Excel**, mastering the efficient manipulation of **text strings** is a crucial skill for data analysts and power users alike. Often, raw datasets contain vital pieces of information that are tightly embedded within longer text sequences, typically marked off by specific characters or phrases. Attempting to extract this precise data manually can be laborious and highly susceptible to errors. Fortunately, modern versions of **Excel** offer sophisticated and straightforward functions--specifically **TEXTBEFORE** and **TEXTAFTER**--that simplify this formerly complex task. These powerful tools are engineered to parse and isolate text based on user-defined **delimiters**, thereby making your **data manipulation** efforts significantly faster and more reliable.

This comprehensive guide is designed to walk you through the practical application of combining the **TEXTBEFORE** and **TEXTAFTER** functions. Our primary objective is to demonstrate how to precisely extract any text segment that is situated between two specific characters or **strings** within a single **cell**. We will examine the core logic, detail the necessary **syntax**, and explore several real-world examples to ensure you develop a thorough, actionable understanding. By mastering this combined technique, you will gain a robust capability to clean, organize, and analyze your data more effectively, seamlessly transforming raw, unstructured text into structured, actionable information.

The general **syntax** required for this extraction involves a powerful nesting structure: placing the **TEXTBEFORE** function inside the **TEXTAFTER** function. This nested arrangement executes a two-stage filtering process: it first identifies the starting boundary of your desired text and then, operating on the truncated result, isolates the content before the ending boundary. This structure serves as the essential blueprint for all text-between extractions.

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

In the foundational illustration above, the **formula** is specifically engineered to extract all characters located between the exact **strings** "this" and "that," based on the contents of **cell** **A2**. The inner **TEXTAFTER** function executes first, processing the **cell** content and discarding everything up to and including the first occurrence of "this." The resulting shortened **string** is then passed to the outer **TEXTBEFORE** function, which extracts everything before the first instance of "that." This precise methodology guarantees the accurate isolation of the desired text segment, minimizing noise and maximizing data quality.

To fully appreciate the versatility and power of this combined approach, the following sections will present several highly practical examples. These scenarios are designed to illustrate common data challenges, ranging from extracting specific keywords to isolating data segments enclosed within special characters like parentheses or asterisks. Each example includes a clear statement of the problem, the exact **formula** utilized, and a visual representation of the outcome. By working through these demonstrations, you will be well-equipped to apply these robust techniques to your

own **Excel** data cleansing and analysis projects.

Understanding the Core Functions: TEXTBEFORE and TEXTAFTER

Before implementing complex, nested **formulas**, it is essential to establish a firm understanding of how **TEXTBEFORE** and **TEXTAFTER** operate individually. The **TEXTBEFORE** function is used to extract the portion of text that precedes a specified **delimiter** within a given **text string**. Its comprehensive **syntax** is documented as `TEXTBEFORE(text, delimiter, , , , )`. In this structure, the `text` argument specifies the **cell** reference or literal **string** to be searched, while `delimiter` defines the character or sequence of characters that marks the end boundary of the desired extracted text. Optional **arguments**, such as `instance_num`, provide crucial control, enabling the user to specify which occurrence of the **delimiter** should be used for the extraction point.

The **TEXTAFTER** function, conversely, performs the opposite operation: it is engineered to extract all text that appears immediately following a specified **delimiter**. Its **syntax** precisely mirrors **TEXTBEFORE**: `TEXTAFTER(text, delimiter, , , , )`. Here, the `delimiter` specifies the starting boundary of the text intended for extraction. Both functions share common optional parameters, including `match_mode`, which controls **case sensitivity** (using 0 for case-sensitive and 1 for case-insensitive matching), `match_end` for boundary conditions, and `if_not_found`, which allows you to define a custom value to return if the specified **delimiter** is absent from the source string.

The true efficiency of these functions is realized when they are nested together, establishing a sophisticated two-step text isolation mechanism. The inner **TEXTAFTER** function acts as the initial filter, efficiently trimming the original **text string** by removing all content up to and including the starting **delimiter**. The resultant, truncated **string** is then seamlessly passed as input to the outer **TEXTBEFORE** function. This outer function completes the process by extracting all characters from the input string up to, but not including, the ending **delimiter**. This methodological nesting ensures that only the desired text, precisely bounded by your specifications, is returned as the final, clean result, providing unparalleled precision in text parsing.

The Combined Approach: Isolating Text Between Two Delimiters

The combined power of **TEXTBEFORE** and **TEXTAFTER** is best realized when addressing the common data challenge of isolating text segments nestled between two distinct **delimiters**. The standard and most effective **formula** structure for this purpose is `=TEXTBEFORE(TEXTAFTER(cell, "start_delimiter"), "end_delimiter")`. This concise and readable solution vastly simplifies the extraction of specific data points from complex **text strings**--a task that previously necessitated cumbersome nesting of older functions like **MID**, **FIND**, and **SEARCH**. The modern nested

TEXTBEFORE/**TEXTAFTER** pattern offers vastly improved clarity and maintainability in all modern **Excel** environments.

The logical execution of this combined **formula** is both straightforward and highly efficient. The inner function, **TEXTAFTER**, initiates the extraction by taking the complete **text string** from the source **cell** (e.g., A2) and returning every character that follows the designated "start_delimiter." This initial step effectively generates a new, shorter string that starts precisely where the desired segment begins. This intermediate result then becomes the required `text` **argument** for the outer **TEXTBEFORE** function. The outer function then operates on this truncated string, extracting all content that occurs before the "end_delimiter." The final output is precisely the text segment originally located between the two specified **delimiters**, guaranteeing high accuracy and efficiency.

When deploying this powerful technique, it is vital to consider several key factors to ensure robust and error-resistant **data manipulation**. Notably, the default behavior of these functions is **case sensitivity**. If your **delimiters** might have inconsistent casing (e.g., "Start" vs. "start"), you must utilize the optional `match_mode` **argument**, setting it to `1` for a case-insensitive match. Furthermore, if your source **text string** contains multiple instances of the same **delimiter**, the `instance_num` **argument** becomes indispensable. This parameter allows you to precisely specify which occurrence of the **delimiter** should be used for both the start and end boundaries, preventing ambiguity and ensuring the formula extracts the exact intended segment of data.

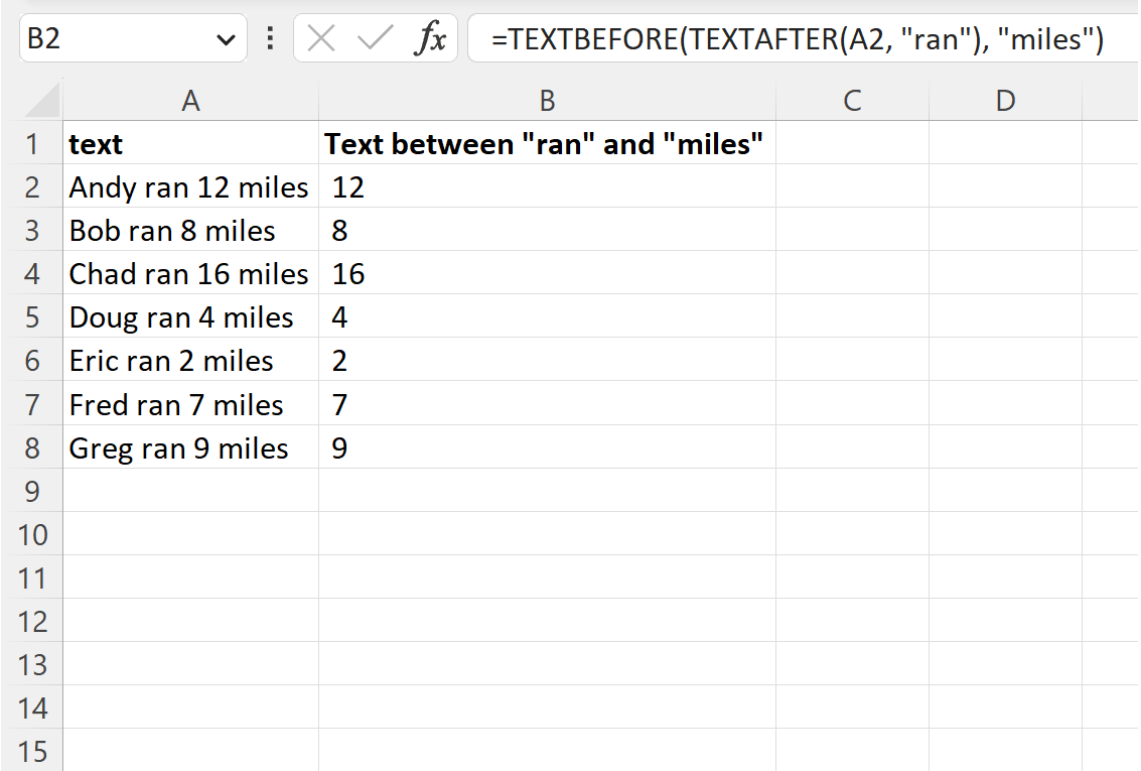
Example 1: Extracting Text Between Specific Strings

Imagine a scenario where you are presented with a column of unstructured descriptive phrases in **Excel**. Within each phrase, a specific numerical value, such as a distance or quantity, is consistently bracketed by two distinct textual **strings**. For illustration, consider entries like "John ran 5 miles quickly" or "Sarah ran 10 miles slowly." Our goal is to isolate the numerical distance, which is invariably found between the starting string "ran" and the ending string "miles." This extraction technique is critically important for quickly converting descriptive textual data into structured numerical data, which is necessary for subsequent calculations and insightful analysis.

To achieve this precise extraction, we leverage the highly effective combination of the **TEXTBEFORE** and **TEXTAFTER** functions. Assuming the original phrases reside in column A, we would input the following **formula** into **cell** **B2**: `=TEXTBEFORE(TEXTAFTER(A2, "ran"), "miles")`. The inner segment, **TEXTAFTER**(A2, "ran"), first processes the **text string** in **A2** and returns everything subsequent to the string "ran." If **A2** contains "John ran 5 miles quickly," this step yields " 5 miles quickly." Subsequently, the outer **TEXTBEFORE** function receives this result and extracts everything before the **string** "miles," resulting in the clean output " 5".

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

Once this extraction **formula** is accurately placed in **cell** **B2**, the process of applying it across your entire dataset is streamlined using the fill handle. By clicking and dragging the small square at the bottom-right corner of **cell** **B2** down the column, you effortlessly replicate the formula for all corresponding rows. This action ensures that **Excel** automatically adjusts the **cell** references (A2 becomes A3, A4, and so on) for each entry, guaranteeing that the extraction is performed correctly for every original **text string** in column A.



	A	B	C	D
1	**text**	**Text between "ran" and "miles"**		
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				

The result in column B will be a meticulously structured list containing only the extracted numerical values. This transformation effectively converts a free-form, descriptive text entry into a clean, standardized numeric data point, ready for immediate aggregation, charting, or further mathematical operations. This showcases the immense utility of the combined **TEXTBEFORE**/**TEXTAFTER** functions in advanced **data manipulation** and analysis within **Excel**.

Example 2: Extracting Text Enclosed by Parentheses

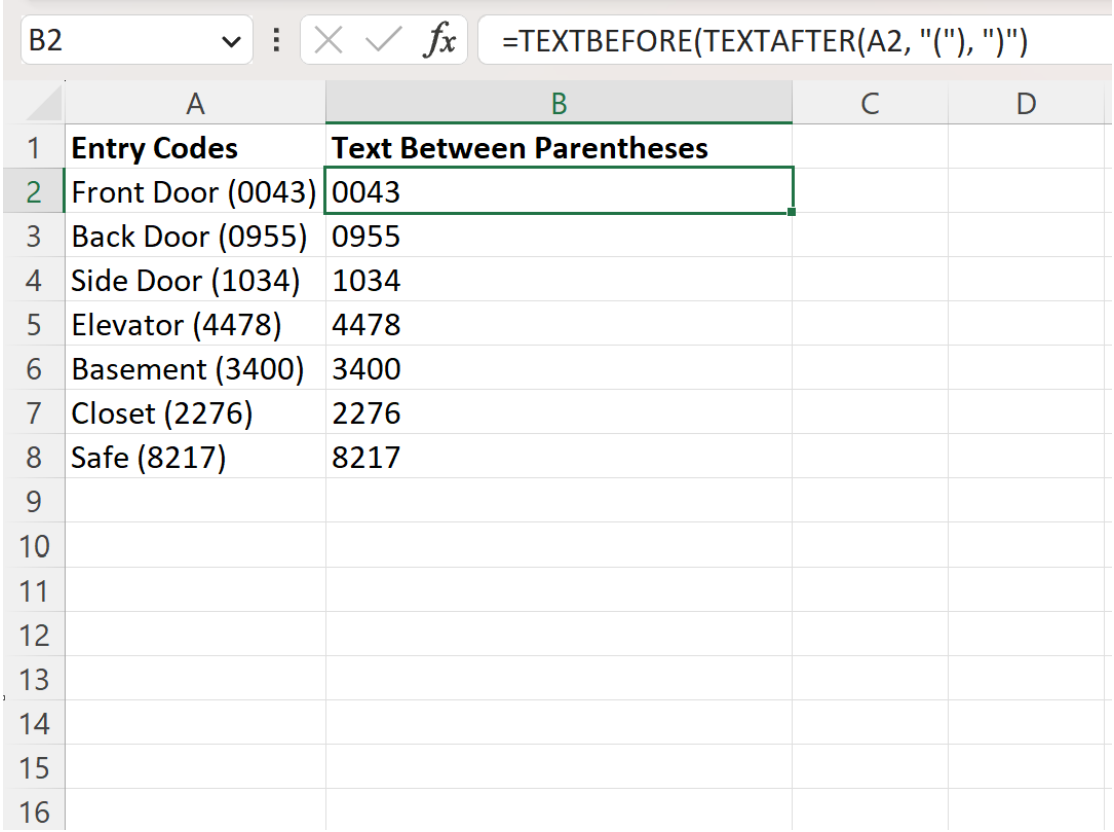
A frequent requirement in **data manipulation** involves isolating information that has been encapsulated within parentheses. These segments often contain critical details such as product specifications, internal identifiers, or supplementary classification codes within a larger **text string**. Extracting this parenthesized content is a necessary step for data cleansing and normalization. Fortunately, the **TEXTBEFORE** and **TEXTAFTER** functions are perfectly suited to this task, as they can treat the opening parenthesis `(` and the closing parenthesis `)` as precise **delimiters**. This

method offers a streamlined and robust solution compared to relying on complex legacy formulas or error-prone manual parsing.

To extract the text strictly enclosed within parentheses, we simply adapt our standard nested **formula** structure. The crucial step is defining the literal parenthesis characters as our starting and ending **delimiters**. Assuming the source **text string** is in **A2**, we input the following **formula** into **cell** **B2**: `=TEXTBEFORE(TEXTAFTER(A2, "("), ")")`. The inner **TEXTAFTER**(A2, "(") segment first processes the content of the **cell**, returning everything that follows the initial opening parenthesis. This intermediate result is then passed to the outer **TEXTBEFORE** function, which extracts all text appearing before the first closing parenthesis in that truncated **string**.

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

Once this extraction **formula** is established in **cell** **B2**, replicating this logic across the rest of your dataset is simple. By utilizing the fill handle--the small square at the bottom-right corner of the selected **cell**--you can quickly drag the formula down to cover all corresponding rows in column B. This automated process ensures that every **text string** in column A is individually parsed, and the content enclosed by parentheses is accurately extracted and displayed in the corresponding **cell** of column B. This significantly accelerates data processing, especially when managing high volumes of data, while ensuring consistent and accurate results.



Upon completion, column B will showcase a cleanly extracted list of all **text strings** that were originally encapsulated within parentheses in column A. This result vividly demonstrates the effectiveness of using **TEXTBEFORE** and **TEXTAFTER** for highly targeted **data manipulation**. The ability to isolate specific parts of text based on simple **delimiters** such as parentheses is a foundational skill for analysts and anyone working with semi-structured text data in **Excel**.

Example 3: Extracting Text Between Asterisks

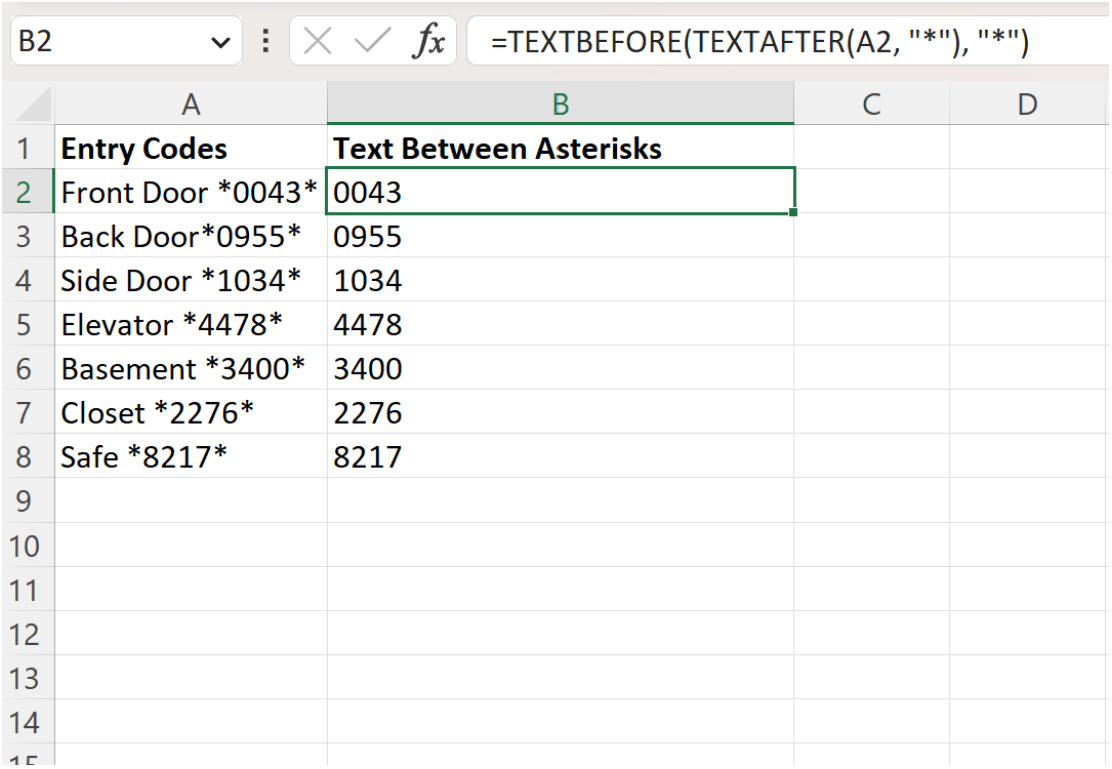
Beyond common punctuation and alphanumeric **strings**, special characters are frequently used to demarcate important data points. The asterisk (`*`), for instance, is often employed as a **delimiter** to highlight keywords or unique identifiers in systems like inventory logs or product descriptions (e.g., "Item *RED-PNT-XL* available"). Extracting the content situated between these asterisks is a necessary step for data normalization, facilitating efficient searching, filtering, and seamless database integration. The consistent nature of the asterisk as a boundary marker makes it an excellent candidate for isolation using the combined power of **TEXTBEFORE** and **TEXTAFTER**.

To successfully extract text located between two asterisks, the core logical principle remains identical to the previous examples. The only change required is substituting the specific **string** or character **delimiters** with the asterisk symbol (`"*"`). Thus, in **cell** **B2**, referencing the original **text string** in **A2**, the required **formula** is: `=TEXTBEFORE(TEXTAFTER(A2, "*"), "*")`. This concise

syntax directs **Excel** to perform two sequential steps: first, identify the content immediately following the initial asterisk using **TEXTAFTER**, and second, from that resultant string, extract everything that precedes the next asterisk using **TEXTBEFORE**.

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

Once the **formula** has been correctly entered into **cell** **B2**, the procedure for applying it throughout your entire dataset is straightforward. You simply select **cell** **B2** and drag the fill handle downwards to extend the formula to the remaining cells in column B. **Excel** automatically manages the adjustment of **cell** references, guaranteeing that each **text string** in column A is processed correctly, and the text enclosed by the asterisks is precisely extracted. This systematic and repeatable approach saves significant time and drastically reduces the potential for human error inherent in manual data cleaning processes.



The outcome in column B will be a clean and concise list containing only the **text strings** originally situated between the asterisks in column A. This demonstrates the robust versatility of **TEXTBEFORE** and **TEXTAFTER** in handling diverse **delimiters**, including special characters, ensuring reliable and accurate **data manipulation**. By converting these embedded values into distinct, usable data points, users significantly enhance the analytical capabilities of their **Excel** workbooks.

Advanced Considerations and Robust Error Handling

While the fundamental application of **TEXTBEFORE** and **TEXTAFTER** is intuitive, leveraging their optional **arguments** provides enhanced precision and versatility for complex **Excel** tasks. A primary consideration is managing **case sensitivity**. By default, both functions perform a case-sensitive search when locating **delimiters**. If your source **text string** data may contain delimiters in various cases (e.g., "CODE" versus "code"), you can modify the `match_mode` **argument** to `1`. This setting enables a case-insensitive search, ensuring your extraction **formulas** are resilient against minor inconsistencies in the source data and prevent valuable information from being missed due to casing differences.

The `instance_num` **argument** is another incredibly powerful parameter, particularly when your **text string** contains multiple identical **delimiters**, and you need to specify which occurrence marks the boundary of the desired text. For example, in a string like "Name: John, ID: 1234, Dept: Sales," if you want to extract the ID, you would need to tell **TEXTAFTER** to use the second comma as the starting point, and **TEXTBEFORE** to use the third comma as the endpoint. By carefully specifying positive or negative values for `instance_num` (positive counts from the start, negative counts from the end), you can precisely target virtually any segment, even within highly complex, multi-delimited strings, offering unparalleled flexibility in text parsing.

Furthermore, incorporating robust error handling is paramount for creating reliable **formula** designs. If a required **delimiter** is absent from the source **text string**, both **TEXTBEFORE** and **TEXTAFTER** will default to returning a problematic `#VALUE!` error. To gracefully manage these situations, you should utilize the optional `if_not_found` **argument**. By setting this argument to an empty **string** (`""`) or a specific, user-friendly message (e.g., `"Boundary Missing"`), you can prevent error propagation throughout your spreadsheet and make your data output much cleaner and easier to interpret. For users migrating from older **Excel** versions (pre-Microsoft 365), these modern functions replace the necessity of combining complex **MID**, **FIND**, and **SEARCH** combinations, representing a massive simplification in text handling capabilities.

Conclusion

The capacity to efficiently parse and extract targeted information from complex **text strings** is fundamentally important for effective **data manipulation** in **Excel**. The introduction of the dedicated **TEXTBEFORE** and **TEXTAFTER** functions has revolutionized this process, providing a uniquely clear, concise, and highly effective nested method for isolating text between any two specified **delimiters**. As demonstrated across several practical examples, these functions possess the versatility required to handle a wide range of real-world scenarios, ensuring high accuracy and dramatically improving the readability of your **formulas**.

By mastering the nested application of **TEXTBEFORE** and **TEXTAFTER**, along with judicious use of optional **arguments** like `instance_num` and `if_not_found`, users are fully equipped to confidently tackle complex text parsing challenges. These essential tools empower you to transform raw, unstructured text data into clean, structured, and actionable information, which is indispensable for accurate analysis, reporting, and integration into other organizational systems. We highly recommend practicing these **formulas** with your own datasets to fully grasp their efficiency and the significant time savings they can bring to your daily **Excel** workflows.

Additional Resources for Advanced Excel Mastery

For those eager to expand their proficiency in **Excel** and explore complementary data manipulation techniques, the following curated tutorials offer valuable insights into related common tasks. Expanding your knowledge base beyond specific text extraction methods can significantly enhance your overall productivity and analytical capabilities within the spreadsheet environment, enabling you to manage and analyze data more comprehensively.