Extracting Text Between Spaces: A Step-by-Step Guide for Excel

Authored by

Mohammed looti

November 9, 2025

Introduction to Advanced Text Manipulation in Excel

Microsoft **Excel** stands as the undisputed industry standard for sophisticated data management, comprehensive analysis, and complex numerical computations. While its foundation rests primarily on handling quantitative data, modern data processing frequently necessitates advanced text operations. This crucial process, commonly termed **string manipulation**, is essential for cleaning, standardizing, and preparing raw textual datasets sourced from external systems or reports. A recurring and often complicated challenge for data professionals is the precise extraction of specific segments--known as substrings--that are clearly demarcated by two known occurrences of a **delimiter**, such as a space, comma, or pipeline character. Historically, achieving this level of precision in older versions of Excel demanded the construction of convoluted, error-prone nested formulas that relied heavily on positional indices, making the process time-consuming and difficult to audit.

The landscape of text handling in Excel has been dramatically simplified and modernized through the introduction of powerful, declarative text functions available in Microsoft 365 and recent perpetual versions. This tutorial focuses on demonstrating a highly efficient and readable technique that expertly utilizes the complementary capabilities of the **TEXTBEFORE** and **TEXTAFTER** functions. These modern tools empower users to isolate desired text segments based not just on the delimiter type, but also on its specific numerical occurrence within the source string. Our specific focus will be on illustrating how to efficiently isolate the text located exactly between the second and third spaces within a single cell. This precise requirement frequently arises when attempting to parse structured data formats, such as extracting middle names from full names, isolating specific elements from complex product identifiers, or parsing address components.

By mastering the strategic nesting of these two contemporary functions, users can fundamentally transform the efficiency and maintainability of their data cleaning and preparation workflows. The resulting formulas are significantly more robust, intuitive, and readable compared to their legacy counterparts. Furthermore, understanding the underlying logic of how to combine these boundary-defining functions provides a versatile and scalable framework for addressing a multitude of diverse text extraction scenarios that depend entirely on identifying and targeting specific positional markers within a given text **string**. This method minimizes reliance on error-prone character counting and maximizes the formula's ability to adapt to varying text lengths.

Understanding the Core Functions: TEXTBEFORE and TEXTAFTER

The remarkable effectiveness and efficiency of this specialized extraction technique are built upon the precise, complementary functionality inherent in the two primary modern text functions. The **TEXTAFTER** function is specifically engineered to return every character and segment of text that follows a defined **delimiter**. Its essential syntax requires three core arguments: the source text, the

delimiter character (e.g., " " for space), and, most critically for positional extraction, an optional instance number. This instance number explicitly instructs Excel which occurrence of the delimiter should be used as the starting boundary. For instance, if a user specifies '2' as the instance number and ' ' (a space) as the delimiter, the function intelligently returns everything from the character immediately following the second space throughout the remainder of the input text string.

In contrast, the **TEXTBEFORE** function performs the reverse operation: it systematically extracts the text that precedes a specified delimiter. When this function is used on its own, it typically defaults to returning the text leading up to the first instance of the specified delimiter, unless a specific instance number is provided to target a later occurrence. The real power, however, emerges when these two functions are strategically nested together. This nesting creates a highly functional and versatile text slicing mechanism. The output generated by the inner function--which effectively establishes the exact starting boundary for the desired text--is then seamlessly fed as the primary input text argument for the outer function. The outer function subsequently defines the precise ending boundary, isolating the target segment between the two defined markers.

For our specific goal--extracting the text segment located exactly between the second and third spaces--the strategic nesting order is paramount to success. We must first employ the **TEXTAFTER** function to isolate a large, truncated segment of the original string, ensuring that this new segment begins precisely after the second space. Once this starting point is established, we then apply the **TEXTBEFORE** function to this resulting, shortened segment. This final step captures only the desired word or phrase that appears before the very next space, which, in the context of the original complete string, corresponds exactly to the third space. This two-step process elegantly converts a complex positional problem into a simple sequential operation.

Implementing the Combined Extraction Formula

To successfully extract the specific textual information situated between the second and third spaces in a designated cell, such as **A2**, we deploy a highly effective and remarkably concise syntax. This formula represents the modern, clean solution to a text parsing challenge that historically demanded extensive calculation of character indices and complex string arithmetic. The elegance of the modern text functions lies in their declarative nature, allowing users to specify **what** they want (text after the second space, text before the next space) rather than **where** it is (character positions).

The required structure for this powerful extraction formula, which will be utilized throughout our practical examples, is defined as follows:

=TEXTBEFORE(TEXTAFTER(A2, " ", 2), " ")

This structural configuration is exceptionally efficient, highly readable, and dramatically reduces the potential for common errors associated with positional formulas. The inner calculation, specifically the component `TEXTAFTER(A2, " ", 2)`, executes first and serves the critical role of defining the start boundary. It systematically trims the beginning of the source string, discarding all characters and the spaces leading up to and including the second space. The immediate result of this operation is a newly truncated text string, which now commences exactly after the second space. Subsequently, the outer function, **`TEXTBEFORE`**`(..., " ")`, is applied to this truncated result. Since we deliberately omit the optional instance number argument in this outer **TEXTBEFORE** function, it defaults to extracting all text that precedes the *first* occurrence of the space within the newly received input string. Critically, this first space encountered corresponds precisely to the third space of the initial input string, ensuring the extracted text is perfectly bounded.

This single, nested formula can be effortlessly adapted to reference any cell containing the source text string you intend to parse, providing a scalable and reliable method that eliminates the need for redundant manual calculations or the creation of cumbersome helper columns. The simplicity of this approach drastically reduces the potential for error compared to manual calculation of character positions or the overuse of helper columns.

Step-by-Step Practical Application

To provide a clear demonstration of this formula's practical utility, let us consider a typical data analysis scenario. Imagine we are working with a dataset comprising various descriptive codes or phrases, and our specific analytical objective requires us to isolate the third word--which is intrinsically defined as the segment of text residing between the second and third spaces--from every entry. Suppose our sample data is organized in Column A, starting from cell **A2**, as depicted in the following table visualization:

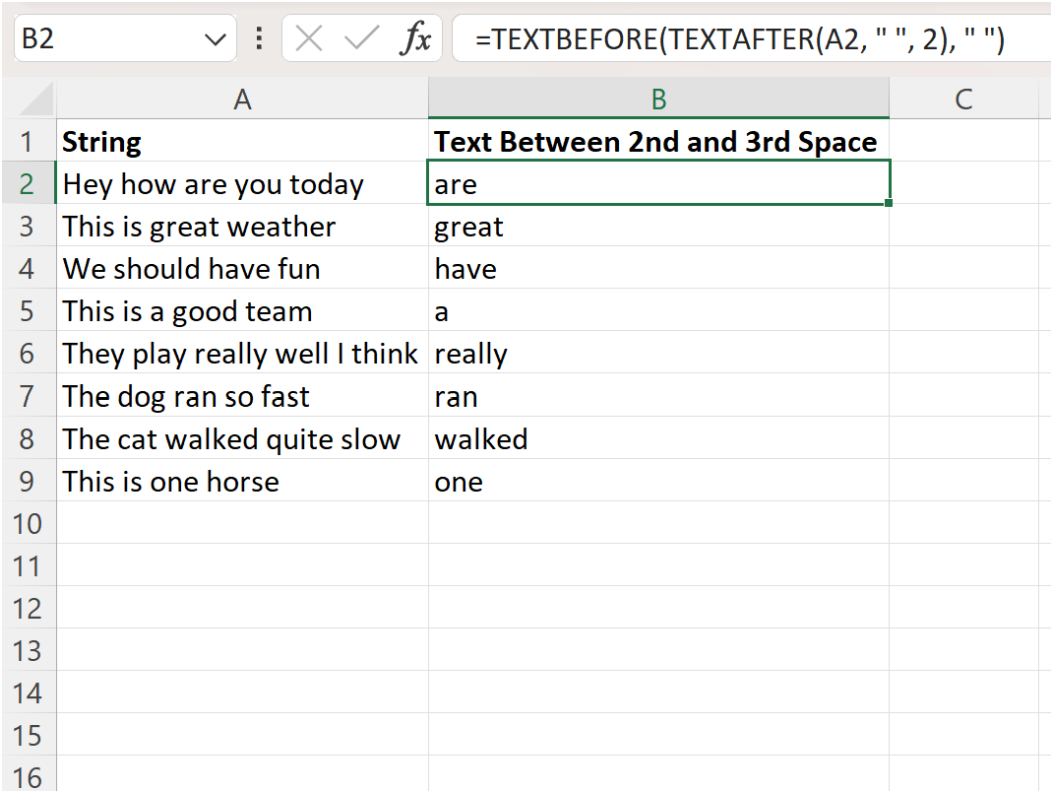
	A	B	C	D
1	**String**			
2	Hey how are you today			
3	This is great weather			
4	We should have fun			
5	This is a good team			
6	They play really well I think			
7	The dog ran so fast			
8	The cat walked quite slow			
9	This is one horse			
10				
11				
12				
13				
14				
15				
16				
17				

Our goal is to populate Column B with the extracted, targeted text segment corresponding to each source string in Column A. This extraction must be precise, capturing only the content that exists exactly between the second and third space characters throughout the list.

We initiate the extraction process by entering the complete target formula into cell **B2**. It is essential to ensure that the formula accurately references the first data entry in **A2**:

=TEXTBEFORE(TEXTAFTER(A2, " ", 2), " ")

Once the formula is correctly input, we execute it by pressing the Enter key to calculate the result for the inaugural row. Following successful calculation, we leverage Excel's efficient auto-fill mechanism. This involves clicking and dragging the small fill handle--the square located at the bottom-right corner of cell B2--downwards across the range of cells in Column B corresponding to our data. This action triggers the automatic adjustment of the relative cell references (A2 sequentially becomes A3, A4, and so forth) for every subsequent row, consistently applying the precise and accurate extraction logic across the entirety of the dataset.



	A	B	C
1	**String**	**Text Between 2nd and 3rd Space**	
2	Hey how are you today	are	
3	This is great weather	great	
4	We should have fun	have	
5	This is a good team	a	
6	They play really well I think	really	
7	The dog ran so fast	ran	
8	The cat walked quite slow	walked	
9	This is one horse	one	
10			
11			
12			
13			
14			
15			
16			

As clearly demonstrated by the resulting table, Column B now contains text segments that were accurately and exclusively situated between the second and third spaces of their corresponding source strings in Column A. This successful outcome firmly validates the efficacy, reliability, and sheer ease of use provided by the nested **TEXTBEFORE** and **TEXTAFTER** structure, confirming its status as the superior method for accurate positional text extraction in modern Excel environments.

Deconstructing the Formula Logic for Precision

To fully grasp the intricate yet elegant mechanism underpinning this solution, it is crucial to meticulously deconstruct the exact sequence in which **Excel** evaluates the complete formula: `=TEXTBEFORE(TEXTAFTER(A2, " ", 2), " ")`. As with all nested functions, the evaluation proceeds systematically from the innermost parentheses outward, ensuring that the results of the inner functions are correctly prepared for the outer functions. This two-step process guarantees the accurate isolation of the target substring.

Inner Function Execution: TEXTAFTER(A2, " ", 2)

The calculation process always begins with the inner function, which defines our critical starting boundary. The specified parameters instruct Excel to target the content of cell **A2**, use the space character (" ") as the designated delimiter, and crucially, begin the extraction after the second instance of that delimiter. Consider the sample string used previously, "Hello world are you today".

The second space occurs immediately following the word "world". Consequently, this inner function successfully executes its task by returning the resulting, truncated text segment: **are you today**. This result is now ready to be processed by the final extraction function.

Outer Function Execution: TEXTBEFORE(Result, " ")

The intermediate result ("are you today") is subsequently passed as the primary input argument to the outer **TEXTBEFORE** function. The calculation now effectively translates to `TEXTBEFORE("are you today", " ")`. Since the instance number argument has been intentionally omitted from this outer function, **TEXTBEFORE** operates under its default behavior: extracting everything that precedes the very first space it encounters within this newly trimmed string. The first space in "are you today" separates the word "are" and the word "you". Therefore, the function returns the final, isolated text segment: **are**.

The final returned result, **are**, is precisely the word that was originally situated between the second and third spaces of the input string located in **A2**. This systematic, precise two-step evaluation confirms that the chosen nesting structure is flawlessly designed to isolate the desired substring by meticulously defining the necessary starting boundary (achieved via **TEXTAFTER**) and then defining the required ending boundary (achieved via **TEXTBEFORE**) relative to that newly established starting point. This logical flow is what makes the modern solution so robust and reliable.

Alternative Text Extraction Methods in Legacy Excel

While the combination of **TEXTBEFORE** and **TEXTAFTER** represents the most modern, elegant, and highly recommended approach for users utilizing Excel subscriptions that support these advanced features, it remains essential to understand the methods required for older, legacy versions of Excel (specifically pre-Microsoft 365 or pre-Excel 2021). These older environments lack the declarative nature of the modern functions and necessitate significantly more complex string manipulation relying on fundamental positional functions. These traditional methods typically involve manually calculating precise character positions using functions like **FIND** or **SEARCH**, and then extracting the specific length of text using the **MID** function.

To successfully achieve the exact same extraction result--isolating the text between the second and third spaces--in legacy Excel versions, a user is required to construct a formula that first calculates the position of the second space (defining the start), then calculates the position of the third space (defining the end), and finally computes the exact length of the string segment residing between those two calculated indices. The resulting legacy formula, while functional, is substantially more cumbersome, demanding multiple complex instances of nested logic and significantly increasing the probability of error during implementation or auditing:

=MID(A2, FIND(" ", A2, FIND(" ", A2)+1)+1, FIND(" ", A2, FIND(" ", A2, FIND(" ", A2)+1)+1) -

(FIND(" ", A2, FIND(" ", A2)+1)+1))

A direct comparison between the lengthy, highly technical legacy formula above and the concise modern solution, `=TEXTBEFORE(TEXTAFTER(A2, " ", 2), " ")`, starkly highlights the immense productivity gains, clarity, and reduced cognitive load afforded by the newer functions. Users operating within environments that fully support **TEXTBEFORE** and **TEXTAFTER** should always prioritize the deployment of these solutions. Their superior readability, efficiency, and dramatically reduced complexity are invaluable for minimizing debugging time, streamlining data preparation, and maximizing overall data integrity across large projects. The legacy method serves as a strong reminder of how far Excel's text processing capabilities have evolved.

Additional Resources for Mastering Excel Text Functions

Mastering the art of text manipulation is an absolutely critical skill set for anyone serious about effective data preparation, cleaning, and analysis within the **Excel** environment. The advanced techniques demonstrated in this guide are highly adaptable and can be readily modified to extract text based on virtually any type of **delimiter**, whether the requirement involves working with standard characters like commas and hyphens, specialized symbols such as pipe or tilde, or even complex, multi-character sequences. We strongly encourage all data analysts and users to continue their exploration of Excel's extensive and powerful library of text functions, which are designed to handle even the most intricate parsing requirements and complex data transformation tasks.

Further enhancing your proficiency in text operations will enable you to solve a wider array of data cleaning problems. For example, knowing how to handle varying whitespace or how to replace certain characters can drastically improve data consistency. Similarly, understanding the difference between **FIND** (case-sensitive) and **SEARCH** (case-insensitive) is crucial when dealing with case-specific delimiters or keywords.

To assist in this ongoing mastery, the following tutorials and functions explain how to perform other common and essential text manipulation tasks in Excel and related topics:

How to Extract Text Between Parentheses in Excel using various methods, including conditional logic.
Using the **TRIM** function effectively for reliable data cleaning and standardization by removing extraneous leading and trailing spaces.
Combining Text using the modern **CONCAT** or **TEXTJOIN** functions, or the traditional Ampersand (&) Operator, for creating unified reports or merged identifiers.

By diligently integrating these powerful and modern functions into your daily data workflow, you

can effectively automate what were once complex and tedious data preparation tasks. This shift allows you to significantly reduce the need for manual intervention, minimize human error, and allocate more of your valuable time toward meaningful data analysis, interpretation, and strategic decision-making.