Learn How to Extract Substrings in Excel: A Comprehensive Guide with Examples

Authored by

Mohammed looti

November 10, 2025

Mastering data manipulation is essential for effective spreadsheet analysis, and one of the most common tasks in Microsoft Excel involves isolating specific parts of a text string. This process, known as extracting a substring, allows analysts to clean data, parse codes, or separate names from complex identifiers. We will explore the most robust and flexible formulas available in Excel to accomplish this, categorized into methods based on position or based on surrounding text delimiters.

To efficiently extract certain substrings from a text string, you can utilize the following five powerful formulas. These methods cover virtually every scenario, whether you know the exact position or rely on surrounding characters.

Summary of Substring Extraction Methods

Method 1 (LEFT): Extracts characters starting from the beginning of the string.
Method 2 (MID): Extracts characters from the middle of the string based on a starting position and length.
Method 3 (RIGHT): Extracts characters starting from the end of the string.
Method 4 (TEXTBEFORE): Extracts text preceding a specified delimiter.
Method 5 (TEXTAFTER): Extracts text following a specified delimiter.

The following formulas provide a quick reference for each approach, assuming the source text is contained within cell **A2**:

Method 1: Extracting Substring from Beginning of String

To retrieve a fixed number of characters from the left side of a string, the LEFT() function is the tool of choice. This is invaluable when dealing with structured data like product codes or regional prefixes that always appear at the start of a cell's content.

```
#return first 3 characters of string in cell A2
=LEFT(A2, 3)
```

The **LEFT()** function requires two primary arguments: the target text string (or cell reference) and the number of characters you wish to extract. If the second argument is omitted, Excel defaults to returning only the first character. This method is highly efficient but relies entirely on positional data; if the desired information varies in length or placement, a more complex solution involving **FIND()** or **SEARCH()** might be necessary, though for fixed-length prefixes, **LEFT()** is perfect.

Method 2: Extracting Substring from Middle of String

When the desired substring is embedded deep within the text and neither the beginning nor the

end, the MID() function provides the necessary precision. Unlike **LEFT()** or **RIGHT()**, **MID()** requires three arguments: the text string, the starting position, and the number of characters to return. This combination gives you granular control over the extraction process, making it suitable for parsing IDs where crucial information starts at a specific index.

#return 8 characters of string in cell A2 starting at position 2
=MID(A2, 2, 8)

The power of **MID()** comes from its ability to start the extraction anywhere within the string. For instance, in the example above, we begin counting from the second character and extract the next eight characters. If the starting position is greater than the total length of the text, the function returns an empty string. If the requested length extends beyond the string's end, **MID()** simply returns all remaining characters. This makes it a robust tool for fixed-format data sets where the starting point is always known.

Method 3: Extracting Substring from End of String

Conversely, when the required data segment is located at the tail end of the text, the RIGHT() function is used. This is common when extracting extensions (like ".csv" or ".pdf") or suffixes (like version numbers) that consistently anchor to the end of a string. Similar to the **LEFT()** function, **RIGHT()** is straightforward, requiring only the text string and the count of characters to be extracted from the right side.

#return last 3 characters of string in cell A2
=RIGHT(A2, 3)

The **RIGHT()** function is exceptionally useful for its simplicity when dealing with fixed-length suffixes. However, if the length of the string segment you need to extract varies, you may need to combine **RIGHT()** with functions that calculate length, such as **LEN()**, or functions that locate specific characters, such as **FIND()**, to dynamically determine the correct number of characters to retrieve. For the purpose of simple extraction from the end, however, its syntax is the cleanest.

Method 4: Extracting Substring Before Certain Text (Delimiter-Based)

The three positional functions (LEFT, MID, RIGHT) are limited when the string length or internal structure is inconsistent. For modern versions of Excel (Microsoft 365 and Excel 2021+), the introduction of the **TEXTBEFORE()** function revolutionized delimiter-based extraction. This function allows you to effortlessly pull all content that occurs before a specified piece of text--the delimiter--without needing complex nested functions involving **FIND()** and **LEN()**.

```
#return all text before the string "there" in cell A2
=TEXTBEFORE(A2, "there")
```

The TEXTBEFORE() function simplifies data parsing immensely. Its basic syntax requires the text string and the delimiter. Crucially, **TEXTBEFORE()** offers optional arguments that enhance its utility, such as specifying which instance of the delimiter to use (e.g., the second comma), whether the search should be case-sensitive, and what to return if the delimiter is not found. This robustness makes it the preferred modern approach for extracting the first part of URLs, email usernames, or descriptive text preceding a separator like a hyphen or colon.

Method 5: Extracting Substring After Certain Text (Delimiter-Based)

Mirroring the functionality of **TEXTBEFORE()**, the TEXTAFTER() function handles the retrieval of content that follows a specified delimiter. This is essential for isolating the latter half of a string, such as extracting a domain name following the "@" symbol in an email address or retrieving a specific value that appears after an identifying label.

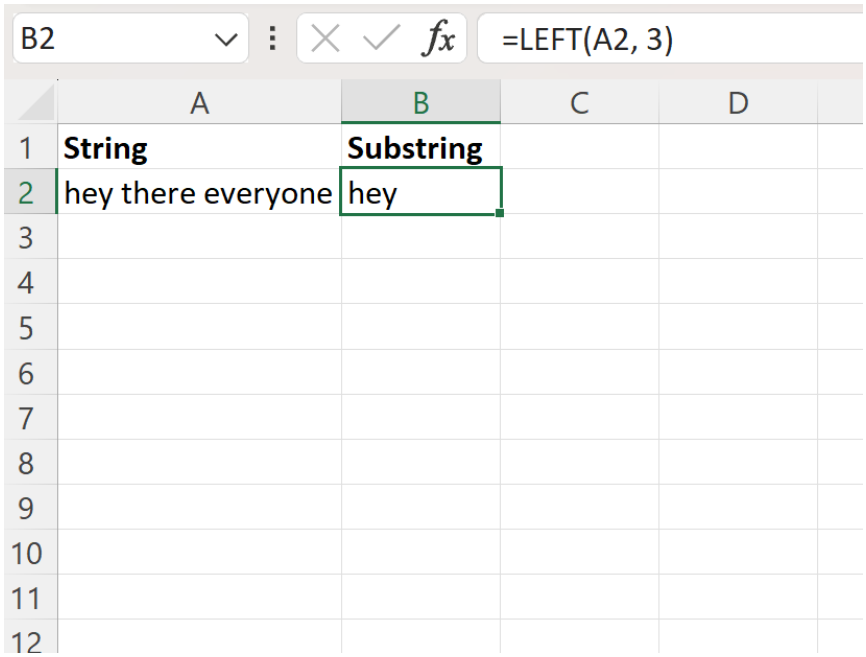
```
#return all text after the string "there" in cell A2
=TEXTAFTER(A2, "there")
```

Like its counterpart, **TEXTAFTER()** is an advanced function that drastically cuts down on the complexity traditionally required to achieve this result. Before its introduction, users had to combine **MID()**, **FIND()**, and **LEN()** to calculate the exact starting position and required length after a delimiter. Now, with **TEXTAFTER()**, the process is streamlined to specifying the text and the delimiter, offering similar optional arguments for instance number, case sensitivity, and handling cases where the delimiter is absent. These two text functions represent a massive improvement in Excel's text parsing capabilities for modern users.

The following practical examples illustrate precisely how to implement each of these five methods within a worksheet environment.

Practical Implementation: Method 1 (LEFT)

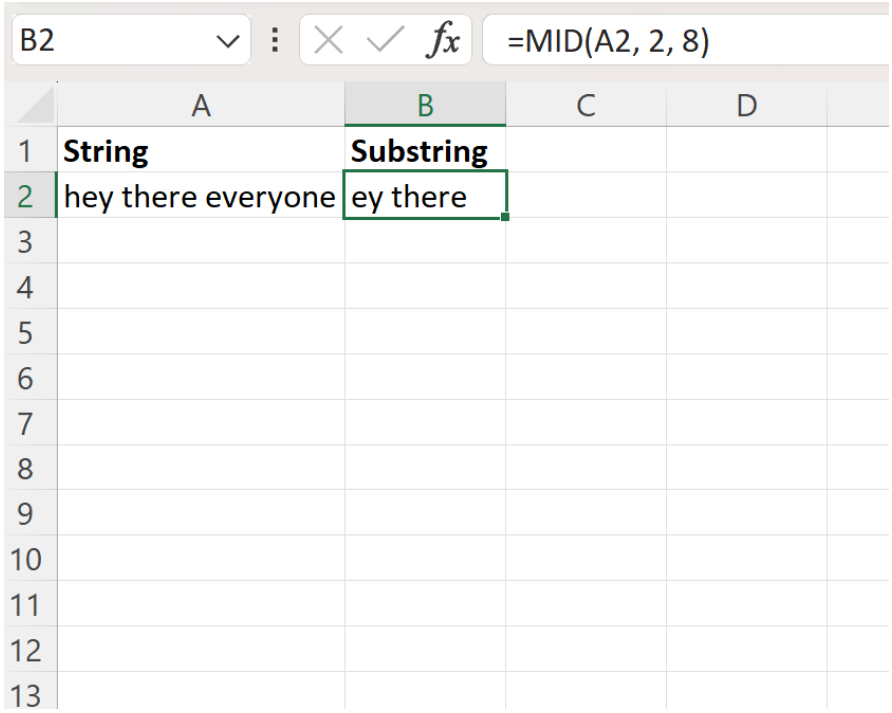
The **LEFT()** function is fundamental for standardizing data where the initial characters hold specific meaning, such as extracting the three-digit currency code "USD" from a string like "USD 125.00". The following screenshot demonstrates using the **LEFT()** function to return the first three characters from the content in cell **A2**, isolating the prefix from the main string.



If the data structure is highly reliable (i.e., the desired segment is always the first N characters), **LEFT()** is the fastest and most efficient choice. However, remember that if the source string itself is shorter than the requested number of characters, **LEFT()** will simply return the entire string without error, which is useful but requires vigilance if data quality is a concern.

Practical Implementation: Method 2 (MID)

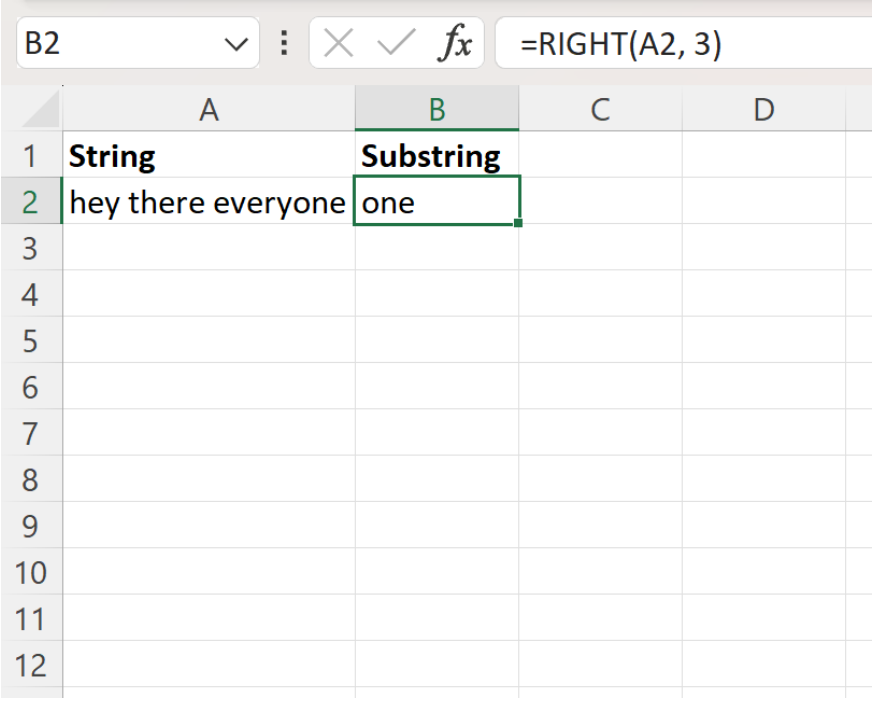
For scenarios involving structured identifiers, such as tracking numbers or employee codes where a key piece of information is located in a fixed middle position, the **MID()** function is essential. The following screenshot illustrates how to use **MID()** to extract eight characters from the text in cell **A2**, specifically starting at the second position.



The careful selection of the **start_num** argument is critical when using **MID()**. If the starting number is calculated dynamically (perhaps using the position returned by **FIND()**), you must account for the length of the delimiter itself to ensure the extraction starts precisely after the delimiter, not on it. This function remains the standard for handling fixed-length, position-dependent data segmentation.

Practical Implementation: Method 3 (RIGHT)

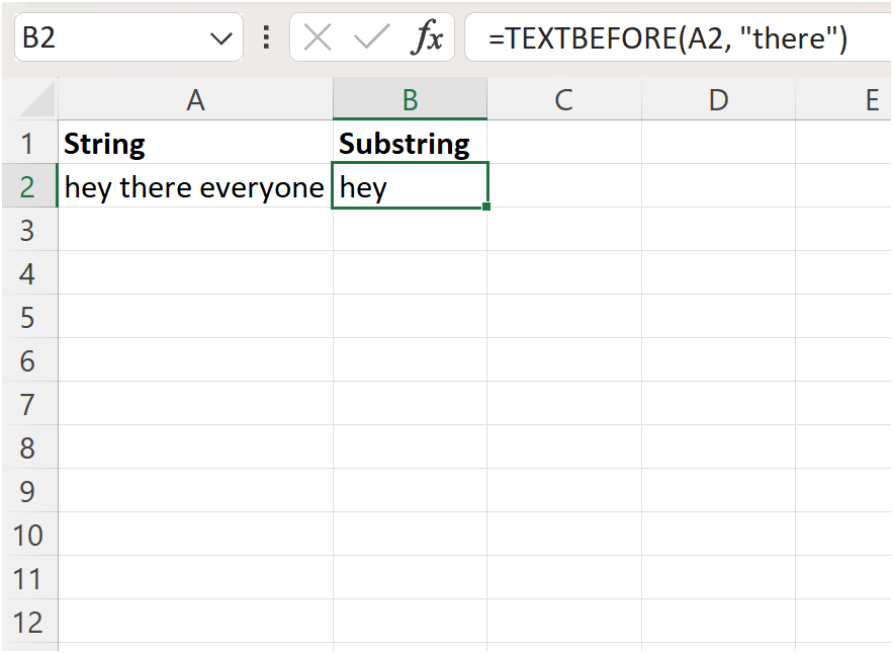
The **RIGHT()** function is most frequently employed when working with file names or other data where the suffix contains crucial information. For example, extracting the last four digits of a serial number or the file extension at the end of a path. The screenshot below shows the application of the **RIGHT()** function to return the last three characters from cell **A2**.



While simple for fixed lengths, using **RIGHT()** often requires integration with other functions when the length is variable. A common complex scenario involves finding the position of the last delimiter (like a slash or period) in a string and subtracting that position from the total length to calculate the exact number of characters needed for the **num_chars** argument. For basic, fixed-length extraction, however, **RIGHT()** provides an elegant solution.

Practical Implementation: Method 4 (TEXTBEFORE)

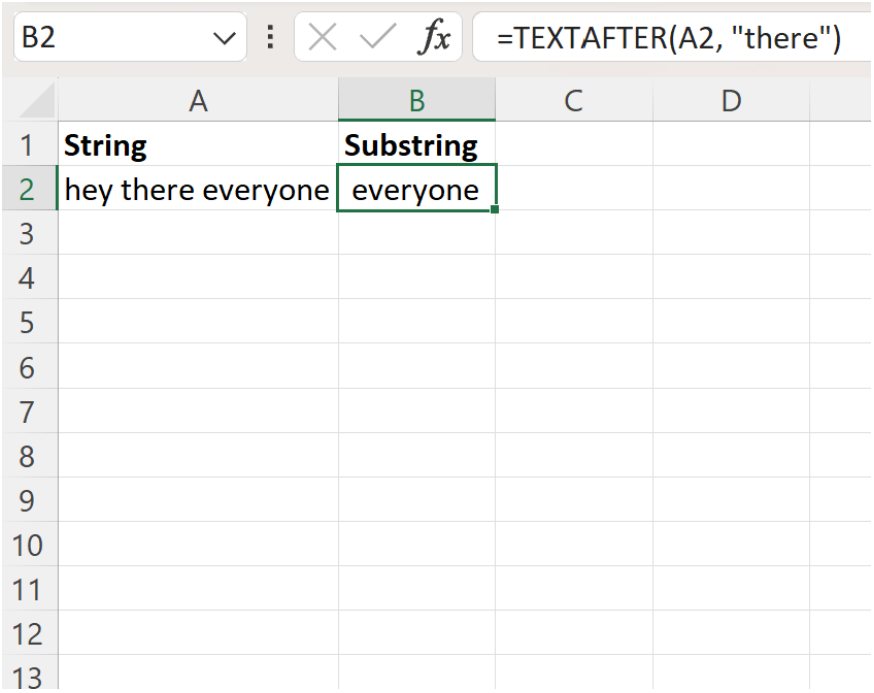
The introduction of **TEXTBEFORE()** significantly simplifies the process of parsing strings where the data is separated by known separators, but the length of the data segments varies widely. Instead of counting characters, we identify the marker. The following screenshot demonstrates using the **TEXTBEFORE()** function to return all of the text that precedes the specific string "there" in cell **A2**.



This function is especially useful for extracting names from full email addresses (using "@" as the delimiter) or retrieving street names from addresses (using the first comma as the delimiter). The optional arguments allow users to search from the beginning or end of the string, handle case sensitivity, and manage situations where the delimiter might appear multiple times, offering unparalleled flexibility in text manipulation compared to older methods.

Practical Implementation: Method 5 (TEXTAFTER)

When data is structured such that the desired value always follows a label or separator, **TEXTAFTER()** provides the solution. It eliminates the tedious calculation of starting positions required by **MID()**. The example below shows how to use the **TEXTAFTER()** function to efficiently return all text that follows the string "there" in cell **A2**.



A typical application of **TEXTAFTER()** is extracting the domain name from an email address or retrieving the value associated with a key in a configuration string (e.g., retrieving "100" from "Max_Value:100"). Just like **TEXTBEFORE()**, the function's ability to specify which instance of the delimiter to use (e.g., extracting the second token after the third space) makes it incredibly powerful for complex data parsing tasks, providing modern Excel users with tools that were previously only available through advanced programming languages.

Additional Resources for Advanced Text Manipulation

Understanding how to extract substrings is the foundation of many data cleaning and analysis operations in Excel. To further enhance your skills, consider exploring tutorials on related text functions that complement these extraction methods, such as:

CONCATENATE or TEXTJOIN: Useful for combining extracted substrings back into a new, cleaned format.
FIND and SEARCH: Essential for determining the precise starting position for **MID()** when the location is not fixed.
TRIM: Necessary for removing leading or trailing spaces that often result from text extraction or data import processes.

The following tutorials explain how to perform other common operations in Excel: