How to Easily Extract Substrings in Google Sheets Using LEFT, RIGHT, MID, and FIND

Authored by

stats writer

December 3, 2025

Mastering string manipulation is fundamental for effective data analysis in tools like Google Sheets. One of the most common requirements when cleaning or processing datasets is the ability to extract a substring--a specific segment of characters--from a larger text string. Whether you are isolating product codes, separating names, or pulling domain extensions, Google Sheets provides a powerful suite of functions tailored precisely for this purpose.

The core functions used for this task include LEFT, RIGHT, and MID, which handle extraction based on position and length. When the position is not fixed, we rely on locator functions like FIND or SEARCH to pinpoint the exact location of a specific character or delimiter. By combining these functions, users can create robust, dynamic formulas capable of handling complex data structures efficiently.

This guide delves into the syntax and practical application of these crucial functions, illustrating five principal methods for extracting the desired substring. Understanding these combinations is essential for anyone aiming to move beyond basic spreadsheet tasks and achieve sophisticated data processing capabilities.

To quickly reference the primary extraction formulas available in Google Sheets, review the methods outlined below. These examples use simple, fixed length or position arguments:

The Foundational Substring Extraction Methods

Method 1: Return Substring from Beginning of String

The LEFT function is designed to pull a specified number of characters starting from the very first character on the left side of the string. It is often used for extracting prefixes, initial codes, or shortened identifiers.

```
#return first 4 characters of string in cell A1
=LEFT(A1, 4)
```

Method 2: Return Substring from Middle of String

When the desired substring resides within the middle of the text, the MID function is required. This function needs three arguments: the cell containing the text, the starting position index, and the total length of the segment you wish to extract.

```
#return 4 characters of string in cell A1 starting at position 2
=MID(A1, 2, 4)
```

Method 3: Return Substring from End of String

To extract characters from the right end of a string, use the RIGHT function. This is particularly useful for separating suffixes, such as file extensions or area codes, where the length from the end is consistent.

#return last 4 characters of string in cell A1
=RIGHT(A1, 4)

Combining Functions for Dynamic Extraction

While the basic functions (LEFT, RIGHT, MID) are excellent for fixed-length extractions, real-world data often requires dynamic solutions. Dynamic extraction involves using a locator function like SEARCH or FIND to locate a delimiter (e.g., a comma, a space, or a specific word) and then instructing the extraction function (LEFT, RIGHT, or MID) to use that position as a calculation point. This makes the formula resilient, regardless of the length of the string before or after the delimiter.

Method 4: Return Substring Before Certain Text

This method combines LEFT with SEARCH. The SEARCH function identifies the position of the target text (the delimiter). Since we want the text **before** the delimiter, we subtract 1 from the position found by SEARCH, which gives the exact number of characters needed for the LEFT function.

#return all text before the string "there" in cell A1
=LEFT(A1, SEARCH("there", A1)-1)

Method 5: Return Substring After Certain Text

Extracting the text that follows a delimiter is slightly more complex as it requires calculating the total length of the string, the starting position of the delimiter, and the length of the delimiter itself. This generally involves combining RIGHT, SEARCH, and the **LEN** function (which finds the total length of the string). Note that the original formula provided below is a simplified version; for complex dynamic extraction after text, the formula is usually: `=RIGHT(A1, LEN(A1) - SEARCH("there", A1) - (LEN("there")-1))`. The provided example below utilizes a less common structure that implicitly assumes a fixed length calculation from the right based on the search position.

#return all text after the string "there" in cell A1
=RIGHT(A1, SEARCH("there", A1)-1)

Detailed Analysis of Extraction Functions

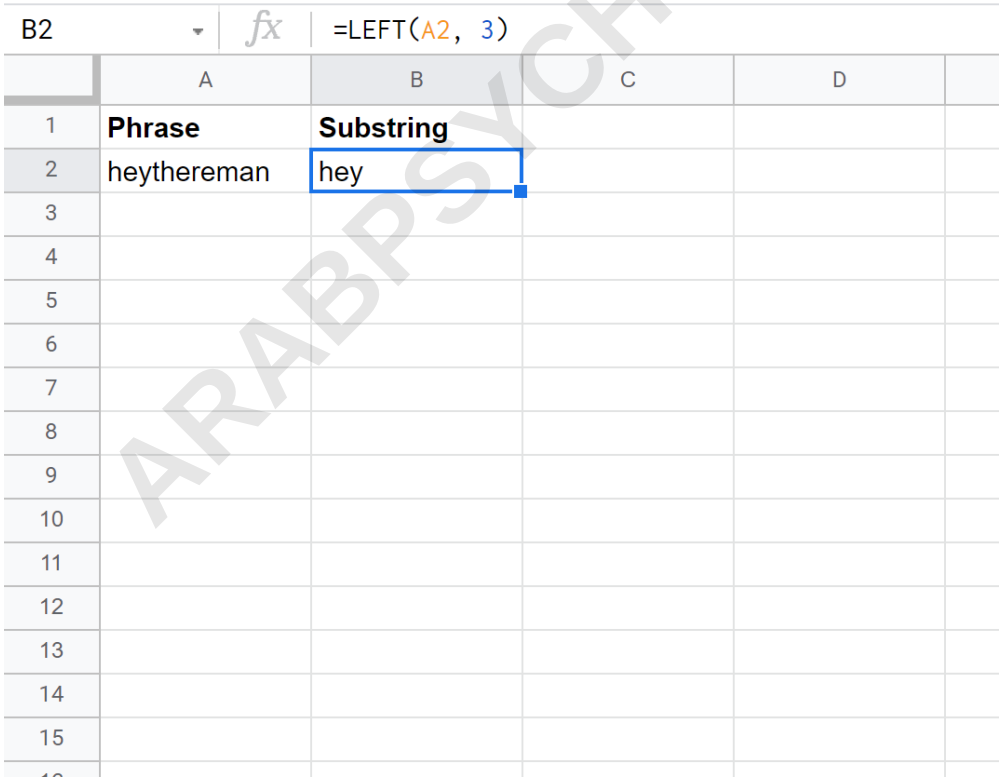
The following sections provide a practical demonstration of how each of the extraction methods is implemented within a Google Sheets environment. We will use various data scenarios to illustrate their effectiveness.

Understanding the LEFT Function Syntax and Use Cases

The LEFT function is the simplest way to retrieve characters from the start of a cell's content. Its syntax is straightforward: `=LEFT(string, )`. The `string` is the cell reference containing the text (e.g., A2), and the optional `number_of_chars` specifies how many characters to return. If `number_of_chars` is omitted, it defaults to 1.

For example, if you have employee IDs structured as "2024-ABC-1234," and you only need the year prefix "2024," you would use LEFT to grab the first four characters. This function is vital for standardizing data fields where introductory identifiers are uniform in length across the dataset.

The following screenshot shows how to use the **LEFT()** function to return the first three characters from cell A2:



If the specified number of characters exceeds the total length of the string, the LEFT function will simply return the entire string without error, ensuring data integrity even if the source data varies

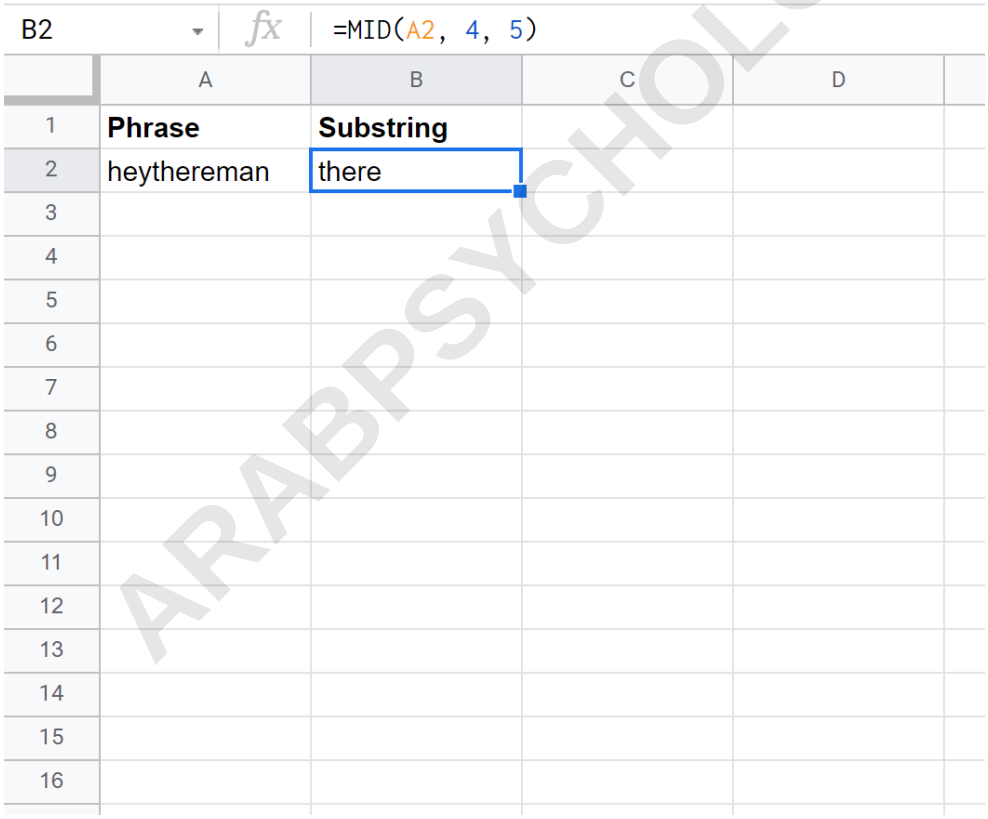
slightly in length.

Mastering the MID Function for Central Data Extraction

The MID function is highly versatile as it allows extraction from any point within a string, provided you know the exact starting position and the desired length. The syntax is `=MID(string, start_position, number_of_chars)`. Unlike LEFT and RIGHT, which only require length, MID mandates a precise starting index.

Consider a situation where product codes like "X-5532-G" always have a five-digit numerical sequence starting at the third position. Using `=MID(A2, 3, 5)` would correctly isolate the "5532" segment. It is crucial to remember that Google Sheets counts characters sequentially starting from 1.

The following screenshot shows how to use the **MID()** function to return the five characters in the middle of cell A2, starting at position 4:



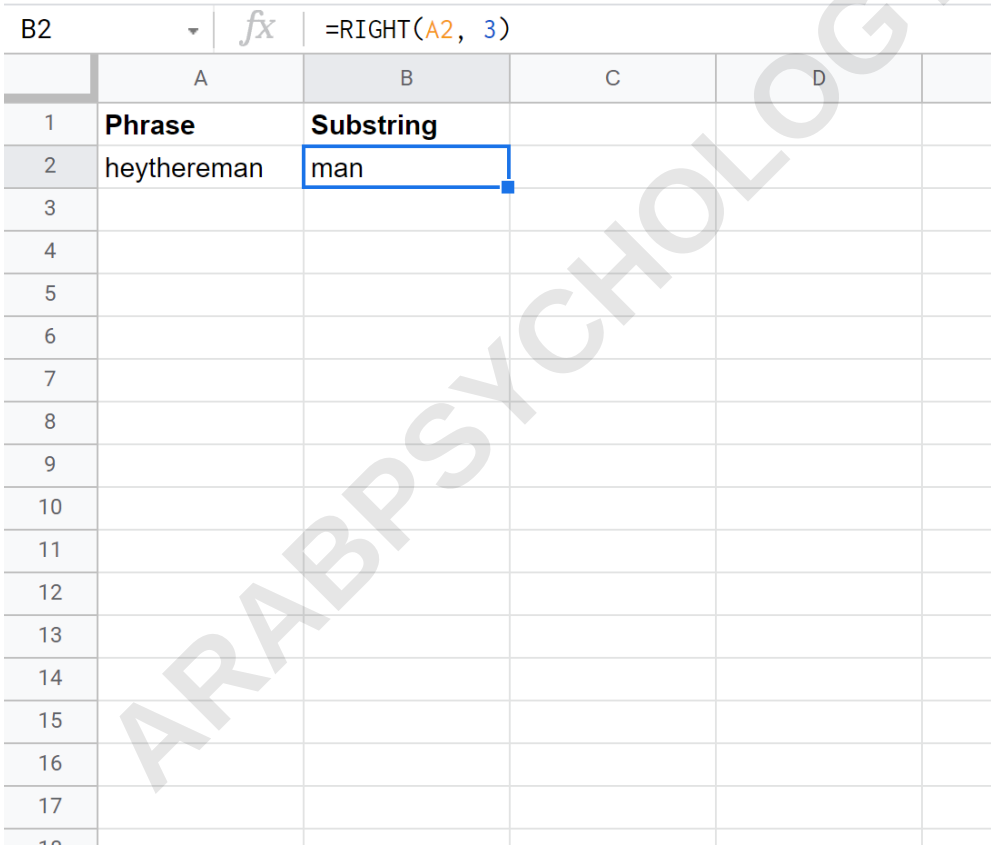
If the calculated start position is beyond the length of the string, MID will return an empty string. If the combined start position and length exceed the string boundary, it will return all characters remaining from the start position until the end of the string.

Implementing the RIGHT Function for Trailing Data

The RIGHT function mirrors the functionality of LEFT but operates from the opposite end of the string. Its syntax is `=RIGHT(string, )`. This function is indispensable when dealing with data where the identifying information is consistently located at the tail end, such as timestamps or extensions.

A common application is extracting the four-digit year from a date string formatted as "MM-DD-YYYY," or isolating the file type (".pdf," ".docx") from a full file name. By setting the `number_of_chars` to 4, you efficiently extract the desired substring.

The following screenshot shows how to use the **RIGHT()** function to return the last three characters from cell A2:



Like LEFT, if the requested number of characters exceeds the total length, RIGHT will return the entire source string, preventing calculation errors in your sheet.

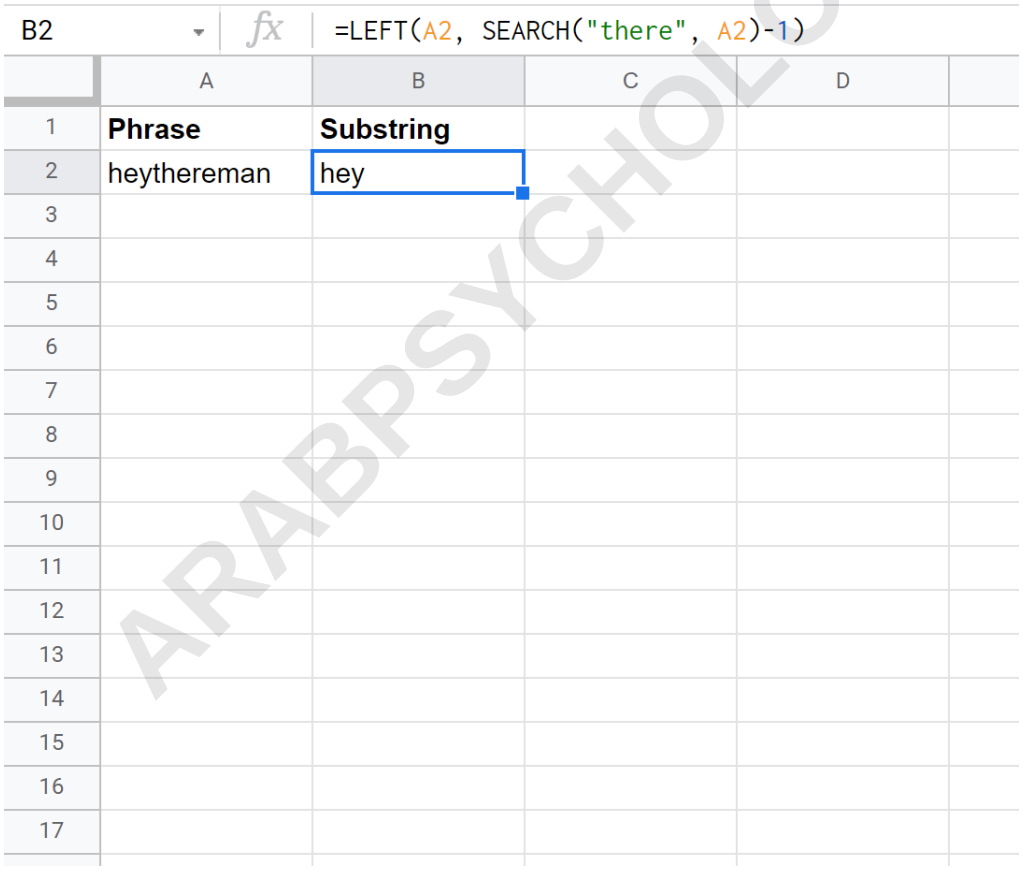
Dynamic Extraction with LEFT and SEARCH/FIND (Substring Before

Delimiter)

When the data is inconsistent--for example, when names or descriptions vary in length before a common delimiter like a space, hyphen, or comma--we must use dynamic functions. The SEARCH function returns the starting position of a specified substring within a text string. The key difference between SEARCH and FIND is that SEARCH is case-insensitive, making it more robust for general text parsing, while FIND is case-sensitive.

To extract the text **before** the delimiter, we use LEFT, specifying the length as the delimiter's position minus one. The formula structure is `=LEFT(A2, SEARCH("delimiter", A2) - 1)`. Subtracting 1 ensures that the delimiter itself is not included in the output, yielding a clean separation of the text segments.

The following screenshot shows how to use the **LEFT()** and **SEARCH()** functions to return all of the text that comes before the string "there" in cell A2:



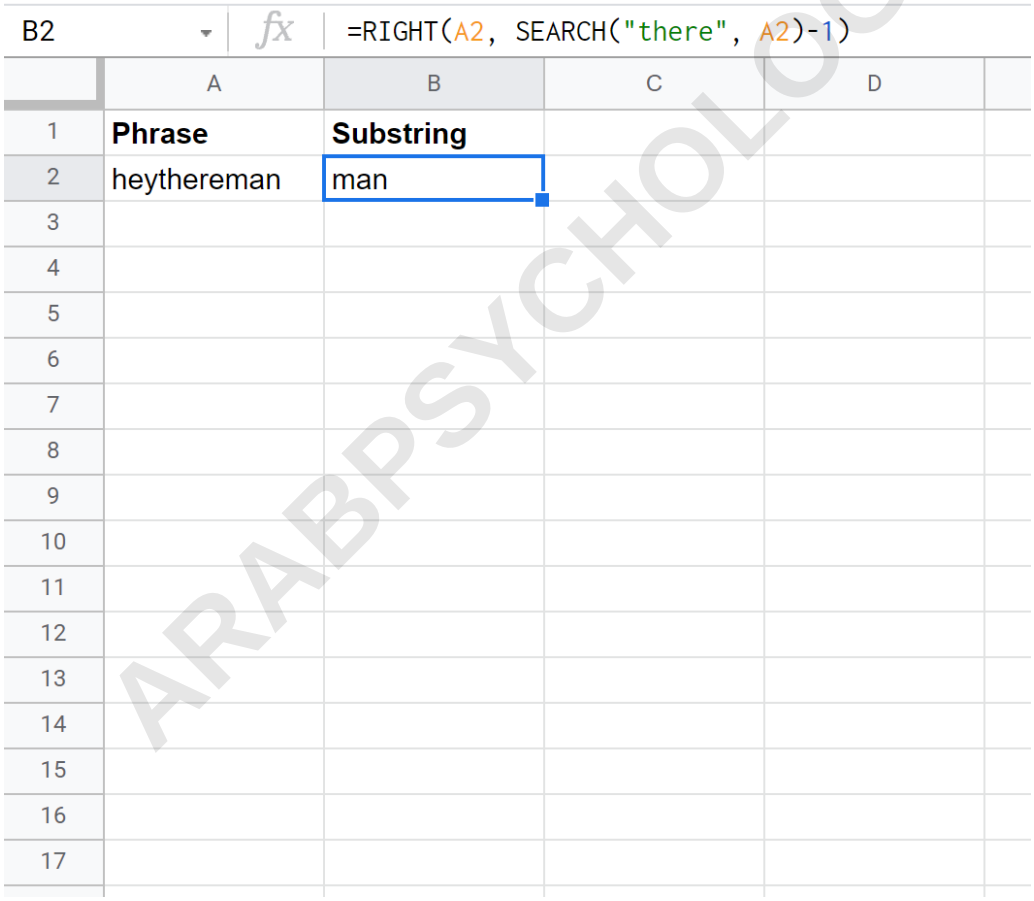
If the SEARCH function fails to find the target delimiter, it returns a `#VALUE!` error, which is a common indicator that the structure of the source data is not as expected. This error can often be managed using **IFERROR**.

Dynamic Extraction with RIGHT and SEARCH/LEN (Substring After Delimiter)

To extract the substring **after** a known delimiter, we must use the RIGHT function, but calculate the length of the trailing text dynamically. This calculation requires finding the total length of the string using `LEN(A2)`, subtracting the position of the delimiter found by SEARCH, and then adjusting for the length of the delimiter itself.

The general robust formula is `=RIGHT(A2, LEN(A2) - SEARCH("delimiter", A2) - LEN("delimiter") + 1)`. This complex calculation correctly determines the exact number of characters remaining after the delimiter, providing the argument necessary for the RIGHT function to operate correctly.

The following screenshot shows how to use the **RIGHT()** and **SEARCH()** functions to return all of the text that comes after the string "there" in cell A2:



This dynamic method is crucial when dealing with varying data, such as extracting the city name when addresses are stored in a single column, provided the state name or ZIP code serves as a consistent delimiter for the preceding information.

Advanced Dynamic Extraction: Substring Between Two Delimiters

One of the most powerful uses of these functions is extracting content located between two markers, such as the text inside parentheses or between two hyphens. This requires the combined application of MID and two separate SEARCH functions.

The logic is as follows: first, find the start position of the first delimiter (D1). Second, find the start position of the second delimiter (D2). The start position for MID is D1 plus the length of D1. The length of the extraction is D2 minus D1, minus the length of D1. This mathematically isolates the characters residing exactly between the two defined markers.

For example, to extract content between hyphens in "Prefix-CODE-Suffix," you would use: `=MID(A2, SEARCH("-", A2) + 1, SEARCH("-", A2, SEARCH("-", A2) + 1) - SEARCH("-", A2) - 1)`. Note the use of the third argument in the second SEARCH to ensure it starts looking **after** the first delimiter, preventing it from immediately finding the first hyphen again. This technique is indispensable for parsing complex structured data.

Choosing Between FIND and SEARCH

When implementing dynamic formulas (Methods 4 and 5), selecting the correct locator function is paramount:

FIND: This function is **case-sensitive**. If you are searching for "ID" but the text contains "id," FIND will fail to locate it. Use FIND when capitalization consistency is mandatory, such as searching for specific acronyms or database keys.

SEARCH: This function is **case-insensitive**. It will find "ID" whether it appears as "id," "Id," or "ID." SEARCH also supports the use of wildcard characters (like `?` and `*`), offering greater flexibility when dealing with inconsistent data patterns. It is generally the preferred choice for high-volume, general text cleanup.

Both functions return the position of the starting character of the found substring, which is critical for dictating the length or starting position arguments in LEFT, RIGHT, or MID.

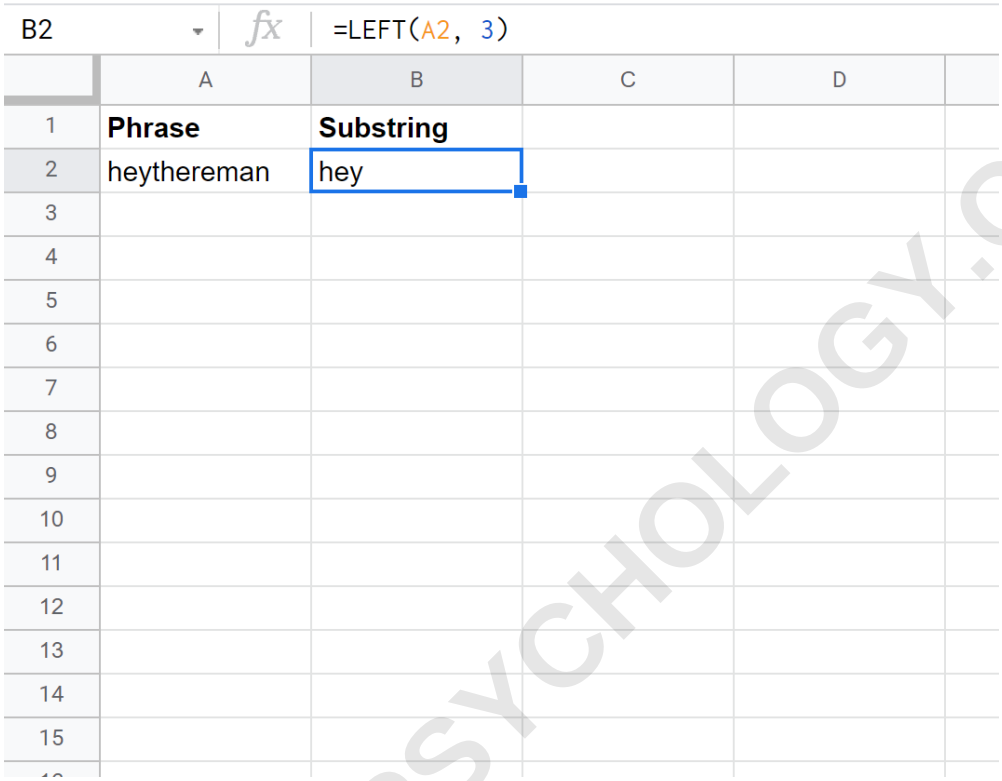
Summary of Substring Extraction Functions

Effective data manipulation in Google Sheets relies heavily on understanding how to combine positional and locator functions. Whether you need a simple, fixed-length extraction using LEFT, RIGHT, or MID, or a complex dynamic split utilizing SEARCH and **LEN**, these tools provide the necessary flexibility to transform raw data into actionable information. Consistent practice with these nested formulas will solidify your expertise in spreadsheet data processing.

The following examples show how to use each of these methods in practice.

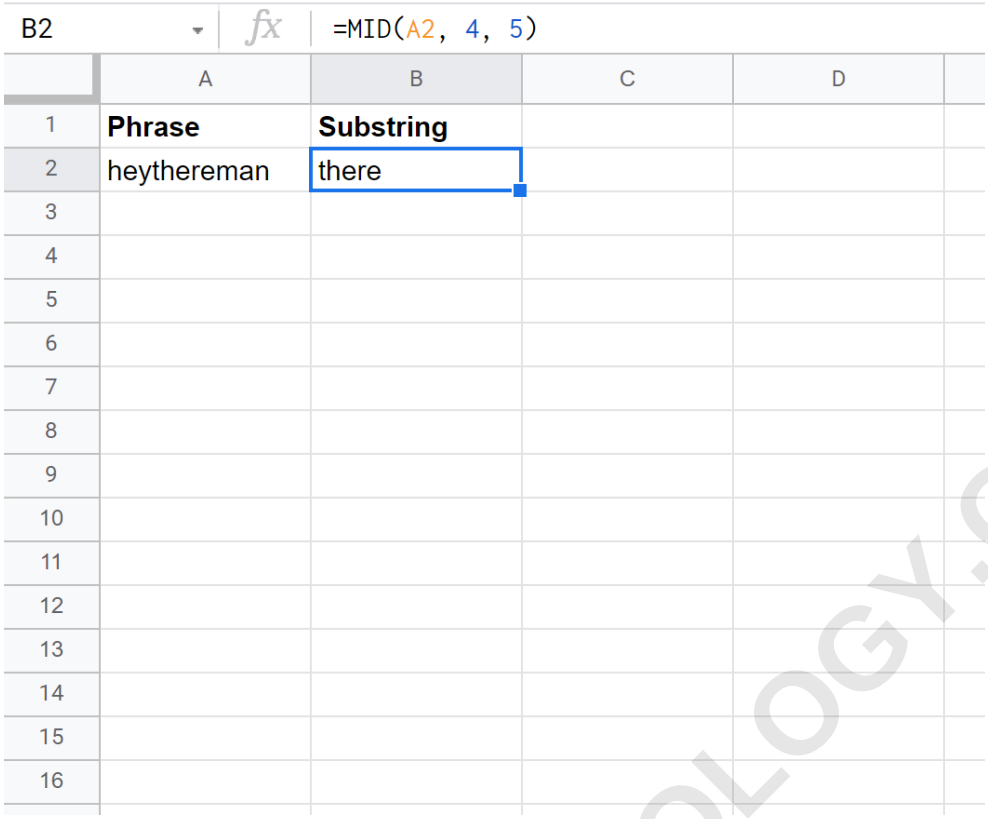
Practical Example 1: Fixed Length Extraction from the Start

The following screenshot shows how to use the **LEFT()** function to return the first three characters from cell A2:



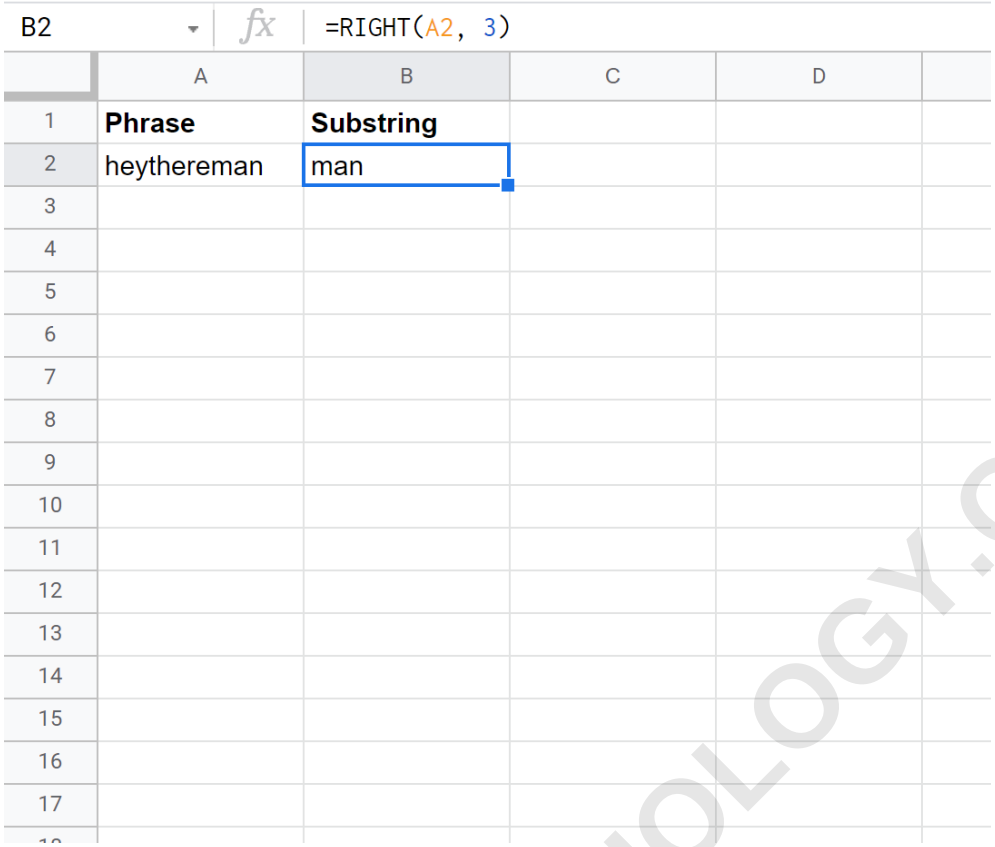
Practical Example 2: Fixed Length Extraction from the Middle

The following screenshot shows how to use the **MID()** function to return the five characters in the middle of cell A2, starting at position 4:



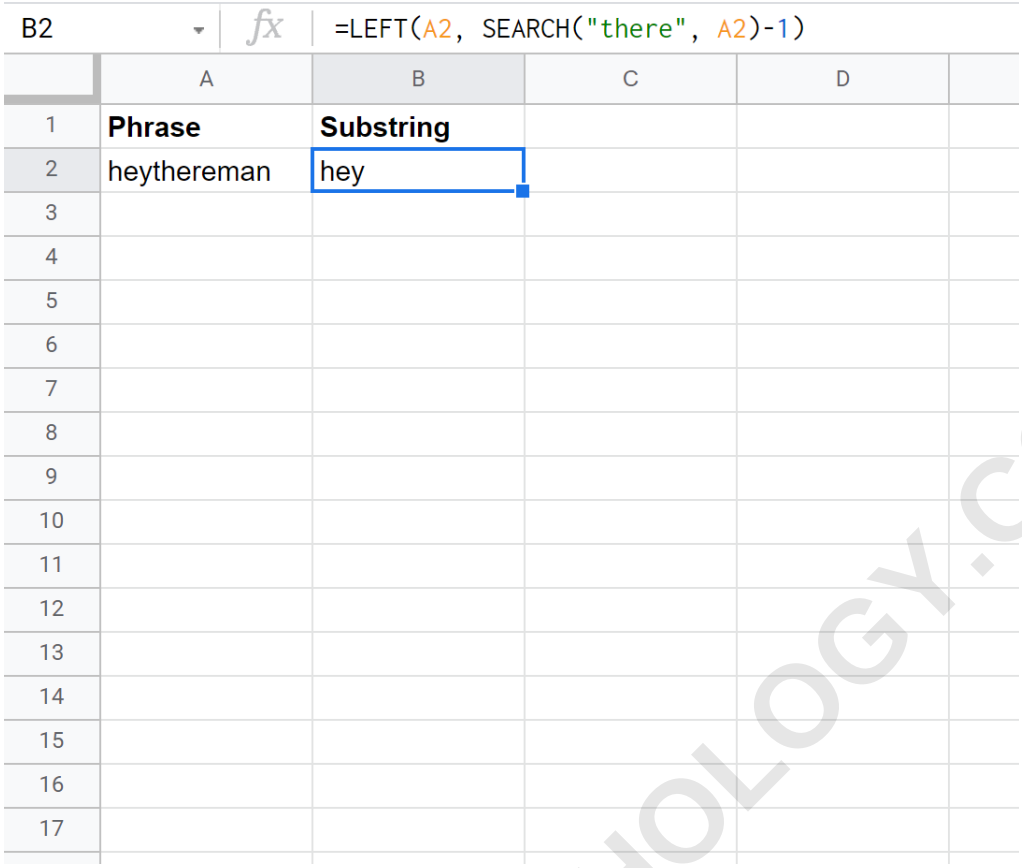
Practical Example 3: Fixed Length Extraction from the End

The following screenshot shows how to use the **RIGHT()** function to return the last three characters from cell A2:



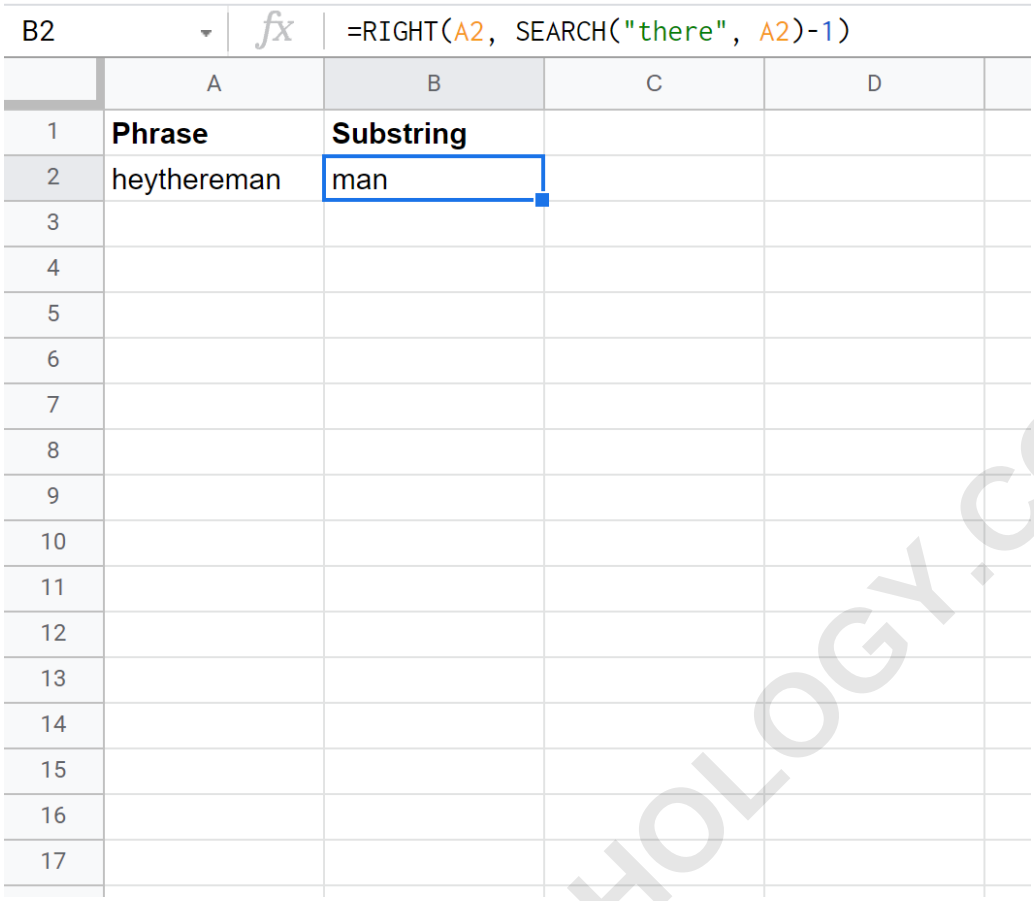
Practical Example 4: Dynamic Extraction Before a Delimiter

The following screenshot shows how to use the **LEFT()** and **SEARCH()** functions to return all of the text that comes before the string "there" in cell A2:



Practical Example 5: Dynamic Extraction After a Delimiter

The following screenshot shows how to use the **RIGHT()** and **SEARCH()** functions to return all of the text that comes after the string "there" in cell A2:



Conclusion: Empowering Your Data Analysis

The ability to accurately and dynamically extract a substring is an essential skill for anyone working extensively with textual data in spreadsheets. By leveraging the power of functions like LEFT, RIGHT, MID, and their crucial locator counterparts, FIND and SEARCH, you can efficiently clean, restructure, and analyze complex datasets. These methods provide the foundation for powerful automated data workflows, reducing manual error and improving the overall quality of your spreadsheet operations.