Google Sheets: Extract Text Between Two Characters

Authored by

stats writer

November 17, 2025

Google Sheets stands as an incredibly robust and essential platform for modern data analysis and complex manipulation tasks. While many users are familiar with basic arithmetic and simple filtering, the true power of this spreadsheet application emerges when dealing with complex text strings--a common occurrence when importing or consolidating information from disparate sources. Efficiently parsing and isolating specific segments of data is often the key bottleneck in preparing information for subsequent calculations or visualizations. Whether you are dealing with log files, database exports, or raw output from programming scripts, the necessity of accurately isolating text located between two defining markers is paramount for data hygiene and structure.

One of the most valuable, yet often underutilized, features within the Sheets ecosystem is the ability to extract text based on defined boundaries or delimiters. This capability allows users to quickly isolate specific data points embedded within longer strings, bypassing the need for complex, nested functions like **MID**, **FIND**, and **LEN**. When working with massive datasets, such as those derived from web scraped data, the raw input often contains extraneous metadata, tags, or surrounding text that obscures the critical information. The ability to surgically extract only the relevant segment significantly streamlines the filtering process, saving considerable time and minimizing the risk of manual error during data preparation.

The technique we will explore utilizes **Regular Expressions** (often shortened to Regex), providing a highly flexible and pattern-based approach to text extraction. This method is far superior to traditional string functions when the delimiters themselves might be complex, or when the position of the desired text is variable within the cell. Mastering this technique transforms Google Sheets from a basic calculation tool into a sophisticated text-processing environment, enabling rapid formatting, organization, and normalization of complex text data required for thorough analysis.

You can use the **REGEXEXTRACT** function in Google Sheets to extract all text between two specific characters in a cell.

You can use the following syntax to do so:

=REGEXEXTRACT(A2,"this(.*)that")

This particular formula extracts all of the text between the characters "this" and "that" in cell **A2**. Note that the desired extracted text is contained within the parentheses `(.*)`, which creates the required capture group for the **REGEXEXTRACT** function.

The following examples show several common ways to extract text between two characters in practice.

The Power of the REGEXEXTRACT Function

To achieve precision extraction of text segments in Google Sheets, we rely almost exclusively on the built-in function, **REGEXEXTRACT**. This powerful function is specifically designed to work with Regular Expressions, allowing you to define a specific text pattern and pull out the segment that matches that pattern. Unlike simpler functions that only look for fixed positions or exact substrings, **REGEXEXTRACT** provides dynamic pattern matching, making it the ideal choice for finding text located between two arbitrary characters or strings. The function syntax is straightforward, requiring only two core arguments: the text you wish to search, and the regular expression pattern itself.

The general syntax for implementing this extraction method is concise and highly effective. You specify the target cell containing the data, followed by the specific Regular Expression that encapsulates the desired text segment. Crucially, the text you want to extract must be captured within parentheses within the Regex pattern. This capture group signals to **REGEXEXTRACT** which portion of the overall match should be returned as the result. If the pattern successfully matches the string, the content of the capture group is returned; otherwise, an error or blank cell will result, indicating no match was found.

The core formula introduced previously--`=REGEXEXTRACT(A2, "this(.*)that")`--perfectly illustrates the marriage of the function with the Regex pattern. The pattern `(.*)` uses the dot operator (`.`), which matches any single character (except newline), and the asterisk quantifier (`*`), meaning "zero or more occurrences" of the preceding character. When combined, this sequence matches and captures any sequence of characters between the defined boundaries ("this" and "that").

Understanding Regular Expressions for Extraction

The effectiveness of the **REGEXEXTRACT** function hinges entirely on understanding the basic components of the Regular Expressions pattern used. The central element in our extraction pattern, `(.*)`, is known as a capture group. The characters used within this group--the dot (`.`), the asterisk (`*`), and the surrounding parentheses--work in concert to define the content we wish to isolate. Specifically, the dot represents a wildcard, matching any character except a line break. The asterisk acts as a quantifier, instructing the engine to look for zero or more occurrences of whatever precedes it. Therefore, `(.*)` instructs the Regex engine to capture any sequence of characters found between the defined start and end strings.

It is important to recognize that by default, most regular expression engines, including the one used by Google Sheets, operate in a "greedy" mode. This means that the `.*` pattern will attempt to match the longest possible string that still allows the overall expression to succeed. If your cell

contains multiple instances of the delimiter strings, a greedy match will extend all the way from the first starting delimiter encountered to the absolute last ending delimiter in the text string. For most extraction tasks, particularly when text spans multiple sentences or phrases, this greedy behavior is generally desired, ensuring that all intervening text is captured.

However, if you are certain that you only want to extract the shortest possible text segment between delimiters (i.e., if you have repeating patterns like "A(1)B A(2)B" and only want "(1)"), you would need to modify the quantifier to be "non-greedy" or "lazy." Achieving a non-greedy match typically involves adding a question mark after the asterisk, resulting in the pattern `(.*?)`. Utilizing this lazy quantifier forces the Regex engine to stop matching as soon as it encounters the first occurrence of the closing delimiter. Understanding the difference between greedy and non-greedy matching is crucial for advanced text data analysis and ensuring accurate results when dealing with complex or repetitive data structures.

Example 1: Extracting Text Between Specific Text Strings

The most common application for the **REGEXEXTRACT** function is isolating content defined by clear, unique word or phrase delimiters. Imagine you have a list of training logs where the distance covered is embedded within descriptive sentences. We want to extract only the distance, which is always located between the word "ran" and the word "miles." This scenario requires a specific pattern that acknowledges both the fixed starting and ending text segments while capturing everything in between, including spaces and numerals.

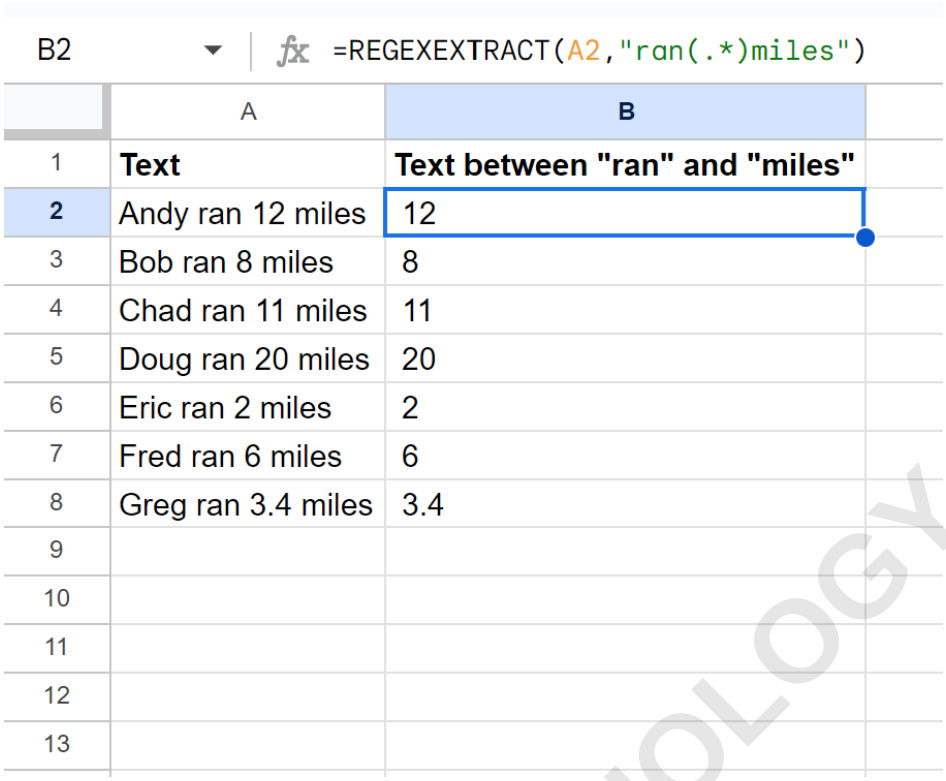
To execute this extraction, we must construct the Regex pattern to explicitly include the starting string ("ran") and the ending string ("miles") surrounding our capture group `(.*)`. By enclosing the target content within this capture group, we instruct the function to discard the delimiters themselves and return only the variable text they enclose. We will apply this formula to the first entry in our dataset, located in cell **A2**, to establish the pattern for the entire column.

We implement the following formula into cell **B2** to perform the targeted extraction. Note that the Regex pattern is case-sensitive, so careful attention to capitalization in the delimiters is necessary to ensure an accurate match. Once the formula is entered, we can efficiently apply it to the remainder of the dataset using the fill handle, propagating the logic down Column B to process all corresponding entries in Column A automatically.

We can type the following formula into cell **B2** to extract the text in cell **A2** between the strings "ran" and "miles":

=REGEXEXTRACT(A2,"ran(.*)miles")

We can then click and drag this formula down to each remaining cell in column B:



	A	B
1	**Text**	**Text between "ran" and "miles"**
2	Andy ran 12 miles	12
3	Bob ran 8 miles	8
4	Chad ran 11 miles	11
5	Doug ran 20 miles	20
6	Eric ran 2 miles	2
7	Fred ran 6 miles	6
8	Greg ran 3.4 miles	3.4
9		
10		
11		
12		
13		

Column B contains the text between the strings "ran" and "miles" for each corresponding cell in column A. This outcome clearly validates the formula's effectiveness in isolating the dynamic data (the distance numbers) from the static descriptive text surrounding it, confirming the utility of Regular Expressions in data cleaning tasks.

Example 2: Extracting Text Between Parentheses

While extracting text between standard text strings is straightforward, text extraction becomes slightly more complex when the delimiters themselves are special characters or symbols that hold reserved meaning within Regular Expressions. Parentheses are one such example. In Regex, the primary purpose of parentheses is to define a "capture group," as we saw with `(.*)`. If we want to treat the literal parenthesis characters `(` and `)` as simple delimiters that define the bounds of the text we wish to extract, we must signal this intent to the Regex engine.

To use a reserved character literally, we must "escape" it using a backslash (``). When the Regex engine encounters a backslash followed by a reserved character, it interprets that character as its literal form rather than its functional command. Therefore, to search for the opening parenthesis character, we use `(`, and for the closing parenthesis, we use `)`. If we simply used `((.*))`, the

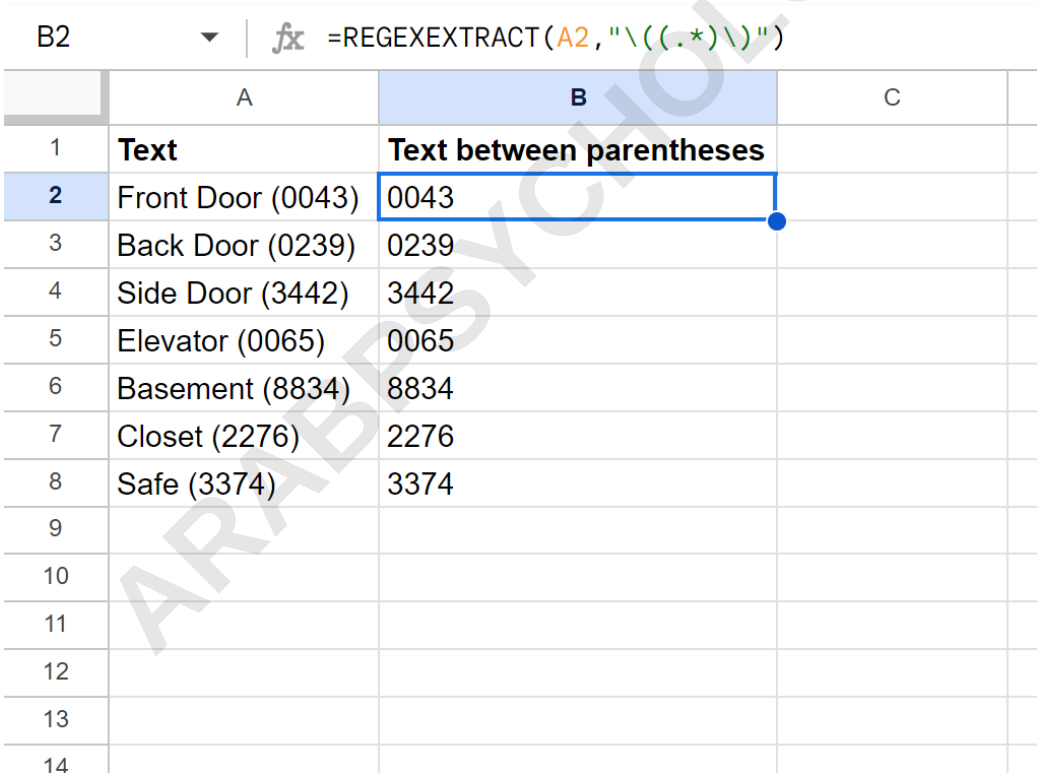
expression would be ambiguous to the engine, potentially leading to errors or unexpected results because it would interpret the outer parentheses as delimiters and the inner ones as capture groups.

For our goal--extracting all content located between actual parentheses in cell **A2**--the pattern must include escaped parentheses acting as the delimiters, with the desired content capture group `(.*)` placed between them. The full pattern required is `((.*))`. The outer backslashes ensure the parentheses are treated as literal boundaries, while the inner `(.*)` remains the capture mechanism for the content between those boundaries.

We can type the following formula into cell **B2** to extract the text in cell **A2** between the parentheses:

=REGEXEXTRACT(A2,"((.*))")

We can then click and drag this formula down to each remaining cell in column B:



Column B contains the text between the parentheses for each corresponding cell in column A. This example highlights a critical principle in advanced text manipulation: when working with functions like **REGEXEXTRACT**, reserved symbols must be correctly escaped to be interpreted as literal characters rather than functional commands.

Example 3: Mastering Escape Characters (Extracting Text Between Asterisks)

Another frequent challenge in text processing involves delimiters that are also powerful quantifiers in Regular Expressions, such as the asterisk (`*`). As established earlier, the asterisk means "zero or more occurrences" of the preceding element. If we wish to extract text bounded by literal asterisks--a common formatting convention for highlighting text, like `*important*`--we face a similar escaping requirement as with parentheses. This challenge is common when processing data from sources like web scraped data or markdown formats.

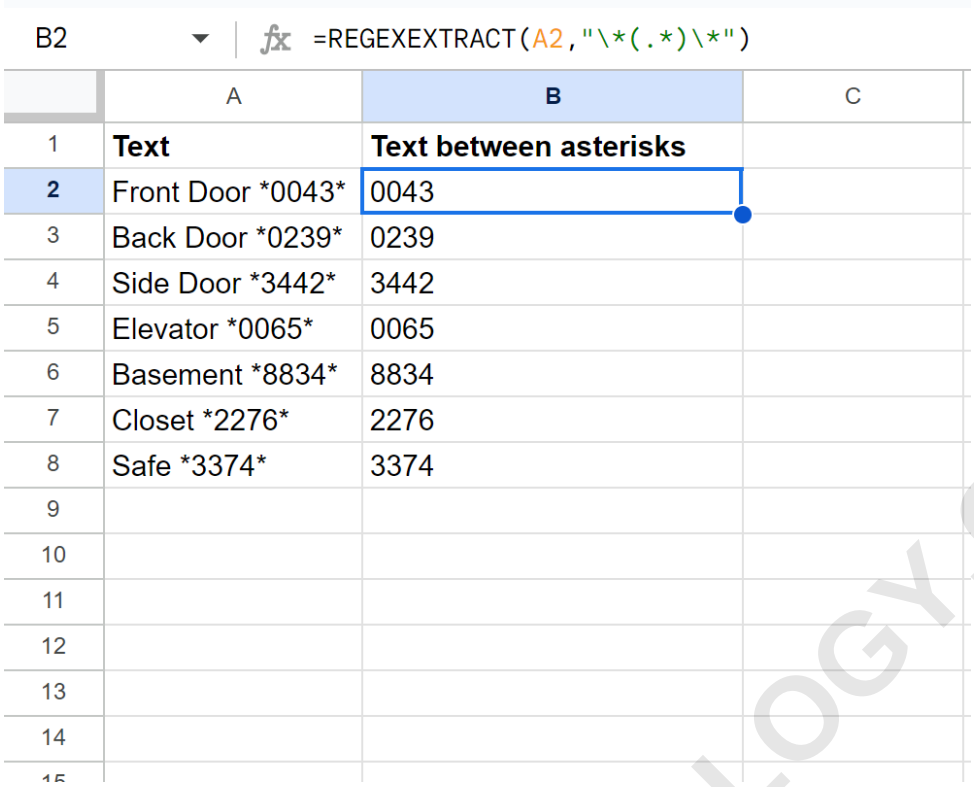
Since the asterisk is a reserved metacharacter, it must be escaped with a backslash. So, if we were writing the Regex pattern in a pure programming environment, we would use `*(.*)*`. We apply this pattern directly within the Google Sheets formula, understanding that Sheets is capable of passing the single backslash correctly to the Regex engine in this context, treating the asterisks as literal boundaries.

The resulting pattern to extract text between two asterisks in cell **A2** becomes `"\*(.*)\*"` (or sometimes `"*(.*)*"` depending on the environment interpretation, but for Sheets, the latter is usually sufficient as shown below). This method ensures that the string passed to the function contains the necessary backslashes required for the Regex engine to interpret the asterisks as delimiters, not quantifiers.

We can type the following formula into cell **B2** to extract the text in cell **A2** between the asterisks:

=REGEXEXTRACT(A2,"*(.*)*")

We can then click and drag this formula down to each remaining cell in column B:



	A	B	C
1	**Text**	**Text between asterisks**	
2	Front Door *0043*	0043	
3	Back Door *0239*	0239	
4	Side Door *3442*	3442	
5	Elevator *0065*	0065	
6	Basement *8834*	8834	
7	Closet *2276*	2276	
8	Safe *3374*	3374	
9			
10			
11			
12			
13			
14			

Column B contains the text between the asterisks for each corresponding cell in column A.

Note: We had to use a slash as an escape character before the asterisks so that the **REGEXTRACT** function recognized these characters as asterisks.

Conclusion and Further Text Functions

The ability to leverage **REGEXEXTRACT** to isolate text between two characters or strings fundamentally elevates the level of data preparation possible within Google Sheets. By utilizing the powerful and flexible syntax of Regular Expressions, specifically the `(.*)` capture group, users can quickly transform messy, unstructured text into clean, organized data points suitable for rigorous analysis. We have demonstrated that whether dealing with simple text delimiters or complex reserved special characters, the principle remains the same: define the start and end boundaries, and capture everything in the middle.

A key takeaway from these examples is the necessity of escaping reserved metacharacters. Symbols like parentheses, asterisks, brackets, or plus signs all hold functional meaning in Regex. When these symbols are used literally as delimiters in your raw data, they must be preceded by a backslash to prevent the Regex engine from interpreting them as commands. Mastering this nuance is paramount for achieving reliable and consistent text extraction results across varied datasets.

While **REGEXEXTRACT** is exceptionally powerful for capturing content between delimiters, it forms part of a larger suite of text manipulation tools within Google Sheets. For related data processing needs, particularly when dealing with fixed positions or simpler delimiters, users should also explore related functions:

REGEXMATCH: Used to check if a specific pattern exists within a string, returning a boolean value (TRUE/FALSE).

REGEXREPLACE: Used to find a pattern and substitute it with a new string, useful for cleaning and normalizing data.

SPLIT: Used for breaking a cell's content into multiple cells based on a simple, fixed delimiter, such as a comma or space.

By integrating **REGEXEXTRACT** into your workflow, you gain sophisticated control over your text data, drastically improving efficiency in data cleaning and preparing your information for advanced data analysis tasks.

Google Sheets: How to Extract Text After a Character
Google Sheets: How to Extract Text Before a Character