Learning to Extract Text Between Characters in Google Sheets Using REGEXTRACT

Authored by

Mohammed looti

November 10, 2025

Welcome to this comprehensive guide on manipulating text strings within Google Sheets.
Data cleansing and extraction are fundamental processes in spreadsheet analysis, and often, analysts need to isolate specific substrings nestled between two defined delimiters or markers. Fortunately, Google Sheets provides a powerful tool for this purpose: the **REGEXEXTRACT** function.

The **REGEXEXTRACT** function uses **Regular Expressions** (often abbreviated as **Regex**) to search for patterns within a text string and return the matched substring. This method is vastly superior to combining simpler functions like `MID`, `FIND`, or `SEARCH`, especially when dealing with variable string lengths or complex delimiters.

To successfully extract all text between two specific characters or strings in a cell using **REGEXEXTRACT**, you must construct a precise regular expression pattern. This pattern uses a specific mechanism known as a **capturing group**, which tells the function exactly which part of the match result should be returned.

The general syntax utilized for this common task is shown below. This example demonstrates how to define both the start and end boundaries of the desired text extraction:

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

In this introductory formula, the function searches cell **A2**. It looks for the starting string "this," followed by any sequence of characters (represented by the regular expression pattern `(.*)`), and finally, the ending string "that." The crucial component is the **capturing group** `( )` surrounding `.*`. The `.` matches any character (except newline), and `*` specifies zero or more occurrences of the preceding element. Therefore, `(.*)` captures and returns everything between the defined boundaries.

Example 1: Extract Text Between Defined Strings

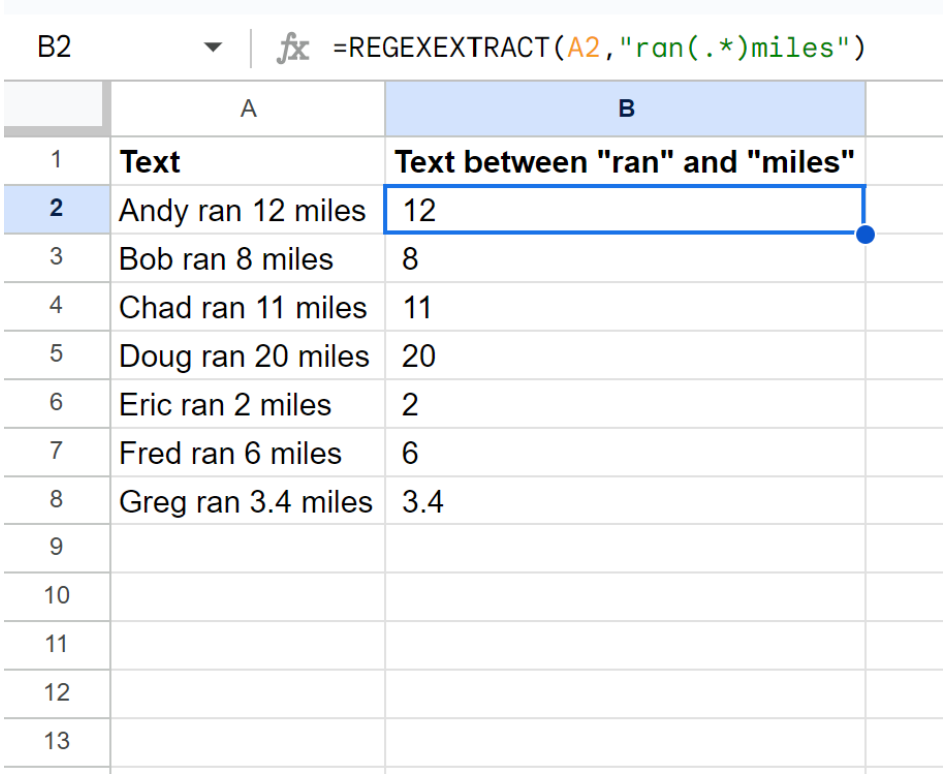
A frequent requirement in data processing involves extracting a value that is consistently positioned between two distinct text anchors. Consider a scenario where column A contains records detailing training runs, and we need to isolate only the time taken, which is always bracketed by the words "ran" and "miles."

To achieve this, we will enter the precise formula into cell **B2** and apply it across the dataset. This formula specifically targets the content in cell **A2**, isolating the text sandwiched between the specified anchor strings, "ran" and "miles."

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

Upon entering this formula, **REGEXEXTRACT** first locates the sequence "ran," then captures every subsequent character until it encounters the sequence "miles." The powerful flexibility of the **Regular Expressions** engine ensures that the extraction is performed reliably, regardless of the length or complexity of the text contained within those boundaries.

After successfully applying the formula to cell B2, the subsequent step involves using the fill handle--clicking and dragging the formula down--to apply the same logic to all remaining cells in column B. This procedure is efficient for processing large datasets in Google Sheets.



	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		

As evidenced by the resulting output, column B now accurately displays the text substring that existed between "ran" and "miles" for every corresponding entry in column A. This demonstrates the seamless capability of **REGEXEXTRACT** when dealing with standard text delimiters.

Example 2: Extract Text Between Parentheses

When the delimiters used for text extraction are special characters, or **metacharacters**, the regular expression syntax requires careful adjustment. Parentheses `()`, in particular, hold a reserved meaning in Regex--they are used to define **capturing groups**. If we intend to use the parentheses themselves as literal boundaries for the extraction, they must be "escaped" or handled specifically within the pattern.

The core challenge here is differentiating between the parentheses that define the literal text

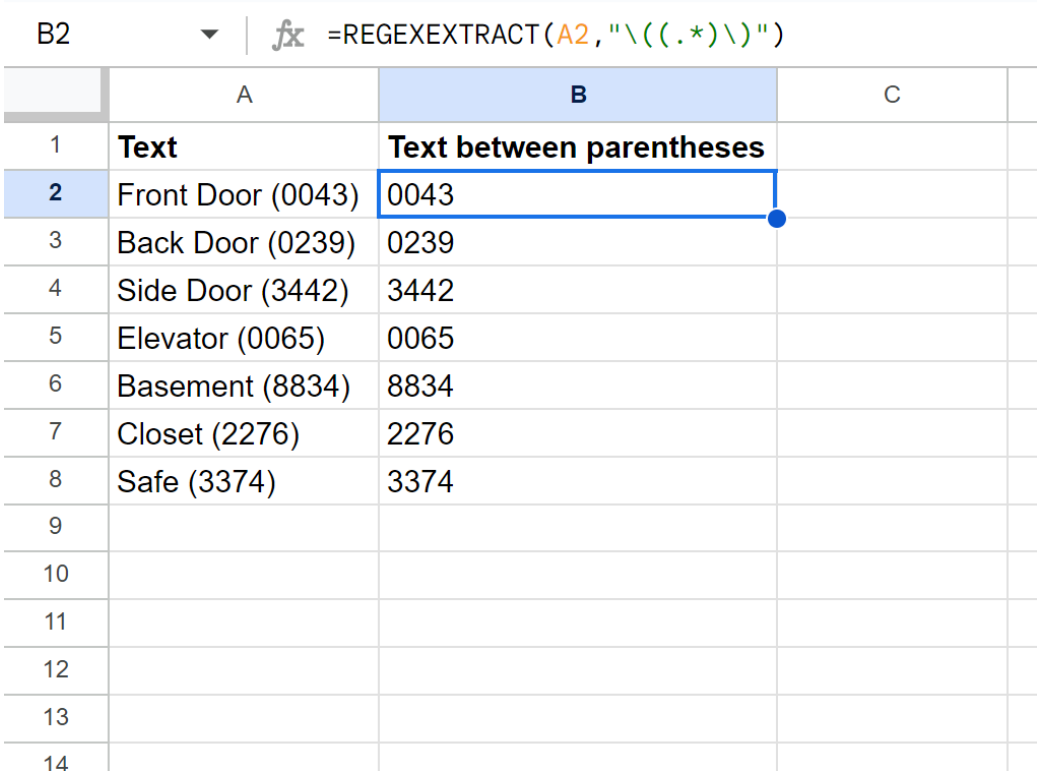
boundaries in the source string and the parentheses required by the **REGEXEXTRACT** function to define the content to be captured.

We will input the following formula into cell **B2** to extract the text located within the parentheses in cell **A2**:

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

The pattern `((.*))` instructs **REGEXEXTRACT** to look for a literal opening parenthesis, followed by any characters captured by `(.*)`, which is then terminated by a literal closing parenthesis. It is crucial to understand that the inner `(.*)` pair is what defines the content that is ultimately returned by the function, while the outer parentheses in this specific implementation define the literal search boundaries.

Once B2 is calculated, we efficiently replicate this formula down column B using the drag-and-fill method, ensuring that all entries in column A are processed according to the defined regular expression logic.



	A	B	C
1	**Text**	**Text between parentheses**	
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			

The resulting data in column B now isolates and displays only the text content that was contained within the parentheses of the corresponding strings in column A, demonstrating effective handling of reserved **metacharacters** used as delimiters.

Example 3: Extract Text Between Asterisks

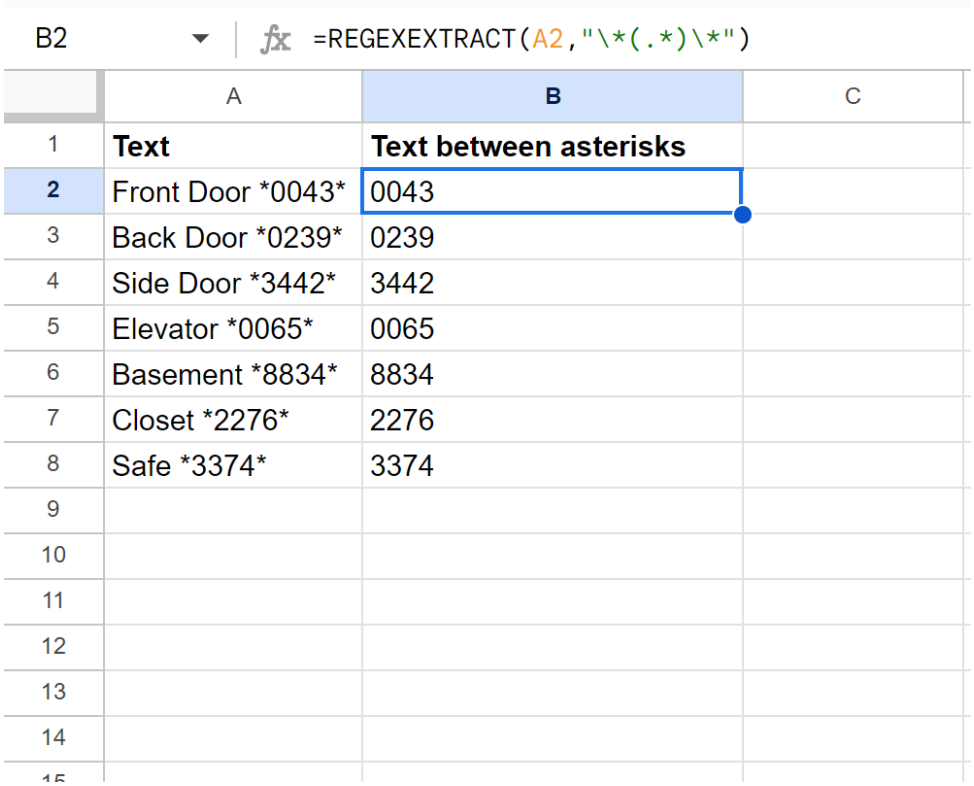
The asterisk (`*`) is another powerful **metacharacter** in **Regular Expressions**, signifying "zero or more occurrences" of the preceding item. Therefore, if we wish to use the asterisk as a literal boundary marker in our text extraction, we absolutely must use the **escape character** to strip the asterisk of its special meaning.

Failure to escape the asterisk will result in a faulty regex pattern that either returns an error or matches unintended parts of the string, as the engine will interpret the asterisk as a quantifier rather than a literal character. The standard escaped pattern for this scenario should be `*(.*)*`.

To extract the text in cell **A2** that is surrounded by asterisks, we input the following formula into cell **B2**:

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

After applying the formula to the initial cell, we subsequently use the standard click-and-drag method to populate the results across column B, ensuring that the text extraction is performed consistently for all input strings in column A.



The resulting data in column B confirms the successful extraction of text located between the

asterisks. This example highlights a critical aspect of working with **Regular Expressions** in Google Sheets: the necessity of correctly handling special characters.

Note: When defining a pattern that relies on reserved **metacharacters** like the asterisk (*) or dollar sign ($) as literal boundaries, it is mandatory to precede them with a backslash. This technique, known as escaping, ensures that the **REGEXTRACT** function correctly interprets the character as a physical delimiter rather than a quantifier or anchor within the regular expression engine. Although the visual representation of the formula above might simplify the expression, the underlying logic requires the escaped syntax `*(.*)*` for accurate processing across various Regex implementations.

Advanced Considerations for REGEXEXTRACT

While the basic implementation using `(.*)` is highly effective for most simple extractions, advanced data tasks may require modifications to the pattern. Understanding how **Regular Expressions** handle concepts like greediness and case sensitivity can significantly improve the robustness and reliability of your extraction formulas.

When defining your extraction pattern, especially in long strings, you must consider the greedy nature of the `.*` quantifier. By default, `.*` is "greedy," meaning it attempts to match the longest possible string between the first starting delimiter and the last ending delimiter. This can lead to unexpected results if your source string contains multiple instances of the delimiters.

To ensure the formula captures only the text between the *closest* set of delimiters (a non-greedy match), you must append a question mark to the quantifier: `(.*?)`. This modification forces the engine to stop matching immediately upon encountering the first instance of the closing delimiter. For example, if the source text is "TAG: Apple, pear. TAG: Banana, orange.", using `TAG: (.*?)` will capture only "Apple, pear." in the first extraction, providing the precision necessary for cleaning highly structured or nested data.

Additional Resources for Text Manipulation

Mastering the **REGEXEXTRACT** function is a cornerstone of advanced data manipulation in Google Sheets. For users looking to expand their skills further, the following tutorials explain related common tasks in spreadsheet environments:

Google Sheets: How to Extract Text After a Character

Google Sheets: How to Extract Text Before a Character

By leveraging **regular expressions**, users gain granular control over text data, moving beyond

simple string functions to handle complex, pattern-based extraction requirements efficiently and accurately, thereby significantly improving data quality and analytical capabilities.