Learning to Extract Text After a Specific Character in Google Sheets

Authored by

Mohammed looti

November 10, 2025

1. Introduction to Text Extraction Challenges

Working with data often requires surgical precision, especially when dealing with unstructured or semi-structured text inputs in a spreadsheet environment like Google Sheets. A common requirement for data analysts and power users is the need to isolate and extract specific substrings that occur immediately following a known character, word, or phrase. This process, often referred to as **text parsing**, is essential for cleaning data, isolating key identifiers, or preparing strings for further analysis. While simple functions like **MID** or **RIGHT** might suffice for fixed-length strings, they fail miserably when the delimiter's position changes dynamically within the cell.

The core challenge lies in defining a flexible rule that tells the spreadsheet exactly where to start the extraction, regardless of the text length preceding that point. For instance, if you have a list of product descriptions where the item number always follows the phrase "SKU:", the position of "SKU:" will vary across different cells. This necessitates a robust, pattern-matching solution. Fortunately, Google Sheets provides powerful functionality, primarily through its support for Regular Expressions (Regex), which are specifically designed to handle complex pattern recognition and extraction tasks efficiently.

In this comprehensive guide, we will focus on the most reliable and versatile method for extracting text following a specific character or string: utilizing the REGEXEXTRACT function. This function allows us to define a precise pattern that captures only the desired portion of the text, making it the superior choice over older, concatenated methods involving multiple nested functions. Understanding how to structure the regular expression pattern is key to unlocking this powerful capability within your data workflows.

2. Mastering the REGEXEXTRACT Solution

The most efficient and clean way to extract all text from a cell after a specific character or sequence is by using the **REGEXEXTRACT** function. This function is designed to search a string for a match to a Regular Expression pattern and return the matching substrings. When used correctly, it can isolate the trailing text without requiring complex calculations to determine the starting position or length of the extracted segment.

The general structure of the formula required to perform this specific type of extraction is demonstrated below. We are essentially telling the function to look for everything up to our chosen delimiter and then capture everything that follows it:

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

In the example shown, this specific formula is engineered to extract all content located in cell **A2**

that appears subsequent to the first occurrence of the string "our". It is crucial to understand the components of the regular expression pattern, `".*our(.*)"`, as this is where the precision of the extraction resides. The pattern dictates the search criteria and defines the capture group, which is the exact text the function returns. The simplicity and power of **REGEXEXTRACT** greatly streamline text manipulation tasks, reducing the risk of error associated with long, nested formulas.

This particular formula uses the **REGEXTRACT** function to extract all characters **(.*)** after the designated string, which in this case is **our**. By using parentheses, we ensure that only the text following the specific pattern is returned to the cell, efficiently discarding the leading content and the delimiter itself. This methodology is highly scalable across large datasets requiring consistent parsing logic.

3. Understanding the Regex Pattern Components

To effectively utilize **REGEXEXTRACT**, one must grasp the meaning of the components within the regular expression pattern. The pattern `".*our(.*)"` is composed of three main parts, each playing a vital role in identifying and capturing the desired text segment.

The three primary components are detailed below:

The Wildcard Prefix (`.*`): The initial sequence `.*` acts as a greedy wildcard. The dot (`.`) matches any single character (except newline, by default). The asterisk (`*`) is a quantifier meaning "zero or more occurrences" of the preceding element. When combined, `.*` matches any sequence of characters of any length that occurs before our target delimiter. This ensures that the function ignores all preceding text, effectively moving the starting point of our search to the beginning of the delimiter, regardless of its position in the string.

The Delimiter (`our`): This is the literal string or character that serves as the anchor point. In our primary example, this is "our". The **REGEXEXTRACT** function searches for and matches this exact sequence in the string. If you wanted to extract text after a specific punctuation mark, like a hyphen, you would place the hyphen here, ensuring proper escaping if the character has a special meaning in Regex.

The Capture Group (`(.*)`): This is the most critical element for extraction. The parentheses `()` define a "capture group." When **REGEXEXTRACT** finds a match for the entire pattern, it only returns the content enclosed within the capture group. Similar to the prefix, `.*` inside the parentheses matches "zero or more of any character." By placing this after the delimiter, we instruct the function to capture and return every single character that follows the anchor point until the end of the cell's content, thereby fulfilling the extraction requirement.

Therefore, the pattern `".*our(.*)"` translates to: "Match and consume any characters (`.*`), then

find the exact string 'our', and finally, capture and return all remaining characters ((`.*`))." This structure provides incredible flexibility, allowing users to easily substitute "our" with any other character or string they wish to use as a separator, ensuring the extracted text is always accurate and relevant to the desired position. This powerful combination is why regular expressions are indispensable for dynamic data processing.

4. Step-by-Step Example: Extracting Text After a Specific String

To demonstrate the practical application of the **REGEXEXTRACT** function, let us walk through a typical scenario involving a list of mixed text data where we need to consistently pull information following a known string. This example shows how to use this formula in practice.

Example: Extract Text After Character in Google Sheets

Suppose we have the following list of phrases in Google Sheets, located in column A:

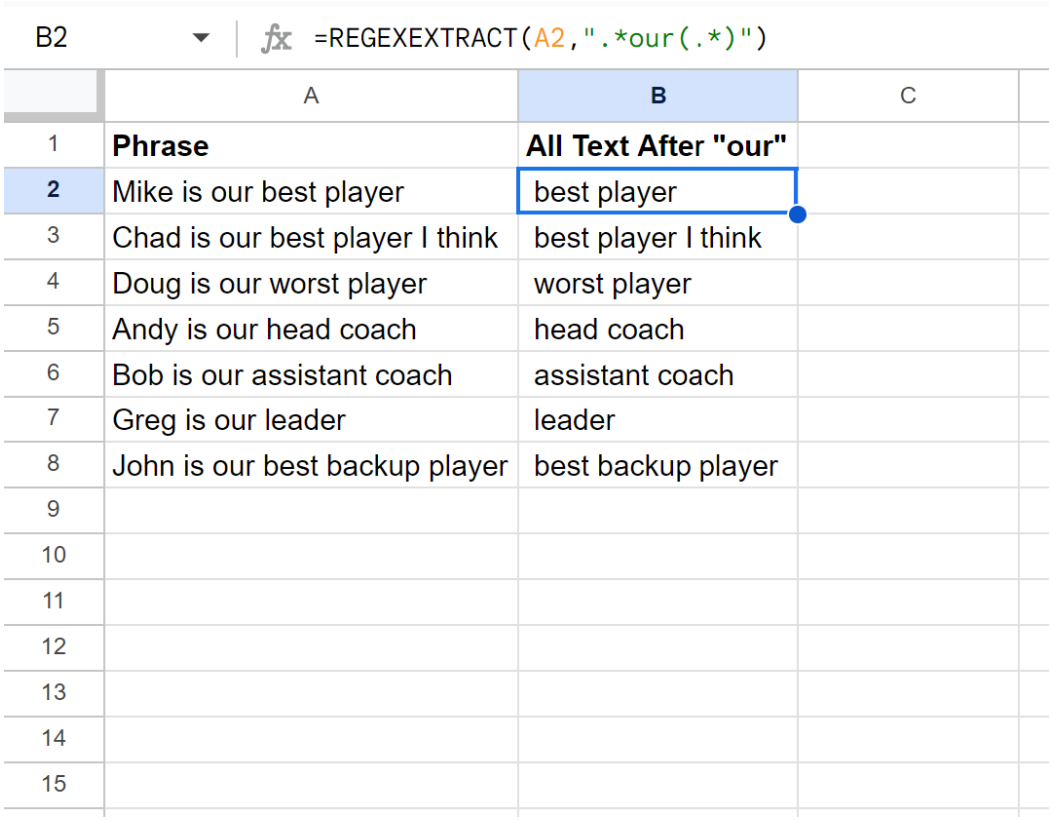
	A	B	C
1	**Phrase**		
2	Mike is our best player		
3	Chad is our best player I think		
4	Doug is our worst player		
5	Andy is our head coach		
6	Bob is our assistant coach		
7	Greg is our leader		
8	John is our best backup player		
9			
10			
11			
12			
13			
14			
15			
16			

Now suppose that we would like to extract all text from each cell after the specific string "our" is encountered. This requires the formula to dynamically locate "our" and begin extraction immediately after that point.

To do so, we can type the following formula into cell **B2**, targeting the content of cell A2:

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

Once the formula is entered, we can then click and drag this formula down to each remaining cell in column B. This autofill action propagates the pattern matching logic across the entire range of data in column A, ensuring every cell is processed correctly based on the delimiter "our".



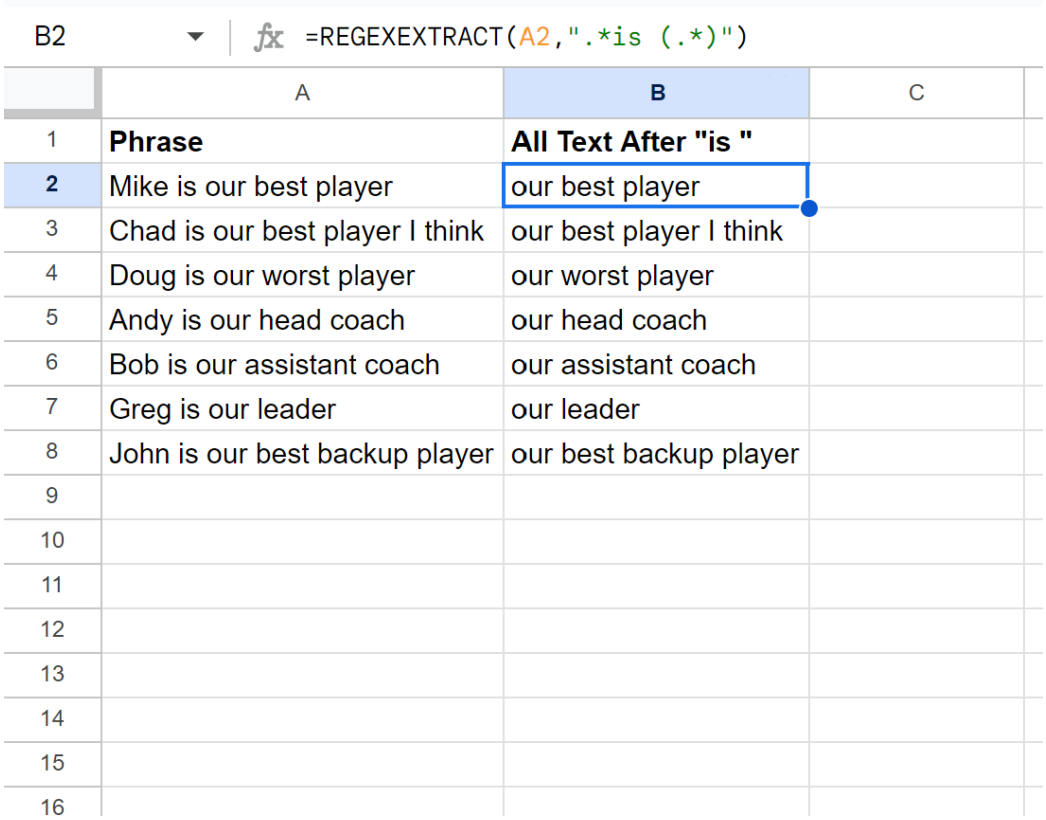
	A	B	C
1	**Phrase**	**All Text After "our"**	
2	Mike is our best player	best player	
3	Chad is our best player I think	best player I think	
4	Doug is our worst player	worst player	
5	Andy is our head coach	head coach	
6	Bob is our assistant coach	assistant coach	
7	Greg is our leader	leader	
8	John is our best backup player	best backup player	
9			
10			
11			
12			
13			
14			
15			

The result is that Column B now displays all text after "our" for each phrase in column A. This demonstrates the formula's effectiveness in isolating the desired substring dynamically. To extract the text after a different specific character, we simply need to replace **our** with the new delimiter string within the quotation marks.

For instance, if we wish to extract all text that follows the exact phrase "is " (including the space), we modify the pattern accordingly. For example, we could type the following formula into cell **B2** to extract all text after "is " from cell **A2**:

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

Applying this revised formula yields the following outcome:



	A	B	C
1	**Phrase**	**All Text After "is "**	
2	Mike is our best player	our best player	
3	Chad is our best player I think	our best player I think	
4	Doug is our worst player	our worst player	
5	Andy is our head coach	our head coach	
6	Bob is our assistant coach	our assistant coach	
7	Greg is our leader	our leader	
8	John is our best backup player	our best backup player	
9			
10			
11			
12			
13			
14			
15			
16			

Column B now displays all text after "is " for each phrase in column A, confirming the adaptability of the **REGEXEXTRACT** function based on the defined delimiter.

5. Exploring Alternative Extraction Methods

While **REGEXEXTRACT** is the recommended and most flexible method, especially when dealing with complex or varying data, Google Sheets does offer alternative approaches using standard string functions. These alternatives might be considered if the user is required to avoid regular expressions or if the delimiter is a single, non-special character. These methods generally require nesting multiple functions.

The most common non-regex approach involves combining MID and SEARCH. The logic is to first locate the delimiter, then calculate the precise starting point for the extraction (position of delimiter + length of delimiter), and finally use **MID** to grab the rest of the string.

For example, to extract text after the string "DELIMITER":

Step 1: Locate the Delimiter. Use the **SEARCH** function to find the starting index: `SEARCH("DELIMITER", A2)`.

Step 2: Calculate Extraction Start. Add the length of the delimiter (10 characters for

"DELIMITER") to the starting index: `SEARCH("DELIMITER", A2) + 10`.

Step 3: Extract the Remaining String. Use **MID**, starting at the calculated position, and specify a large number (like 1000) for the length to ensure all remaining characters are included: `=MID(A2, SEARCH("DELIMITER", A2) + 10, 1000)`.

Although functional, this formula relies on manual counting of the delimiter's length (or requires nesting **LEN** for true dynamism) and becomes cumbersome quickly. Furthermore, if the delimiter is not found, it generates a `#VALUE!` error, necessitating error handling through **IFERROR**.

A third option is the SPLIT function. **SPLIT** is excellent when the goal is to break a single string into multiple adjacent columns using a delimiter. If the user only wants the text that follows the delimiter, they must use **INDEX** to specifically select the second column generated by **SPLIT**. For instance, splitting by a comma `,` and retrieving the second part: `=INDEX(SPLIT(A2, ","), 1, 2)`. This method is concise for single-character delimiters but is less ideal for multi-character delimiters or when the user requires the output to remain strictly within a single cell without generating spillover. For complex, zero-footprint extraction, **REGEXEXTRACT** maintains its position as the superior tool.

6. Additional Resources

The capability to manipulate and parse text is fundamental in advanced spreadsheet analysis. Mastering functions like REGEXEXTRACT provides a strong foundation for handling complex data cleaning tasks. For users interested in delving deeper into text manipulation or other common analytical operations within Google Sheets, exploring related functions is highly recommended.

We encourage you to explore the official documentation and tutorials on related functions that complement text extraction:

REGEXMATCH: Used for checking if a string contains a specific pattern, returning TRUE or FALSE.

REGEXREPLACE: Essential for finding a pattern and substituting it with new text, useful for cleanup or standardization.

MID and SEARCH: Traditional functions for non-regex based extraction, often used in simpler, fixed-delimiter scenarios.

SPLIT: Ideal for separating content into multiple cells based on a defined delimiter.

The following tutorials explain how to perform other common tasks in Google Sheets: