# Converting Text to Numbers in Google Sheets: A Step-by-Step Guide with Examples

Authored by

**Mohammed looti**

October 31, 2025

Working with robust data analysis requires absolute precision, especially when handling datasets within applications like Google Sheets. A critical step in preparing data involves ensuring that numerical information is stored using the correct data type. Frequently, users encounter scenarios where values that appear to be numbers--such as financial figures, quantities, or IDs--are mistakenly classified by the spreadsheet software as text strings. This classification prevents standard mathematical operations, leading to failed formulas, inaccurate sums, and compromised reporting integrity.

The process of converting these text-formatted numbers back into usable numerical values is essential for effective data cleaning. This guide serves as a comprehensive resource, detailing three distinct and highly effective methods available in Google Sheets. Each method is specifically tailored to address a different complexity level, ranging from simple direct conversions to sophisticated extraction techniques required for parsing unstructured data. Understanding these specialized functions is fundamental to maintaining a clean and calculable dataset.

We will systematically explore the functions, their appropriate contexts, and provide detailed, step-by-step examples. By the conclusion of this tutorial, you will possess the expertise to confidently transform text into numerical formats, ensuring your data is always optimized for calculation and insightful analysis.

## Understanding the Data Type Conflict in Spreadsheets

The core issue arises when a cell contains characters that look like a number but are treated by the spreadsheet program as mere text. This often occurs during data importation from external sources, such as CSV files or databases, where formatting rules are misinterpreted. Another common cause is manual entry, particularly if the user precedes a number with a single apostrophe ('), which forces Google Sheets to interpret the entry as a literal string, overriding automatic type detection.

When data is stored as text, fundamental functions like **SUM()**, **AVERAGE()**, or operations involving multiplication will ignore those cells, leading to results that are unexpectedly lower than anticipated. Furthermore, text-based numbers sort alphabetically rather than numerically (e.g., "10" comes before "2"), severely hindering proper data organization. Recognizing these common pitfalls is the first step toward effective data remediation.

## Method 1: The Direct Conversion Using VALUE()

The VALUE() function is the most direct and efficient tool for converting clean text strings that strictly represent a number into a genuine numerical value. It is specifically designed to handle strings that might have been accidentally categorized as text due to import settings or a leading apostrophe. This function attempts to interpret the provided string based on standard numerical

formatting rules.

It operates by taking a single argument: the cell reference or the text string itself. If the string is a valid representation of a number--including standard decimal points or thousands separators recognized by your locale settings-- VALUE() will successfully return the corresponding numerical output. However, it is crucial to note its limitation: if the text contains arbitrary non-numeric characters (such as letters or excessive symbols), the function will return a **#VALUE!** error, indicating that the text cannot be interpreted as a standalone number.

This function is the foundation of simple text-to-number conversion and should be the first method attempted when dealing with seemingly clean numerical data that is behaving like text. The fundamental structure is simple and powerful:

**=VALUE(A1)**

## Method 2: Stripping Non-Numeric Formatting with TO_PURE_NUMBER()

When the numerical data is embedded with specific formatting elements, such as currency indicators ($ or €), percentage signs (%), or parentheses used for negative numbers, the TO_PURE_NUMBER() function offers a significantly more robust solution than VALUE(). This function is specifically engineered to strip away all non-numerical characters that are not essential to the core numerical value, returning only the raw numerical essence.

The utility of TO_PURE_NUMBER() lies in its ability to standardize dirty financial or statistical data quickly. For instance, if a cell contains the string "$1,250.00," this function recognizes the dollar sign and comma as formatting noise and efficiently isolates the number 1250. This makes it an indispensable tool during the initial phase of data preparation, especially when dealing with reports sourced from diverse systems that use inconsistent formatting conventions.

By relying on TO_PURE_NUMBER(), data analysts can bypass complex text manipulation formulas often needed to manually remove symbols, thereby ensuring that all subsequent calculations are based purely on the numerical magnitude of the data. The syntax remains straightforward:

**=TO_PURE_NUMBER(A1)**

## Method 3: Advanced Extraction from Mixed Strings

The most complex scenario involves unstructured data where numerical values are intertwined with arbitrary alphanumeric characters (e.g., "Order #4592 due date 01/01"). To handle such

cases, a powerful, multi-functional approach is required, leveraging the array manipulation capabilities of SPLIT() and CONCATENATE().

This advanced technique works by cleverly defining the delimiters. Instead of specifying what to keep, we specify what to ignore. The inner SPLIT() function breaks the original string into an array using the characters '0' through '9' and the decimal point as separators. This effectively isolates all non-numeric characters, which are then joined together using CONCATENATE() to form a single, comprehensive delimiter string.

The outer SPLIT() function then uses this newly created delimiter string (containing all the non-numeric characters) to split the original text again. The result is an array containing only the numerical segments. This is highly effective for sophisticated data parsing tasks in Google Sheets.

Here is the intricate formula designed for this extraction task:

**=SPLIT(A1,CONCATENATE(SPLIT(A1,".0123456789")))**

The following practical examples demonstrate how to implement each of these powerful formulas effectively, providing context for when and why you choose one method over the others.

## Example 1: Converting Simple Text Values to Numbers

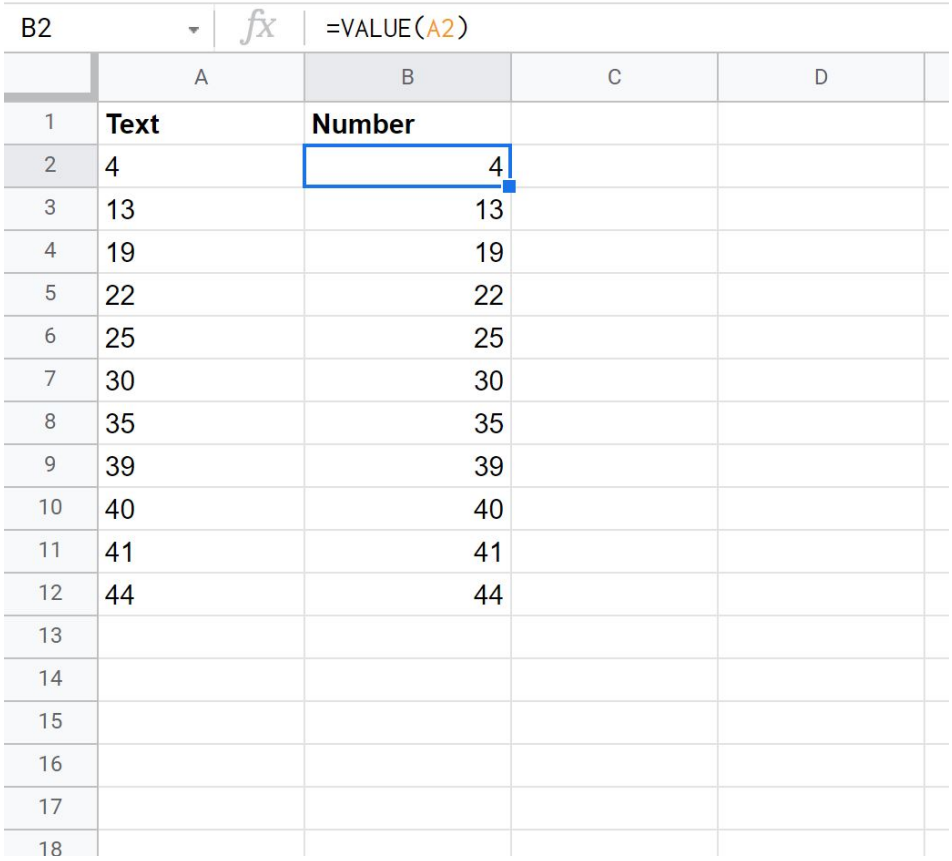
Consider a scenario where you have imported a column of simple numerical data (e.g., product counts, scores) into your spreadsheet, but they are displaying left-aligned, signifying that Google Sheets recognizes them as text strings. To rectify this fundamental data type error and enable these values to be used in standard arithmetic functions, we utilize the VALUE() function. This is the simplest fix for values that are numerically clean but incorrectly formatted.

To initiate the conversion, locate an adjacent blank column (Column B, for instance). In cell **B2**, input the formula **=VALUE(A2)**. This command explicitly instructs the software to interpret the content of cell **A2** as a number. Upon pressing Enter, the resulting value in **B2** should immediately display right-aligned, confirming its new numerical data type.

After successfully converting the first entry, you must apply this formula consistently across the entire dataset. Efficiently propagate the formula by clicking and dragging the fill handle (the small square at the bottom-right corner of cell **B2**) down to the last row of your data. This action instantaneously converts all corresponding text entries from Column A into their numerical equivalents in Column B, significantly streamlining your dataset and preparing it for mass calculations.

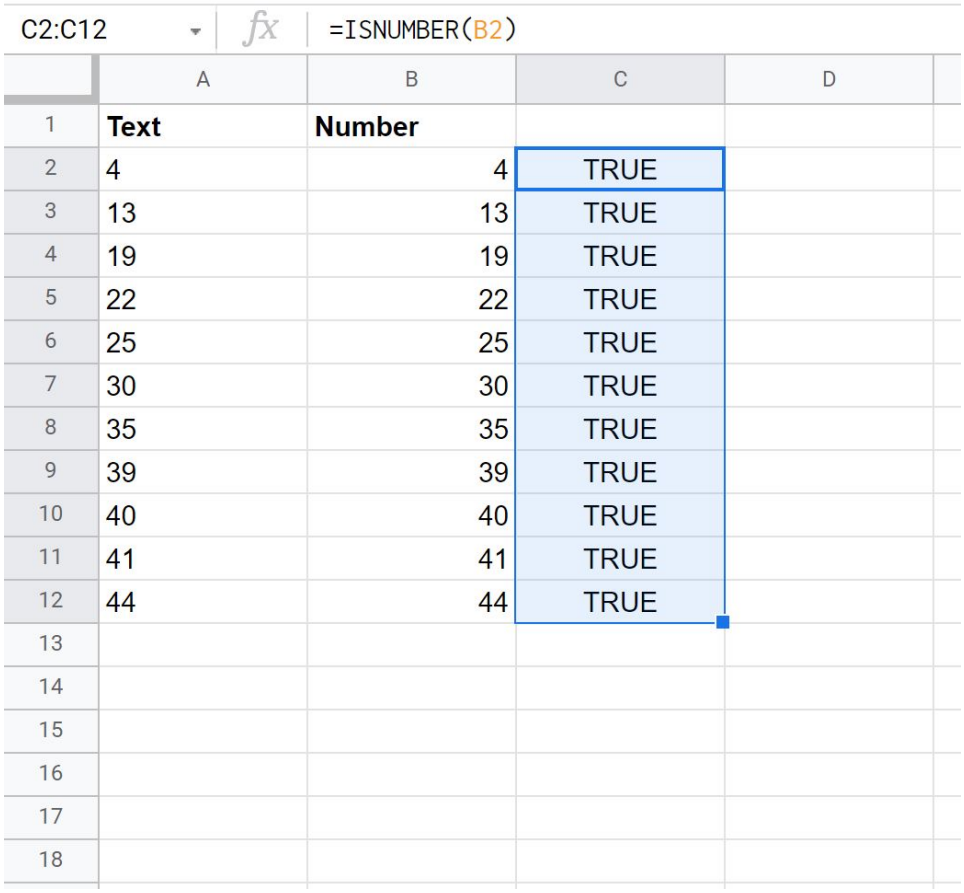
**=VALUE(A2)**

Once the conversions are complete, all the cells in Column B should now contain valid numbers, making them fully operational for any subsequent mathematical operations, statistical analysis, or charting requirements you wish to perform. The visual shift from left-alignment (text) to right-alignment (number) provides immediate confirmation of the successful operation.




| | A | B | C | D |
|---|---|---|---|---|
| 1 | **Text** | **Number** | | |
| 2 | 4 | 4 | | |
| 3 | 13 | 13 | | |
| 4 | 19 | 19 | | |
| 5 | 22 | 22 | | |
| 6 | 25 | 25 | | |
| 7 | 30 | 30 | | |
| 8 | 35 | 35 | | |
| 9 | 39 | 39 | | |
| 10 | 40 | 40 | | |
| 11 | 41 | 41 | | |
| 12 | 44 | 44 | | |
| 13 | | | | |
| 14 | | | | |
| 15 | | | | |
| 16 | | | | |
| 17 | | | | |
| 18 | | | | |

To confirm that the conversion was successful and that the values in Column B are indeed recognized as numbers by Google Sheets, employ the ISNUMBER() function. In cell **C2**, type the verification formula **=ISNUMBER(B2)**. This function returns a boolean value--**TRUE** if the cell contains a number, **FALSE** otherwise--providing unequivocal proof of data integrity.




| | A | B | C | D |
|---|---|---|---|---|
| 1 | **Text** | **Number** | | |
| 2 | 4 | 4 | TRUE | |
| 3 | 13 | 13 | TRUE | |
| 4 | 19 | 19 | TRUE | |
| 5 | 22 | 22 | TRUE | |
| 6 | 25 | 25 | TRUE | |
| 7 | 30 | 30 | TRUE | |
| 8 | 35 | 35 | TRUE | |
| 9 | 39 | 39 | TRUE | |
| 10 | 40 | 40 | TRUE | |
| 11 | 41 | 41 | TRUE | |
| 12 | 44 | 44 | TRUE | |
| 13 | | | | |
| 14 | | | | |
| 15 | | | | |
| 16 | | | | |
| 17 | | | | |
| 18 | | | | |

As clearly illustrated in the image above, the ISNUMBER() function yields **TRUE**. This result definitively confirms that the output in cell **B2** is now recognized as a legitimate numerical value, concluding the conversion and validation process for simple text entries.

## Example 2: Transforming Currency Formats into Pure Numbers

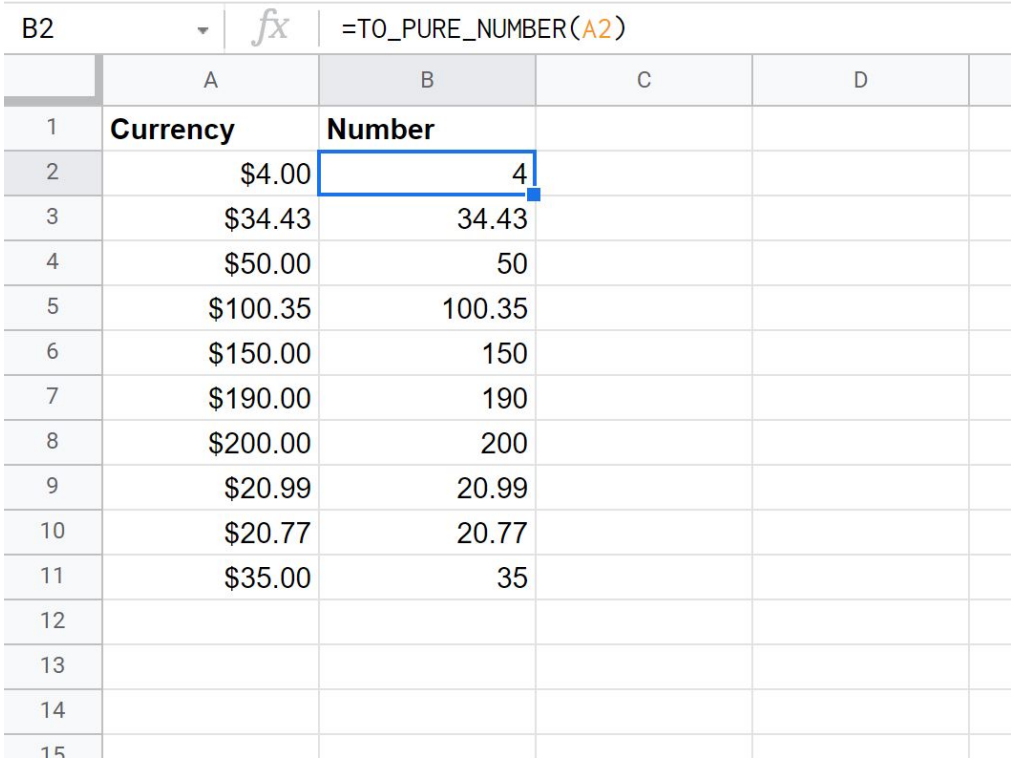
In financial analysis, raw data often includes currency symbols, commas for thousands, and other descriptive elements that inhibit direct calculation. For instance, if you receive a spreadsheet where monetary values are explicitly formatted as text (e.g., "$1,234.50" or "(500.00)"), the goal is to extract the underlying magnitude without manually cleaning each entry. The TO_PURE_NUMBER() function is specifically designed for this task, acting as a powerful filter that strips away all extraneous non-numeric formatting.

To begin the extraction, place the cursor in cell **B2**, parallel to the first currency entry in **A2**. Input the formula **=TO_PURE_NUMBER(A2)**. This function processes the text string in **A2**, intelligently identifying and removing the dollar sign, comma, and any potential text indicators, leaving only the fundamental numerical component. This is critical for ensuring consistency across columns derived from various data sources.

Once the formula is correctly entered in **B2**, replicate the conversion across the entire column. Utilize the fill handle to copy the formula down column B. This efficient propagation ensures that all subsequent currency values in Column A are systematically converted into a clean, unformatted numerical dataset in Column B. This resulting pure numerical data is now ready for aggregation, complex financial modeling, or input into external analytical tools.

**=TO_PURE_NUMBER(A2)**

As demonstrated in the visual aid below, every cell in Column B now holds a distinct numerical value, successfully isolated from its original currency or special text formatting. Note that while the original formatting is removed, you can easily reapply standard number formatting (e.g., two decimal places) to Column B without losing its numerical integrity.




| | A | B | C | D |
|---|---|---|---|---|
| 1 | **Currency** | **Number** | | |
| 2 | $4.00 | 4 | | |
| 3 | $34.43 | 34.43 | | |
| 4 | $50.00 | 50 | | |
| 5 | $100.35 | 100.35 | | |
| 6 | $150.00 | 150 | | |
| 7 | $190.00 | 190 | | |
| 8 | $200.00 | 200 | | |
| 9 | $20.99 | 20.99 | | |
| 10 | $20.77 | 20.77 | | |
| 11 | $35.00 | 35 | | |
| 12 | | | | |
| 13 | | | | |
| 14 | | | | |
| 15 | | | | |

To confirm the successful transformation and guarantee data usability, perform a final validation using the ISNUMBER() function. Entering **=ISNUMBER(B2)** into cell **C2** provides immediate, binary confirmation of the data type. This step is indispensable for quality control in any data cleaning workflow.

C2 fx =ISNUMBER(B2)

| | A | B | C | D |
|---|---|---|---|---|
| 1 | **Currency** | **Number** | | |
| 2 | $4.00 | 4 | TRUE | |
| 3 | $34.43 | 34.43 | | |
| 4 | $50.00 | 50 | | |
| 5 | $100.35 | 100.35 | | |
| 6 | $150.00 | 150 | | |
| 7 | $190.00 | 190 | | |
| 8 | $200.00 | 200 | | |
| 9 | $20.99 | 20.99 | | |
| 10 | $20.77 | 20.77 | | |
| 11 | $35.00 | 35 | | |
| 12 | | | | |
| 13 | | | | |
| 14 | | | | |
| 15 | | | | |
| 16 | | | | |

The returned output of **TRUE** from the ISNUMBER() function confirms that the value in cell **B2** is correctly recognized as a true number, validating the effectiveness of the TO_PURE_NUMBER() function in handling formatted text entries.

## Example 3: Isolating Numerical Data from Mixed Text Strings

Sometimes, numbers are embedded within longer text descriptions (e.g., "Product ID: 12345" or "Quantity is 50 units"). Extracting these numbers requires a more sophisticated approach. This example demonstrates how to use a combination of SPLIT() and CONCATENATE() functions to precisely isolate numerical values from mixed text strings.

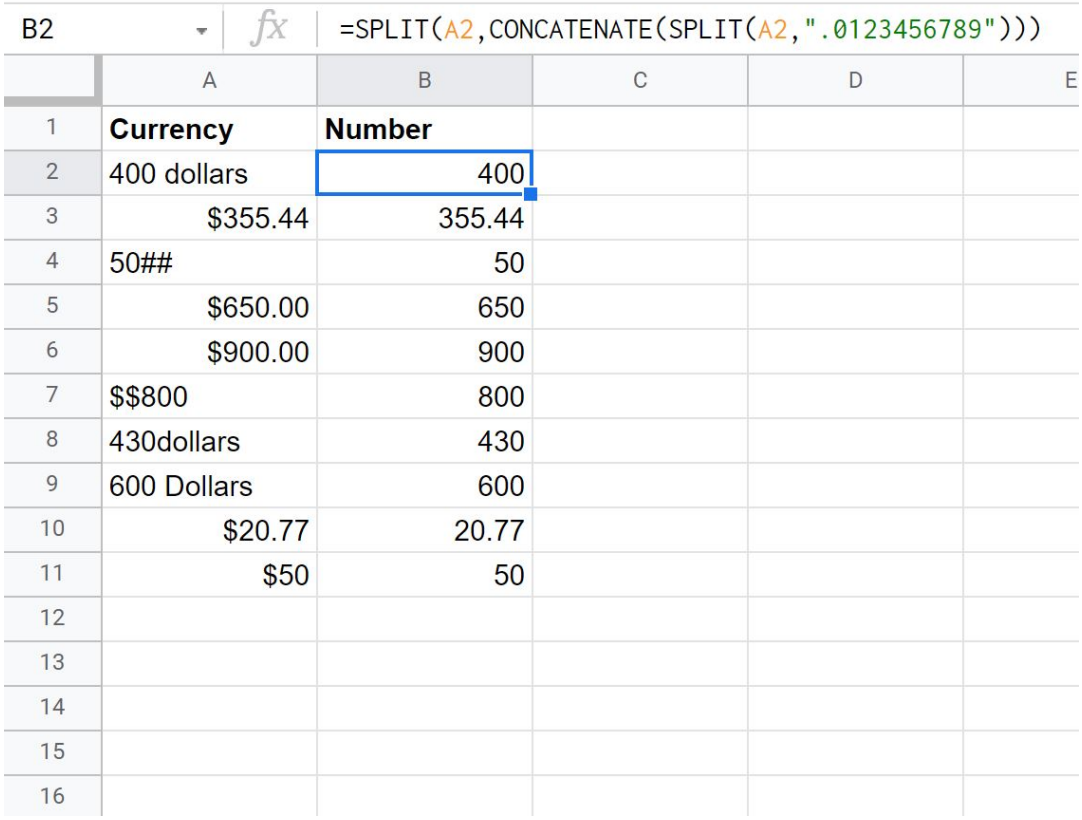
To execute this advanced extraction, input the composite formula into cell **B2**, referencing the mixed text string in **A2**. The formula's brilliance lies in its recursive use of SPLIT(): it first identifies all characters that are *not* a number or decimal point, aggregates them into a single delimiter via CONCATENATE(), and then uses that aggregated list of unwanted characters to perform the final split, isolating the numerical fragments.

This method often results in multiple columns containing fragments of numbers if the original text contained multiple distinct numerical sequences. Analysts must then manage the resulting array to select the specific numerical value needed, often by using an **INDEX()** function if the desired

number is consistently located in the first resulting column. For this simplified example focused on extraction, the formula successfully segments the number from the text.

**=SPLIT(A2,CONCATENATE(SPLIT(A2,".0123456789")))**

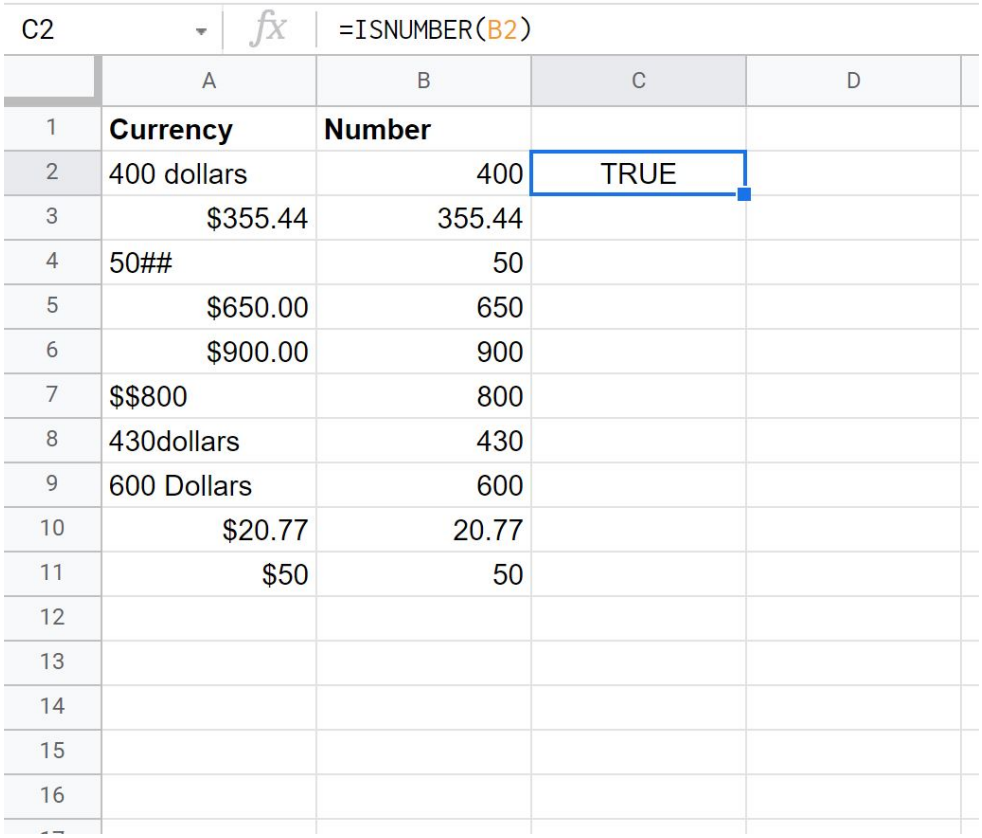
Once the complex formula is correctly established in **B2**, replicate it by dragging the fill handle down the column. This systematic application will process all corresponding mixed text strings in Column A, resulting in Column B containing the extracted numerical data, now ready for use as unique identifiers or quantifiable metrics.




| | A | B | C | D | E |
|---|---|---|---|---|---|
| 1 | **Currency** | **Number** | | | |
| 2 | 400 dollars | 400 | | | |
| 3 | $355.44 | 355.44 | | | |
| 4 | 50## | 50 | | | |
| 5 | $650.00 | 650 | | | |
| 6 | $900.00 | 900 | | | |
| 7 | $$800 | 800 | | | |
| 8 | 430dollars | 430 | | | |
| 9 | 600 Dollars | 600 | | | |
| 10 | $20.77 | 20.77 | | | |
| 11 | $50 | 50 | | | |
| 12 | | | | | |
| 13 | | | | | |
| 14 | | | | | |
| 15 | | | | | |
| 16 | | | | | |

Upon successful execution, every cell within column B will now contain the extracted numerical data, converted into a usable number format.

To confirm the successful extraction and the proper data type conversion, perform the final validation step. Enter the ISNUMBER() function--**=ISNUMBER(B2)**--into cell **C2**. This provides immediate verification of the outcome.




| | A | B | C | D |
|---|---|---|---|---|
| 1 | **Currency** | **Number** | | |
| 2 | 400 dollars | 400 | TRUE | |
| 3 | $355.44 | 355.44 | | |
| 4 | 50## | 50 | | |
| 5 | $650.00 | 650 | | |
| 6 | $900.00 | 900 | | |
| 7 | $$800 | 800 | | |
| 8 | 430dollars | 430 | | |
| 9 | 600 Dollars | 600 | | |
| 10 | $20.77 | 20.77 | | |
| 11 | $50 | 50 | | |
| 12 | | | | |
| 13 | | | | |
| 14 | | | | |
| 15 | | | | |
| 16 | | | | |

The appearance of **TRUE** as the output from the ISNUMBER() function confirms that the complex extraction method was successful, and the resulting value in cell **B2** is correctly recognized as a numerical format, thereby validating the integrity of the transformed data.

## Conclusion and Best Practices for Data Integrity

Mastering the conversion of text to numbers in Google Sheets is not merely a technical step; it is a crucial prerequisite for efficient and accurate data management and analysis. Whether your challenge involves simple text entries using VALUE(), handling formatted currency with TO_PURE_NUMBER(), or parsing complex mixed strings via the combined power of array functions, the methods detailed in this guide provide robust and scalable solutions.

Ensuring that your data types are correctly assigned prevents calculation errors, enables proper sorting, and significantly enhances the reliability of your spreadsheet operations. Adopting a proactive approach to data cleaning, particularly after data importation, saves substantial time and mitigates the risk of drawing flawed conclusions from compromised data. Always remember to utilize verification functions like **ISNUMBER()** to confirm that your transformations have achieved the desired result.

We strongly encourage you to integrate these functions into your regular workflow. Proficiency in

these text-to-number techniques is a hallmark of an expert spreadsheet user, paving the way for more sophisticated data modeling and insightful reporting.

For additional insights and to further enhance your Google Sheets skills, consider exploring the following related tutorials: