Learning SAS: A Guide to String Concatenation with CAT, CATS, and CATX Functions

Authored by

Mohammed looti

October 31, 2025

RECOMMENDED CITATION

Mohammed looti (2025). *Learning SAS: A Guide to String Concatenation with CAT, CATS, and CATX Functions*. PSYCHOLOGICAL STATISTICS. Retrieved from https://statistics.arabpsychology.com/?p=7451

String manipulation is a fundamental requirement in almost every data processing workflow. In SAS, the process of combining two or more character variables into a single variable is known as string concatenation. While traditional methods like the double pipe operator (`||`) exist, SAS provides a powerful family of functions--the CAT functions--that are specifically designed to handle common data issues like leading and trailing blanks, ensuring cleaner and more predictable results.

This guide, written for data analysts and programmers, will explore the three most essential CAT functions available within the SAS Data Step. We will demonstrate how to use each function effectively to combine strings, control spacing, and introduce custom separators, ultimately leading to more robust data preparation scripts.

Understanding String Concatenation in SAS

Concatenation involves linking data elements end-to-end. For instance, combining a first name and a last name to form a single full name field. In SAS, character variables often retain their defined length, which can result in trailing spaces or "padding." If you use the simple `||` operator, these trailing spaces are preserved, leading to unexpected gaps or formatting errors in the resulting string.

The CAT family of functions--CAT functions--were created to address this problem. These functions automatically strip or ignore leading and trailing blanks from the arguments before concatenation, offering far greater control over the output compared to manual methods. Furthermore, they handle numeric arguments gracefully, automatically converting them to character strings before combining them with other variables.

Choosing the right CAT function depends entirely on how you want to manage the spacing between the concatenated elements. Do you need a single space, no space, or a specific character as a separator? The following methods provide solutions for each scenario.

The Three Essential SAS Concatenation Functions

The three primary CAT functions provide distinct ways to manage the separation of your input strings. Understanding their nuances is key to writing efficient and error-free SAS code.

The first method, utilizing the **CAT function**, is designed for scenarios where you require a single blank space between each concatenated item. This is the most common use case when dealing with names or addresses where readability is paramount. It efficiently removes all trailing and leading blanks from the arguments and then inserts a single space delimiter between them.

new_variable = CAT(var1, var2);

The second powerful function is **CATS**, standing for "Concatenate and Trim Spaces." This function is the fastest and most direct method when absolutely no space or delimiter is desired between the elements. It strips both leading and trailing blanks from all arguments and joins them together seamlessly. This is ideal for constructing unique identifiers or file paths.

new_variable = CATS(var1, var2);

Finally, the **CATX function** (Concatenate with Delimiter) provides the ultimate flexibility, allowing the user to specify a custom separator--such as a comma, a hyphen, or a pipe symbol--to be inserted between the arguments. Like the other CAT functions, it intelligently handles and strips leading and trailing blanks from the source variables, ensuring that the custom delimiter is placed correctly and only between elements that contain values.

new_variable = CATX("-", var1, var2);

Setting Up the Demonstration Dataset

To effectively illustrate the difference between these three concatenation methods, we will apply them to a simple dataset containing basic employee information. This dataset includes a first name, a last name, and a numeric variable representing points. Note that in SAS, the dollar sign (`$`) following a variable name in the `input` statement denotes a character variable, which is necessary for string concatenation.

The following code block demonstrates the creation of the sample dataset named `my_data1`, followed by the necessary procedure (`proc print`) to display the structure and content of the source data before any transformations occur. This baseline view is crucial for comparing the outputs in the subsequent examples.

```
/*create dataset*/
data my_data1;
input firstName $ lastName $ points;
datalines;
Austin Smith 15
Brad Stevens 31
Chad Miller 22
Dave Michaelson 19
Eric Schmidt 29
Frank Wright 20
Greg Gunner 40
Harold Anderson 35
```

```
;
run;

/*view dataset*/
proc print data=my_data1;
```

The resulting table confirms that we have two character fields (`firstName` and `lastName`) ready for combination, alongside a numeric field (`points`).

Obs	firstName	lastName	points
1	Austin	Smith	15
2	Brad	Stevens	31
3	Chad	Miller	22
4	Dave	Michaels	19
5	Eric	Schmidt	29
6	Frank	Wright	20
7	Greg	Gunner	40
8	Harold	Anderson	35

Example 1: Concatenating Strings Using CAT (Space Delimiter)

The CAT function is perhaps the most frequently used of the concatenation family when dealing with human-readable text. Its primary behavior is to automatically insert a single space between the strings it combines, after first removing any extraneous leading or trailing blanks that might be present in the source variables. This ensures that regardless of the defined length of the input variables, the resulting string contains only one clean space between them.

In this example, we create a new variable named **fullName** by combining the **firstName** and **lastName** variables. Notice that the syntax is straightforward: simply list the variables to be combined inside the function arguments. This operation is performed within a new SAS Data Step (`my_data2`) which reads from the original dataset (`my_data1`).

The following code shows how to create a new column called **fullName** that concatenates the **firstName** and **lastName** columns using a blank space as a delimiter, which is the default behavior of the CAT function:

```
/*create new dataset with concatenated strings*/
data my_data2;
```

```
set my_data1;
fullName = CAT(firstName, lastName);
run;

/*view new dataset*/
proc print data=my_data2;
```

As evidenced in the output, the resulting **fullName** column successfully combines the names with a single, clean space separating the first and last name for every observation, confirming the effective blank-stripping and spacing provided by the CAT function.

Obs	firstName	lastName	points	fullName
1	Austin	Smith	15	Austin Smith
2	Brad	Stevens	31	Brad Stevens
3	Chad	Miller	22	Chad Miller
4	Dave	Michaels	19	Dave Michaels
5	Eric	Schmidt	29	Eric Schmidt
6	Frank	Wright	20	Frank Wright
7	Greg	Gunner	40	Greg Gunner
8	Harold	Anderson	35	Harold Anderson

Example 2: Concatenating Strings Using CATS (No Space/Trimmed)

When the requirement is to merge strings without any intervening characters, the **CATS function** is the ideal choice. CATS stands out because it not only strips leading and trailing blanks from all arguments but also ensures that zero spaces or separators are introduced during the string concatenation process. This is particularly useful in data transformation tasks where strings must be tightly packed, such as forming login credentials, email addresses (though typically involving more complex logic), or unique keys composed of multiple data fields.

Using the same source dataset, this example demonstrates the use of CATS to create the **fullName** variable. Unlike CAT, which defaults to a space, CATS performs a direct merge. It is crucial to remember that the CATS function is the most aggressive in removing whitespace; if you intended to retain any internal spaces within an argument (for example, a multi-word city name), you must handle that trimming before passing it to CATS.

The following code shows how to create a new column called **fullName** that concatenates the **firstName** and **lastName** columns using no space as a delimiter, highlighting the direct trimming

and merging capabilities of the CATS function:

```
/*create new dataset with concatenated strings*/
data my_data2;
set my_data1;
fullName = CATS(firstName, lastName);
run;

/*view new dataset*/
proc print data=my_data2;
```

Reviewing the output reveals that the first and last names are merged immediately adjacent to one another. For example, "Austin Smith" becomes "AustinSmith." This result confirms that CATS successfully stripped any padding from the input variables before combining them, fulfilling the requirement for tightly packed strings.

Obs	firstName	lastName	points	fullName
1	Austin	Smith	15	AustinSmith
2	Brad	Stevens	31	BradStevens
3	Chad	Miller	22	ChadMiller
4	Dave	Michaels	19	DaveMichaels
5	Eric	Schmidt	29	EricSchmidt
6	Frank	Wright	20	FrankWright
7	Greg	Gunner	40	GregGunner
8	Harold	Anderson	35	HaroldAnderson

Example 3: Concatenating Strings with Custom Delimiter

The **CATX function** is the most flexible of the CAT functions, allowing the user to define exactly what separates the concatenated elements. This function requires the delimiter to be specified as the very first argument. Like its counterparts, CATX automatically handles leading and trailing blanks, but it also offers a key advantage: it will only insert the custom delimiter between non-missing or non-blank arguments. If an argument is empty, CATX skips the insertion of the delimiter for that position, preventing erroneous consecutive delimiters (e.g., "Name--").

In this final example, we will use a hyphen (`-`) as our custom delimiter. This scenario is common when creating standardized codes, file names, or hyphenated names. The use of CATX provides a robust solution that is superior to manually inserting delimiters using the `||` operator, especially in

datasets where missing values might occur.

The following code shows how to create a new column called **fullName** that concatenates the **firstName** and **lastName** columns using a dash as a delimiter, clearly demonstrating the syntax required for the CATX function:

```
/*create new dataset with concatenated strings*/
data my_data2;
set my_data1;
fullName = CATX("-", firstName, lastName);
run;

/*view new dataset*/
proc print data=my_data2;
```

The output displays the **fullName** variable with the first and last names separated precisely by the hyphen, such as "Austin-Smith." This confirms that CATX is functioning correctly, handling the blank suppression and inserting the specified custom separator between the non-missing string components.

Obs	firstName	lastName	points	fullName
1	Austin	Smith	15	Austin-Smith
2	Brad	Stevens	31	Brad-Stevens
3	Chad	Miller	22	Chad-Miller
4	Dave	Michaels	19	Dave-Michaels
5	Eric	Schmidt	29	Eric-Schmidt
6	Frank	Wright	20	Frank-Wright
7	Greg	Gunner	40	Greg-Gunner
8	Harold	Anderson	35	Harold-Anderson

Summary and Further Resources

The SAS CAT family of functions offers essential tools for manipulating character data within the SAS Data Step. By automatically trimming leading and trailing blanks, these functions eliminate common pitfalls associated with manual string concatenation, providing analysts with reliable and readable output.

To summarize your choice:

Use **CAT** when you need a single, standard space between elements (e.g., proper names).

Use **CATS** when you need zero spacing between elements (e.g., forming technical codes or IDs).

Use **CATX** when you need a specific character (delimiter) between elements, or when dealing with missing values that should not result in consecutive separators.

Mastering these three functions is a critical step toward efficient data cleaning and preparation in SAS.

Additional Resources

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