How to Easily Distinguish Between the PUT and INPUT Statements in SAS

Authored by

stats writer

November 19, 2025

The SAS programming environment offers powerful tools for data manipulation, chief among them the PUT and INPUT elements. While often discussed together, it is essential to recognize that both terms refer to distinct functionalities within the system: they exist as both SAS statements used for data flow and as SAS functions used for data type conversion.

Understanding the context is critical when working with these commands. As a statement, the PUT statement is primarily used for writing data--it outputs a character string, a numeric value, or the formatted value of a variable to destinations like the SAS log or an external file. Conversely, the INPUT statement performs the opposite action, reading raw data values from an external source or an internal line of data, and subsequently storing those values into defined SAS variables within the DATA step. This distinction between writing out (PUT) and reading in (INPUT) is fundamental to data handling in SAS.

PUT and INPUT as Data Type Conversion Functions

Beyond their role as statements for I/O operations, the PUT function and the INPUT function are indispensable tools for converting variables between different data types, such as converting a numeric variable into a character variable, or vice versa. These functions are often utilized within assignment statements inside the DATA step to ensure compatibility when performing calculations, merging data, or exporting results.

The core difference lies in the direction of the conversion and the types of variables they handle. The PUT function is specialized for converting data into character format, while the INPUT function is used for converting character data into either numeric or character formats, typically numeric. Mastering these functions is vital for complex data preparation and cleaning tasks in the SAS environment.

Detailed Behavior of the PUT and INPUT Functions

The functionality of the PUT function can be summarized simply: it accepts either character or numeric variables as its input argument, but it will **always output character variables**. This conversion process requires specifying a SAS format, which dictates how the input value should be represented as a character string. For example, converting a date value (which SAS stores as a number) requires a date format within the PUT function to output a human-readable date string.

In contrast, the INPUT function operates under stricter input requirements. It **only accepts character variables** as its input. However, it offers flexibility in output type, capable of generating either a character variable or a numeric variable, depending on the informat specified. The informat instructs SAS on how to interpret the character string to produce the desired output data type. This is commonly used when raw data is read into SAS as character strings (to prevent data loss from

non-standard characters) and must then be explicitly converted to numeric values for calculation.

The following detailed examples illustrate how these two crucial functions are employed in a practical SAS context to manage data type transformations effectively.

Example 1: Using PUT to Convert Numeric Variable to Character Variable

Consider a scenario where we have sales data, and the day identifier, currently stored as a number, needs to be treated as a character string for merging purposes or specific reporting requirements. We begin by creating a simple dataset that records total sales over ten consecutive days, ensuring both variables, `day` and `sales`, are initially stored as numeric variables.

```
/*create dataset*/
data original_data;
input day sales;
datalines;
1 7
2 12
3 15
4 14
5 13
6 11
7 10
8 16
9 18
10 24
;
run;

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

Obs	day	sales
1	1	7
2	2	12
3	3	15
4	4	14
5	5	13
6	6	11
7	7	10
8	8	16
9	9	18
10	10	24

Reviewing Initial Data Types with PROC CONTENTS

To confirm the data structure before conversion, we use the powerful PROC CONTENTS procedure. This utility provides comprehensive metadata about the dataset, including the type and length of each variable. This step is crucial for verifying the necessity of data conversion.

```
/*display data type for each variable*/
proc contents data=original_data;
```

Alphabetic List of Variables and Attributes			
#	Variable	Type	Len
1	day	Num	8
2	sales	Num	8

As confirmed by the output, both the `day` and `sales` variables are designated as numeric variables. Our objective is now to transform `day` into a character variable using the PUT function.

Executing Numeric-to-Character Conversion using PUT

We proceed to implement the PUT function within a new DATA step to create the converted variable. The syntax `char_day = put(day, 8.);` instructs SAS to take the numeric value of `day` and convert it into a character string, utilizing the standard numeric format (`8.`) to define the length of the resulting character field. We simultaneously drop the original numeric `day` variable to retain only the new character version in the resulting dataset.

```
/*create new dataset where 'day' is character*/
data new_data;
set original_data;
char_day = put(day, 8.);
drop day;
run;

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

Obs	sales	char_day
1	7	1
2	12	2
3	15	3
4	14	4
5	13	5
6	11	6
7	10	7
8	16	8
9	18	9
10	24	10

To verify the success of the transformation, we run PROC CONTENTS again on the `new_data` dataset. This final check confirms that the variable `char_day` is indeed a character type, validating the correct application of the PUT function.

```
/*display data type for each variable in new dataset*/
proc contents data=new_data;
```

Alphabetic List of Variables and Attributes			
#	Variable	Type	Len
2	char_day	Char	8
1	sales	Num	8

The successful execution confirms that `day` was converted from numeric to a new character

variable, `char_day`, demonstrating the precise utility of the PUT function in SAS.

Example 2: Using INPUT to Convert Character Variable to Numeric Variable

For the second example, we explore the reverse transformation, converting a character variable into a numeric variable using the INPUT function. This is often necessary when data sources, such as imported CSV files, default numeric fields to character type to handle potential inconsistencies or missing values. We start with a dataset where the `day` variable is explicitly defined as character using the `$` sign in the INPUT statement.

```
/*create dataset*/
data original_data;
input day $ sales;
datalines;
1 7
2 12
3 15
4 14
5 13
6 11
7 10
8 16
9 18
10 24
;
run;

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

Obs	day	sales
1	1	7
2	2	12
3	3	15
4	4	14
5	5	13
6	6	11
7	7	10
8	8	16
9	9	18
10	10	24

We confirm the initial data types using PROC CONTENTS to verify that `day` is indeed recognized as a character variable before we attempt the conversion. This step is critical to ensure that the INPUT function is being applied correctly to the source data type it expects.

```
/*display data type for each variable*/
proc contents data=original_data;
```

Alphabetic List of Variables and Attributes			
#	Variable	Type	Len
1	day	Char	8
2	sales	Num	8

The output confirms that `day` is a character variable, indicated by the 'Char' type listing, while `sales` remains a numeric variable. Now we can proceed with the transformation using the INPUT function.

Executing Character-to-Numeric Conversion using INPUT

To perform the conversion, we use the INPUT function, specifying the character variable `day` and an appropriate SAS informat. The statement `numeric_day = input(day, comma9.);` tells SAS to read the character contents of `day` and convert it to a numeric value, treating it according to the `comma9.` informat (although a standard numeric informat like `8.` would also suffice here, `comma9.` is used to demonstrate the syntax structure typically required for complex numeric inputs). We

create a new variable, `numeric_day`, and drop the original character variable `day`.

```
/*create new dataset where 'day' is numeric*/
data new_data;
set original_data;
numeric_day = input(day, comma9.);
drop day;
run;

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

Obs	sales	numeric_day
1	7	1
2	12	2
3	15	3
4	14	4
5	13	5
6	11	6
7	10	7
8	16	8
9	18	9
10	24	10

The final step is to confirm the successful conversion. Running PROC CONTENTS on `new_data` verifies that the new variable, `numeric_day`, has been correctly assigned the numeric data type, thus fulfilling the objective of the conversion.

```
/*display data type for each variable in new dataset*/
proc contents data=new_data;
```

Alphabetic List of Variables and Attributes			
#	Variable	Type	Len
2	numeric_day	Num	8
1	sales	Num	8

The results show that we successfully transformed the original character variable `day` into a numeric variable called `numeric_day`, completing the reverse conversion using the INPUT function.

Summary of PUT and INPUT Function Usage

The distinction between the PUT and INPUT functions hinges entirely on the data type transformation required:

PUT Function: Facilitates conversion **from any type (Numeric or Character) TO Character**. It requires a SAS format to define the output appearance.

INPUT Function: Facilitates conversion **from Character TO any type (Numeric or Character)**. It requires a SAS informat to define how the character string should be interpreted.

These functions are cornerstones of efficient data management within SAS, allowing analysts to maintain data integrity and flexibility regardless of the source data structure. Mastery of these functions is essential for any advanced data manipulation tasks.

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