What are the syntax rules for SAS?

Authored by

stats writer

June 24, 2024

SAS (Statistical Analysis System) is a programming language commonly used for statistical analysis and data management. It follows a specific set of syntax rules that dictate the structure and organization of the code. These rules include using proper punctuation and indentation, following a specific order of statements, and using specific keywords and operators for data manipulation and analysis. The syntax rules for SAS also include guidelines for variable naming, data types, and the use of functions and procedures. Adhering to these syntax rules ensures the proper execution of code and enhances the readability and maintainability of SAS programs.

SAS Syntax

A SAS program is written in the Editor window and contains a series of statements that tell SAS what to do (e.g., import a dataset, give a frequency count of a variable). You can save your program so that it can be edited and reused after it's written.

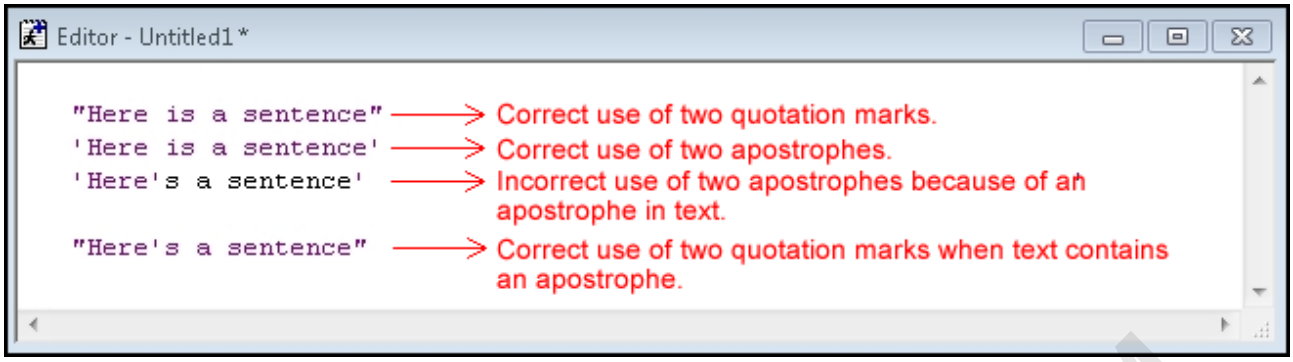
SAS *syntax* is the set of rules that dictate how your program must be written in order for SAS to understand it. There are some conventions of SAS syntax that new users should know before getting started.

Semicolons

Every statement must end with a semicolon. This generally corresponds to every line ending in a semi-colon, but sometimes your commands or statements will be more than one line and a semicolon is only necessary at the end of the statement. Omitting the semicolon is the most common mistake new users make.

Quotation Marks

SAS recognizes text as long as it is enclosed in quotation marks (`"text"`) or apostrophes (`'text'`). It doesn't matter which one you choose, but be sure each text block starts and ends with the same one. You will need to enclose text in quotation marks or apostrophes if you need to reference values of a character variable, reference a file directory, or assign a title to your output, to name a few example. You know you have properly typed text values when SAS changes the color of the words into a purplish-pink color. One word of caution: if your text contains an apostrophe, then you must enclose it with quotation marks. The example text string below shows that SAS recognizes the first two lines as text (indicated by the coloring), but only recognizes part of the text as such in the third line.

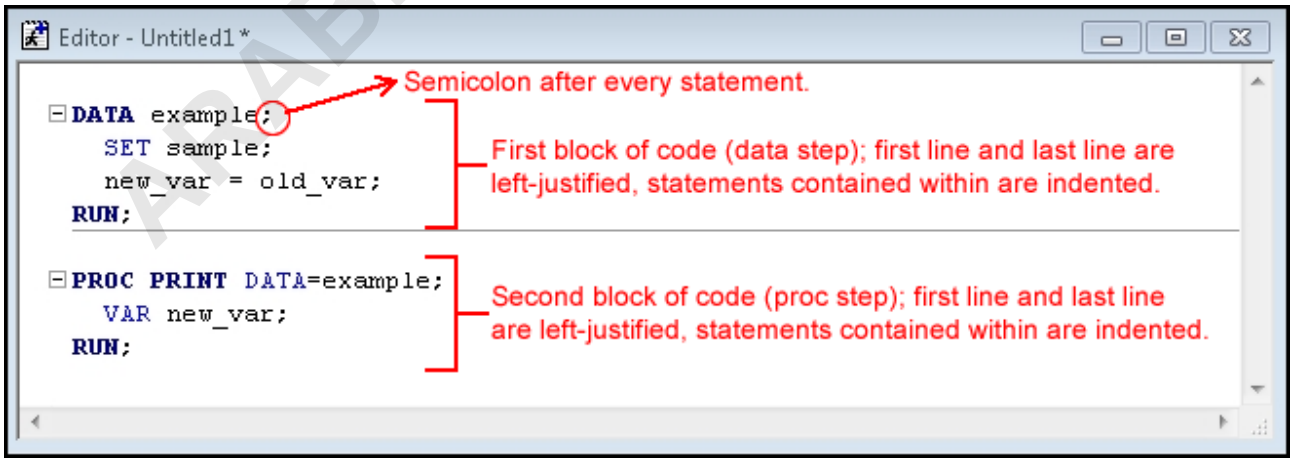


Formatting

SAS is more relaxed than other coding languages when it comes to capitalization, indentation, and line breaks.

SAS is not case sensitive; uppercase and lowercase letters are recognized as the same, even for variable names.Indentations or spacing before a statement are ignored.Extra lines between statements are ignored.Multiple statements on the same line are okay, provided they are separated by a semicolon.A statement can span more than one line, as long as it ends with a semicolon.

Most SAS programmers use capital letters, indentations, and spacing in a way that makes it easier for themselves and other users to read and understand their program. Here is an example of how a typical program would be set up, making use of indentations and spacing, one statement per line, and capital letters.



Comments

A *comment* is a line or block of text that SAS ignores during the execution of a program. Comments make a written program more understandable by documenting what the program is doing (or should be doing), and why. A well-commented program helps you to remember what your thought process was when you first created the program, and helps other users decipher what your program is doing.

There are two ways to "comment out" a line of text or code in a SAS program:

Add an asterisk at the beginning of the line, and add a semicolon at the end of the text being commented out. All text between the asterisk and the semicolon will be commented out.Add a forward slash and an asterisk at the beginning of the comment, and place an asterisk and a forward slash at the end of the line.

An example SAS program containing comments might look like this:

```
* Check the variables in the most recently used dataset using the CONTENTS procedure;
PROC CONTENTS;
RUN;

/* Print the contents of the most recently used dataset using the PRINT procedure.*/
PROC PRINT;
RUN;
```

Note that if you have a comment that spans several lines, it is good practice to place a "start comment" marker at the beginning of each new line.

If you want to comment out one or more lines in your program, highlight the text you want to turn in the comments, then press Ctrl + /. To uncomment those lines, highlight the commented lines and press Ctrl + Shift + /.

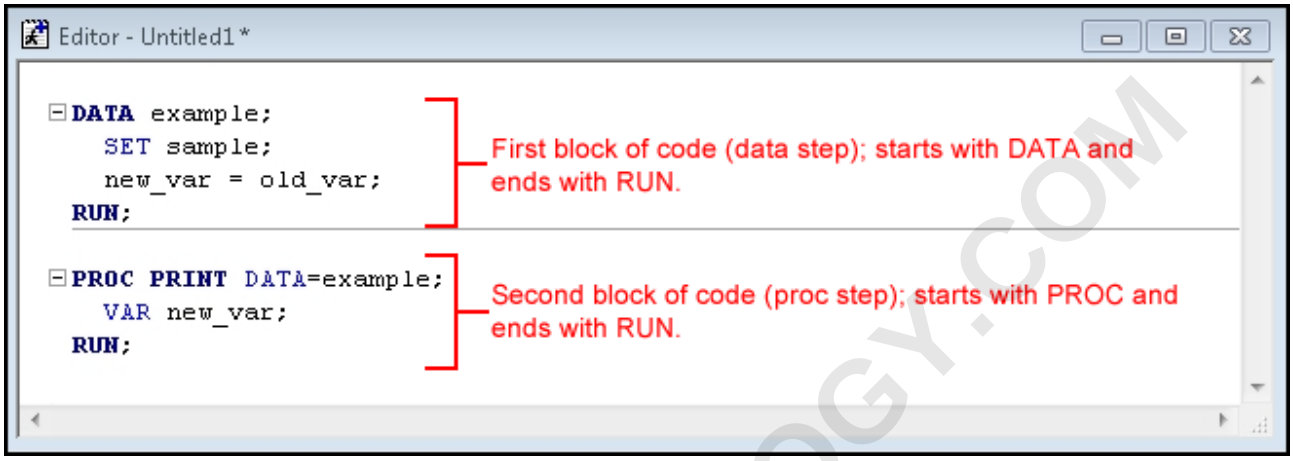
The Data Step vs. The Proc Step

A typical SAS program is organized into blocks of code, called *steps*. Specifically, the *data step* is where data creation and manipulation takes place, and the *proc* (or *procedure*) step is where statistical analysis takes place.

Some statements occur outside of the data step or proc steps; these are called *global statements*. When global statements are executed, their effects are held until the end of the SAS session. One example is the `OPTIONS` statement, which allows the user to customize the headings that print when output is generated (among other things). Another example is the `LIBNAME` statement, which

defines a SAS library.

Every data step begins with the word `DATA` and every proc step begins with the word `PROC`. For the most part, all data and proc steps should end with a `RUN` statement. The `RUN` statement tells SAS to execute the preceding block.



The proc step in the example program above uses the PRINT procedure. PROC PRINT is just one of the many different procedures in SAS. Each procedure has its own keywords, options, and language rules.

In contrast, there is only one kind of data step. The data step always follows the following format:

```
DATA Dataset-Name (OPTIONS);
.
.
.
RUN;
```

In the syntax above, `DATA` is the keyword that starts the data step. Dataset-Nameis the name of the dataset that you want to create or manipulate. If you want to add any of the dataset options, they would go in the parentheses after you name the dataset. In between the first and last lines you put your statements that create and manipulate the dataset. Note the data step ends with a `RUN` statement and a semicolon.

Inside data steps and proc steps are special commands called *statements* and *options* that instruct SAS how to process the data, or specify what analysis, output, or method SAS should use on the data.

Statements are major directions within a data or proc step, that tell SAS *what to do*. Typically, each

statement begins a new line.

Options are smaller components within a data or proc step that modify the behavior of a statement. Specifically, options tell SAS *how* it should do some action. The available options will vary based on the proc step and statement.

There are two statements that can be used in most PROC and data steps that are frequently confused.

A CLASS statement is used in procedures that do inferential statistics or modeling (such as PROC TTEST or PROC GLM) to define which variable(s) should act as categorical independent variables. (It can also be used with the procedure PROC MEANS to compute a single table of means of continuous variables for groups defined by the CLASS variable.)A BY statement is used to subset your data on one or more grouping variables, and then run the procedure on those subsets. Each subset is treated as "independent" of the other subsets - there are no statistical comparisons between the subsets. Any time you use a BY statement, SAS requires that your dataset is first sorted on the BY variable(s). If used with a procedure like PROC MEANS, you will have separate tables for each group defined by the BY variable(s).

Confusing these two statements can lead to wildly different results (especially for inferential statistical tests), so it's important to double-check that you have specified your PROC statements correctly.

Running a Program

After a program is written, click on the running man icon in the toolbar or press F8 on your keyboard to execute your program. Alternatively, if you wish to only run a specific portion of your program, you can highlight the relevant lines of code and then click on the running man (or press F8).

It is a good idea to always check your Log after you run a program to look for warnings or errors.