ABSTRACT
It's the little things that get you! Punctuation is often relegated to less than even a second thought, only to be considered when an error exists. It is our contention that a systematic review of punctuation in SAS programming yields for even experienced users a number of interesting tidbits and valuable efficiencies. In this paper we first briefly identify the common uses of several type of punctuation. Then from the less obvious uses we found during our search, we present the cases most likely to increase efficiency or provide new functionality. Examples of coverage include the when and why of single versus double punctuation marks (e.g., the semi-colon, the period, the question mark, the exclamation mark, the @ sign, and the pipe) including a discussion of macro processing, a review of punctuation usage differences between SQL and traditional SAS code, and some interesting notes on the underscore character.

INTRODUCTION
Programmers may be an odd lot, but even we don't normally sit around pondering punctuation. That is why a systematic review of punctuation used in SAS programming may yield some interesting tidbits. In this paper we make some general observations on punctuation. Then, by punctuation type, we briefly mention obvious uses and give examples of less routine uses where appropriate. Where appropriate, we discuss double punctuation and equivalent uses. We focus on uses applicable to batch programming and leave out uses from the interactive world and specialized procedures such as SAS/ACCESS, AF, and FSP. Also, we have included several special characters that are not generally grouped into the punctuation category.

It sounds funny to refer to "working with punctuation" in the same way that we would "work with lab data" or "work with macros." Nonetheless, before presenting details for each punctuation type, we make two observations about working with punctuation. First, macro quoting functions such as %STR, %NRSTR, %QUOTE, %NRQUOTE, %BQUOTE, %NRBQUOTE, and %SUPERQ are quite useful in that they mask special characters during compilation of a macro. %STR, for example, is useful for treating a semicolon as text rather than part of a statement or when dealing with unbalanced quotation marks or parentheses. An example of this is found when defining a macro variable containing a possessive: %let a=%str(Ed%'s ball);. SAS Help gives the following example showing how to maintain a blank as a blank in counting words: %let word=%qscan(&string,&count,%str( ));. (There is a space within the %str parentheses.)

A second observation regarding working with punctuation involves noting that differences in punctuation usage exist between the SAS SQL procedure and the SAS Data Step. SAS SQL can have efficiencies over the Data Step, but since SQL existed outside of SAS previous to version 6.06 its conventions are different from traditional SAS code. These differences show up even in punctuation usage. The comma is required as a variable name separator in SQL statements such as SELECT and ORDER BY yet it is optional in the corresponding Data Step. And of course the semicolon is widely used in the data step for each statement, but you may see only one semicolon in the whole SQL code if there is only one SELECT or CREATE statement. We now move on to a look at each special character.

AMPERSAND (&)
The ampersand is most readily known as part of a reference to a macro variable. The ampersand substitutes for the logical AND in multiple conditions, for example, if var1=1 & var2='Y';. The ampersand also has a special use in the INPUT statement, indicating that a character value may have one or more single embedded blanks. Two or more ampersands used next to each other indicate that multiple passes will be made to fully resolve the macro variable. For example, if we define

%let indic1=3;
%let indic2=4;
%let drug_3=drug_c;
%let drug_4=drug_d;
then we can make reference to \&\&drug_&indic1 and \&\&drug_&indic2 with multiple passes resolving to suit our needs. After one pass, these become \&drug_3 and \&drug_4, respectively.

ASTERISK (*)
Besides serving as the multiplication operator, the asterisk is found in various places to indicate interactions or cross-classification (such as the TABLES statement in PROC FREQ) and in indicating the number of times to repeat a character string (as in PROC REPORT). It is used to comment statements alone or in conjunction with the slash (/" ... "/) or the percent sign (%*). The asterisk also compels SAS to determine an array subscript by counting the number of variables in the array. Two asterisks indicate the exponentiation operator.

AT SIGN (@)
The at sign is used for column pointer control in the INPUT and PUT statements. As line-hold specifiers, both the single trailing @ and double trailing @ indicate that a data line should be held for another INPUT or PUT statement, even across iterations of the DATA steps.

BLANK ()
No discussion of special characters would be complete without mention of the blank, also known as the space. It is the ultimate separator and functions as a delimiter in place of a comma in many instances. For example, \texttt{if var1 in (2 3 4);} is equivalent to \texttt{if var1 in (2,3,4);}. The blank is the only character that multiple occurrences is universally fine if one occurrence is fine, excepting quoted text.

CARET (^)
The caret is a synonym for the operator NOT and is used in conjunction with the equal sign such that ^= is equivalent to NE to mean "not equal to."

COLON (:)
An excellent review of the colon (Luo, 2001) will not be completely repeated here. Perhaps the simplest feature that is mentioned in this reference has the most widespread use: the variable name wildcard function. \texttt{PROC PRINT; VAR x::;} will print all variables beginning with x. This feature is also useful when dropping, keeping, and summing similar variables. This works with datasets as well and is useful, for example, in deleting temporary datasets in macro calls, for example, \texttt{proc datasets; delete _::;}. The colon is rarely a synonym of a hyphen and functions in the creation of macro variables in SQL.

COMMA (,)
The comma separates arguments in a function, values in the DO and IN statements, and parameters in macro calls. For the IN statement, commas are optional unless used in SQL where they are required. Commas are often used as delimiters in data files (CSV stands for comma separated values). Special informats and formats (e.g., comma7.) exist for inputting and outputting values with commas separating the thousands, for example. Double commas are often seen in ordered macro calls where at least one parameter is left blank to allow the default parameter to be used. Finally, the comma is useful in specifying an irregular iteration pattern in a do loop: \texttt{do i=1 to 4, 7, 9, 20;} and can be used to delimit character values in a do loop such as \texttt{do=‘MON’, ‘WED’, ‘SAT’;}.

EQUAL SIGN (=)
Besides the equality operator, the equal sign is also used to assign values.

EXCLAMATION MARK (!)
The exclamation mark is sometimes a synonym for the pipe (\|) and is sometimes called a bang. It is used in concatenation and as the logical OR. In some instances it is automatically defined as SAS root and is used in path/directory expressions. See PIPE for further details.

HYPHEN (-)
The hyphen is sometimes equivalent to a slash and is sometimes called a dash. Of course it is a minus sign when used as an operator, and can be used to refer to similarly named variables in a sequence (e.g., x1-x8). Note that this feature does not work with dataset names. The hyphen is a prefix operator, in this case used in conjunction with parentheses: \texttt{x=-(&val1)}.

At times one may wish to override a character such as the hyphen where it appears in certain output. The formchar option can to this. An example that changes the default lines used in common PROC REPORT or PROC TABULATE output is options formchar='|_---|+|---+=|-/<>*';
PERIOD (.)
The period is, of course, commonly seen in numeric data as the decimal placeholder, though the convention in many countries is to use the comma for this purpose. It is also the default indicator of missing numeric values, though this can be altered in an options statement. The period must be used when specifying informats and formats and is also used as a delimiter in pathnames. Double periods are often used with macro variables. For example, in proc sort data=&lib..demog; the first period designates the end of the macro variable &lib while the second period separates the library name from the member name demog. Two important variables specified with a period are created automatically in a dataset in conjunction with a by statement: first. ByVariableName and last. ByVariableName. These enable a great deal of useful processing based on an observation being the first or last of a given value for a given variable. And by the way, who said the period could replace the hyphen in separating the area code, prefix, and phone number?!

PIPE (|)
The pipe is sometimes a synonym for the exclamation point. It substitutes for the logical OR as in if var1 IN (1 2 3) | var2 IN (4 5 6);. The double pipe is used in the concatenation of string variables as in var3=trim(var1)||var2;.

PLUS SIGN (+)
The plus sign is the addition operator. Like the hyphen, the plus is used for pointer control in an INPUT and PUT statement. The plus is used when the pointer should be moved forward a relative number of columns. Like the hyphen, the plus sign is also a prefix operator as in x=+(-1). For some reason, this formulation helps when specifying pointer control movement.

POUND SIGN (#)
The pound sign is used for the line pointer control in the INPUT and PUT statements.

QUESTION MARK (?)
The question mark can also be a format modifier used to suppress error messages in the log as the following log illustrates. This is not to be confused with the nofmtterr option which suppresses errors if a requested format is not loaded.

```
1 data test1;
2 rawdate='02/29/2002';
3 date1=input(rawdate, mmddyy10.);
4 run;

NOTE: Invalid argument to function INPUT at line 3 column 9.
rawdate=02/29/2002 date1=. _ERROR_=1 _N_=1
NOTE: Mathematical operations could not be performed at the following places. The results of the operations have been set to missing values.
Each place is given by:
(NumberOf times) at (Line):(Column).
 1 at 3:9
NOTE: The data set WORK.TEST1 has 1 observations and 2 variables.
```

```
5 data test2;
6 rawdate='02/29/2002';
7 date1=input(rawdate, ?mmddyy10.);
8 run;

rawdate=02/29/2002 date1=. _ERROR_=1 _N_=1
NOTE: The data set WORK.TEST2 has 1 observations and 2 variables.
```

```
9 data test3;
10 rawdate='02/29/2002';
11 date1=input(rawdate, ??mmddyy10.);
12 run;

NOTE: The data set WORK.TEST3 has 1 observations and 2 variables.
```

Since 02/29/2002 (February 29, 2002) is not a valid date, SAS cannot convert it into a valid SAS date value. For the first DATA step, SAS printed a note saying ‘Invalid argument to function INPUT’ and set the automatic variable _ERROR_ to 1 in the log file. When a single ? is used in the second DATA step, the ‘Invalid function’
message was suppressed but SAS still set _ERROR_ = 1 in the log file. As we can see the double ?? in the third DATA statement suppressed both the 'Invalid function' message and the setting of _ERROR_ to 1. However, all three data sets set the numeric variable date1 to missing.

**QUOTE (‘ and ”)**
The quote is used to indicate literal (e.g., character or hexadecimal) strings, including empty strings. Double single quotes are used when a single quote is needed in resolution. An example of this is illustrated with the title statement title 'Joe & Bob's BBQ';. In this case, the title resolves to "Joe & Jim's BBQ" and since we used single quotes instead of double quotes to surround the entire title, the ampersand remains unresolved as we desire, since it is part of these fellow's business name.

The double quote is a synonym of the single quote for many uses but also triggers the processing of a macro variable as text. This latter function is found in the following example. A printout would show 5 as the value for both X and X2, but X is a character variable while X2 is a numeric variable.

```sas
%let y=5;
data any;
  x="&y";
  x2=&y;
run;
```

**SEMI-COLON (;)**
Probably no SAS programmer has gone his or her entire career without leaving a semi-colon off the end of a statement! Semicolons are optional at the end of a macro call. Multiple semi-colons are often seen when conditionally processing part of a macro. For example, the first semi-colon at the end of %if condition %then var1=5;; provides the statement-ending semi-colon for what is processed (var1=5) if condition is true while the second semi-colon ends the conditionally processed statement. Another place multiple semi-colons are used is with the cards4 statement; this is necessary when semicolons in data must be processed semi-colons rather than terminating an input line. The semicolon is second only to the blank in multiple occurrences being acceptable if one occurrence is acceptable.

**SLASH (/) and BACKSLASH (\)**
The slash is sometimes equivalent to a hyphen. One obvious use of a slash is to precede an option list in many procedures and of course it is the division operator. It is also the default split character in PROC REPORT and is used to move the pointer to the next line in INPUT and PUT statements. The slash and backslash are both used as directory level separators, depending on the operating system. In order to write platform independent code, the automatic macro variable &SYSSCP can be used to detect the current operating system to determine which separator to use. Finally, the slash is part of a certain kind of comment (/* ... */).

**TILDE (~)**
The tilde is a format modifier in the INPUT statement. It triggers special treatment of single quotation marks, double quotation marks, and delimiters in character values in that delimiters within quoted character values are read as characters instead of delimiters. The tilde is a synonym for the caret in meaning NOT.

**UNDERSCORE (_)**
While some punctuation marks (slash, backslash, period) are used in pathnames or to separate parts of names, the underscore is the only special character which SAS names (variables, data sets, formats, librefs, etc.) may contain. Two names universally reserved across the SAS system contain underscores, _N_ and _ERROR_. Along with the letters A through Z, the underscore is one of 27 characters recognized as special missing values enabling the differentiation among types of missing values.

The underscore is used in default names several places in SAS. For example, if in a procedure you don't specify a dataset where one is expected, the procedure by default assumes the previously used or generated dataset, which is stored in _last_. If for some reason this is not suitable, it can be changed using, for example, options _last_=mylast;. Another example of this is that the underscore is the default variable prefix in PROC TRANSPOSE. And since SAS-created variables begin with an underscore, drop _:; removes all of these variables. Data _null_ is used when data step processing is required but no usual output dataset is desired. Also, %put _automatic_;
displays all automatic variables while _local_ and _user_ have corresponding results. Other examples of automatic variable exist. When defining an array with array question q1-q9;, for example, the _i_ variable is automatically and temporarily assigned as an iteration index for the current data step. And _all_ is created as a variable in data any; set a; if condition then put _all_; yet it is created as a dataset name in proc contents data=fileref._all_;.

Another area of underscores is seen in the following code.

```sas
Array chars _character_;
do over chars;
  if first.suno then chars=' ';end;
Array nums _numeric_;
do over nums;
  if nums=. then nums=0;end;
```

The above two statements containing an underscore merit special mention. The first sets all character variables to blank. This is useful to ensure that the retain statement does not carry over unwanted values. The second changes the value of numeric variables from missing to zero. Another handy use of the _numeric_ feature is found in proc freq; tables _numeric_;. This automatically selects all numeric variables for inclusion into the TABLES statement.

**OTHER SPECIAL CHARACTERS**

Of course there are other keyboard characters that are used in SAS. The dollar sign ($) is used in arrays and in specifying character formats. The percent sign (%) is used in the macro language as the macro keyword symbol. Parentheses ((())), square brackets ([[]]) and curly brackets, {{}} are used in various places with varying levels of equivalency. The less than sign (<) and the greater than sign (>) are inequality operators when used separately, and this is not just for values but also for dates and character values. When used together, << is the minimum operator and <-> is the maximum operator. The equal sign and asterisk used together (=*) is the 'sounds like' operator and can be used in a fuzzy sort of way to catch phonetically similar strings. It is no longer a common keyboard character, but a synonym of the caret and tilde for NOT is found in the optional hyphen (¬). The tick (’), which is not to be confused with the single quote, represents a challenge to SAS users to find a use distinct from the single quote.

**CONCLUSION**

There are hundreds of uses of punctuation and special characters. Along with a few examples, this present partial repository provides impetus to ponder prodigious employment of punctuation. We hope this has been useful. We welcome candidates for a more complete listing.

**REFERENCES**


**ACKNOWLEDGMENTS**

The authors would like to thank Jim Edgington and Indra Fernando for valuable suggestions.

**CONTACT INFORMATION**

John Morrill
Quintiles, Inc.
P.O. Box 9708
Kansas City MO  64134-0708
Phone: (816) 767-6000
E-mail: John.Morrill@Quintiles.com

George Li
Quintiles, Inc.
P.O. Box 9708
Kansas City MO  64134-0708
Phone: (816) 767-6000
E-mail: George.Li@Quintiles.com