SAS® Macros: Top Ten Questions (and Answers!)

Kevin Russell –Technical Support Analyst
SAS Institute Inc.

SAS® Macros: Top Ten Questions (and Answers!)

1. Can I use SAS functions within the macro facility?
2. What quoting function should I use to mask special characters?
3. How do I resolve a macro variable within single quotation marks?
4. How do I resolve error messages when output that was generated with the MPRINT system option looks fine?
5. What are the differences between the autocall facility and the stored compiled macro facility?

SAS® Macros: Top Ten Questions (and Answers!)

6. How do I conditionally execute a macro from within a DATA step?
7. Why does my macro variable not resolve?
8. How can I use macros to loop through all files in a directory?
9. Can I use DATA step variables in a %IF-%THEN statement?
10. Why am I getting an error that states that a character operand was found in the %EVAL function?

Question 1:

Can I use SAS functions within the macro facility?

Answer:

Yes, by using the %SYSFUNC macro function.

Example: %SYSFUNC macro function

```
%put %sysfunc(date(),worddate20.);
```

Output:

```
95 %put %sysfunc(date(),worddate20.);

October 12, 2010
```

Question 2:

What quoting function should I use to mask special characters such as the ampersand, percent sign, parentheses, and quotation marks?

Answer:

Depends. Use compile-time or execution-time quoting functions for your situation.

- **Compile-time quoting functions:**
  - %STR–masks commas, mnemonics, and unmatched quotation marks and parentheses.
  - %NRSTR–masks percent signs and ampersands in addition to the other special characters.

- **Execution-time quoting functions:**
  - %BQUOTE–masks special characters and mnemonics in resolved values during macro execution.
  - %SUPERQ–masks special characters and mnemonics during macro execution. However, it also prevents further resolution of any macros or macro variables.

Example: Unmatched Quotation Mark (‘)

```
%let singleq=O'neill;
%put &singleq;
```

Solution: %STR Function

```
%let singleq=%str(O%'neill)
%put &singleq;
```

```
3120  %put &singleq;
O'neill
```

Example: Percent Sign (%)

```
%let ex=This macro is called %showme;
%put ex=&ex;
```

```
20 %let ex= This macro is called %showme;
WARNING: Apparent invocation of macro SHOWME
not resolved.
21 %put ex=&ex;
WARNING: Apparent invocation of macro SHOWME
not resolved.
```

Solution: %NRSTR Function

```
%let ex=%nrstr(This macro is called %showme);
%put ex=&ex;
```

```
23 %let ex=%nrstr(This macro is called
%showme);
24 %put ex=&ex;
ex=This macro is called %showme
```

Example: Commas (,)

```
%macro test(value);
  %put &value;
%mend;
%let x=a,b,c;
%put WITH NO QUOTING FUNCTION:;
%test(&x);
```

```
31 %put WITH NO QUOTING FUNCTION:;
WITH NO QUOTING FUNCTION:
32 %test(&x);
ERROR: More positional parameters found than
defined.
```

Solution: %BQUOTE Function

```
%put WITH THE CORRECT QUOTING FUNCTION:;
%test(%bquote(&x));
```

```
34 %put WITH THE CORRECT QUOTING
FUNCTION:;
WITH THE CORRECT QUOTING FUNCTION:
35 %test(%bquote(&x))
a,b,c
```

Example: Ampersand (&)

```
data _null_;
   call symputx('Milwaukee','Beer&Brats');
run;
%put NOT quoted:;
%put &Milwaukee;
```

```
5 %put NOT quoted:;
NOT quoted:
6 %put &Milwaukee;
WARNING: Apparent symbolic reference BRATS
not resolved.
Beer&Brats
```

Solution: %SUPERQ Function

```
%put THIS is quoted:;
%put %superq(Milwaukee);
```

```
8 %put THIS is quoted:;
THIS is quoted:
9 %put %superq(Milwaukee);
Beer&Brats
```

Question 3:

How do I resolve a macro variable within single quotation marks?

Answer:

Use the %STR and %UNQUOTE functions.

Example: %STR and %UNQUOTE functions

```
%let name=Fred;
%put %unquote(%str(%'NAME: &name%'));
```

```
17   %put %unquote(%str(%'NAME:&name%'));
'NAME: Fred'
```

Question 4:

How do I resolve error messages when output that was generated with the MPRINT system option looks fine?

Answer:

Use the %UNQUOTE function.

Example: %UNQUOTE function

```
options mprint;
%macro test;
   %let val=aaa;
   %let test = %str(%'&val%');

   data _null_;
      val = &test;
      put val=;
   run;
%mend test;
%test
```

```
Note: Line generated by the macro variable
"TEST".
1 'aaa'
  -
  386
   ---
   202
ERROR 386-185: Expecting an arithmetic
expression.
ERROR 202-322: The option or parameter is not
recognized and will be ignored.
```

Solution: %UNQUOTE Function

```
options mprint;
%macro test;
   %let val = aaa;
   %let test = %unquote(%str(%'&val%'));

   data _null_;
      val = &test;
      put val=;
   run;
%mend test;
%test
```

Question 5:

What are the differences between the autocall facility and the stored compiled macro facility?

Answer:

The differences are in the features for each facility.

Autocall Facility vs. Stored Compiled Macro Facility

Features	Autocall Facility	Stored Compiled Macro Facility
Easy to maintain if code is shared	Yes	Maybe
Easy to hide code to the end user	No	Yes
Uses less overhead	No	Yes
Easy to transfer across platforms	Yes	No

Autocall Facility

- Macro source code is implicitly included and compiled to the WORK.SASMACR catalog.
- To use autocall macros, you have to set the MAUTOSOURCE and the SASAUTOS= system options.
  - The MAUTOSOURCE option activates the autocall facility.
  - The SASAUTOS= option specifies the autocall library or libraries.
- To use the autocall facility, submit the following statements:

```
filename fileref 'autocall-library-path';
options mautosource sasautos=(sasautos fileref);
```

Stored Compiled Macro Facility

- The stored compiled macro facility provides access to permanent SAS catalogs from which you can invoke compiled macros directly.
- To use these stored compiled macros, you have to set the MSTORED and the SASMSTORE= system options.
  - The MSTORED option searches for the compiled macros in a SAS catalog that you reference with the SASMSTORE= option.
  - The SASMSTORE= option specifies the libref for the SAS library that contains the stored compiled macros.
- To use the stored compiled macro facility, submit the following statements:

```
libname libref 'SAS-data-library-path';

options mstored sasmstore=libref;
```

Question 6:

How do I conditionally execute a macro from within a DATA step?

Answer:

Use the CALL EXECUTE routine.

Example: CALL EXECUTE routine

```
   /* Compile the macro BREAK. The BYVAL */
    /* parameter is generated in the CALL */
    /* EXECUTE routine.                   */
%macro break(byval);
data age_&byval;
   set sorted(where=(age=&byval));
run;
%mend;

proc sort data=sashelp.class out=sorted;
   by age;
run;

options mprint;
data _null_;
   set sorted;
   by age;
   if first.age then call
   execute('%break('||age||')');
run;
```

Output:

```
MPRINT(BREAK):   data age_11;
MPRINT(BREAK):   set sorted(where=(age=11));
MPRINT(BREAK):   run;
MPRINT(BREAK):   data age_12;
MPRINT(BREAK):   set sorted(where=(age=12));
MPRINT(BREAK):   run;
MPRINT(BREAK):   data age_13;
MPRINT(BREAK):   set sorted(where=(age=13));
MPRINT(BREAK):   run;
MPRINT(BREAK):   data age_14;
MPRINT(BREAK):   set sorted(where=(age=14));
MPRINT(BREAK):   run;
MPRINT(BREAK):   data age_15;
MPRINT(BREAK):   set sorted(where=(age=15));
MPRINT(BREAK):   run;
MPRINT(BREAK):   data age_16;
MPRINT(BREAK):   set sorted(where=(age=16));
MPRINT(BREAK):   run;
```

Question 7:

Why does my macro variable not resolve?

Answer:

The macro variable might not resolve for a few reasons:

- macro variable scope
- no step boundary
- timing issues

Reasons for Macro Variable Not Resolving:

- **Macro variable scope:** Local macro variables versus global macro variables.
- **No step boundary:** The DATA step does not have an ending RUN statement before a CALL SYMPUT or a CALL SYMPUTX routine.
- **Timing issues:** a result of using the CALL EXECUTE and CALL SYMPUT routines together.

Debugging Options:

- %PUT _USER_;
- %PUT _GLOBAL_;
- %PUT _LOCAL_;
- %PUT _ALL_;
- %PUT _AUTOMATIC_;

Macro Variable Scope

- *Local macro variables* are not available outside of the macros in which they are defined. Such a macro variable only exists as long as its associated macro is executing.

```
1     %macro showme(test);
2        %let inside=RESOLVE ME.;
3        %put this shows INSIDE: &INSIDE;
4     %mend;
5
6      %showme(VALUE)
this shows INSIDE: RESOLVE ME.
7      %put &inside;
WARNING: Apparent symbolic reference INSIDE not resolved.
&inside
```

- *Global macro variables* are available throughout an entire SAS session.

```
%macro showme(test);
   %let inside=RESOLVE ME.;
   %put this shows INSIDE: &INSIDE;
%mend;

 %global inside;
 %showme(VALUE)
 %put &inside;
```

No Step Boundary

Example: No RUN statement before %PUT statement:

```
data test;
   x="MYVALUE";
   call symputx('macvar',x);
%put WILL I RESOLVE &macvar?;
```

Output:

```
57 %put WILL I RESOLVE &macvar?;
WARNING: Apparent symbolic reference MACVAR not
resolved.
```

Example: RUN statement before %PUT statement

```
data test;
   x="MYVALUE";
   call symputx('macvar',x);
run;
%put WILL I RESOLVE &macvar?;
```

Output:

```
14 %put WILL I RESOLVE &macvar?;
WILL I RESOLVE MYVALUE?
```

Timing Issue

```
data names;
   input code name $;
   cards;
1 first
2 second
3 third
   ;
%macro report(munc);
data check;
   set names(where=(code=&munc));
   call symput ('namemac', name);
run;

%if &namemac ne %then %do;
   proc print data=check;
      title "data for &namemac";
   run;
%end;
%mend report;

data _null_ ;
   set names;
   call execute('%report('||code||')');
run;
```

Common Issue with the CALL SYMPUT and CALL EXECUTE Routines

```
data check;
   set names(where=(code=1);
   call symput('namemac',name);
run;
```

Warning and error / stop macro

Input stack

```
%report(1)
```

Timing Issue

```
WARNING: Apparent symbolic reference NAMEMAC not
resolved.
ERROR: A character operand was found in the %EVAL
Function or %IF condition where a numeric operand
is required. The Condition was:&namemac ne
ERROR: The macro REPORT will stop executing.
```

Solution: Use %NRSTR

Bad Code

```
%macro report(munc);
 data check;
    set names(where=(code=&munc));
   call symput ('namemac', name);
 run;
 %if &namemac ne %then %do;
  proc print data=check;
    title "data for &namemac";
  run;
 %end;
%mend report;

data _null_;
  set names;
  call execute('%report('||code||')');

  run;
```

Good Code

```
%macro report(munc);
  data check;
set names(where=(code=&munc));
 call symput ('namemac', name);
  run;
%if &namemac ne %then %do;
  proc print data=check;
    title "data for &namemac";
       run;
    %end;
  %mend report;

data _null_;
  set names;
call execute('%nrstr(%report
('||code||'))');
  run;
```

Question 8:

How can I use macros to loop through all files in a directory?

Answer:

Use the %DO statement.

Example: %DO statement

```
%macro drive(dir,ext);
  %let filrf=mydir;
  %let rc=%sysfunc(filename(filrf,&dir));
  %let did=%sysfunc(dopen(&filrf));
  %let memcnt=%sysfunc(dnum(&did));
     %do i = 1 %to &memcnt;
     %let name=%qscan(%qsysfunc(dread(&did,&i)),-1,.);
%if %qupcase(%qsysfunc(dread(&did,&i))) ne %qupcase(&ext) %then
%do;
     %if (%superq(ext) ne and %qupcase(&name) = %qupcase(&ext))
or
     (%superq(ext) = and %superq(name) ne) %then %do;
     %put %qsysfunc(dread(&did,&i));
         %end;
        %end;
    %end;
   %let rc=%sysfunc(dclose(&did));
%mend drive;
%drive(c:\,sas)
```

Question 9:

Can I use DATA step variables in a %IF-%THEN statement?

Answer:

No, DATA step variables cannot be used in %IF-%THEN statements.

Example: DATA step variable in %IF-%THEN statement

```
%macro test;
      data sample;
         text="OPEN";
         %if TEXT=OPEN %then %do;
            %put TRUE?;
         %end;
      run;
   %mend;
```

%IF-%THEN statement vs. IF-THEN statement

%IF-%THEN Statement

- Conditionally generates text.
- Expression can only contain operands that are constant text or text expressions that generate text.
- Cannot refer to DATA step variables.
- Executes during macro execution. If the statement is contained in a DATA step, it is executed before DATA step compilation.

IF-THEN Statement

- Conditionally executes SAS statements during DATA step execution.
- Expression can only contain operands that are DATA step variables, character or numeric constants, or date and time constants.
- Executes during DATA step execution. If the statement is contained within a macro, it is stored as text.

Question 10:

How do I resolve an error stating that a character operand was found in the %EVAL function?

Answer:

Use the %SYSEVALF function.

Example: %SYSEVALF function

```
%macro test(val);
   %if &val < 0 %then %put test;
%mend test;
%test(-1.2)
```

```
99    %test(-1.2)
ERROR: A character operand was found in the %EVAL
function or %IF condition where a numeric operand is
required. The
condition was: &val > 0
ERROR: The macro TEST will stop executing.
```

Solution:

```
%macro test(val);
      %if %sysevalf(&val < 0) %then %put TRUE;
   %mend test;
   %test(-1.2)
```

```
673   %macro test(val);
674         %if %sysevalf(&val < 0) %then %put TRUE;
675   %mend test;
676   %test(-1.2)
TRUE
```

References

Question 2: What quoting function should I use to mask special characters?

SAS Usage Note 31012, “An error occurs when a comma is present in the value of a parameter being passed to a macro.” 2008. Cary, NC: SAS Institute Inc. Available at support.sas.com/kb/31/012.html.

Question 3: How do I resolve a macro variable within single quotation marks?

SAS Sample 25076, “Resolve a macro variable within single quotes.” 2005. Cary, NC: SAS Institute Inc. Available at support.sas.com/kb/25/076.html.

Question 5: What are the differences between the autocall facility and the stored compiled facility?

SAS Usage Note 445, “How to have shared access with the stored compiled macro facility.” 2005. Cary, NC: SAS Institute Inc. Available at support.sas.com/kb/00/445.html.

SAS Usage Note 23210, “How to hide macro code so that it does not appear in the log when the program is executed.” 2003. Cary, NC: SAS Institute Inc. Available at support.sas.com/kb/23/210.html.

SAS Usage Note 24451, “Creating an autocall macro on a PC.” 2005. Cary, NC: SAS Institute Inc. Available at support.sas.com/kb/24/451.html.

Question 6: How do I conditionally execute a macro from within a DATA step?

SAS Sample 26140, “Creating a new data set for each BY group in a data set.” 2007. Cary, NC: SAS Institute Inc. Available at support.sas.com/kb/26/140.html.

Question 7: Why does my macro variable not resolve?

SAS Usage Note 23134, “Macro variables created with the CALL SYMPUT routine do not resolve when invoked by the CALL EXECUTE routine.” 2003. Cary, NC: SAS Institute Inc. Available at support.sas.com/kb/23/134.html.

Question 8: How can I use macros to loop through all files in a directory?

SAS Sample 25074, “Listing all files that are located in a specific directory.” 2005. Cary, NC: SAS Institute Inc. Available at support.sas.com/kb/25/074.html.

Recommended Reading

Burlew, Michelle. 2006. *SAS Macro Programming Made Easy, Second Edition*. Cary, NC: SAS Press.

Carpenter, Art. 2004. *Carpenter's Complete Guide to the SAS® Macro Language, Second Edition.* Cary, NC: SAS Press.

Tyndall, Russ. *Give Your Macro Code an Extreme Makeover: Tips for even the most seasoned macro programmer* . Cary, NC: SAS Institute Inc. Available at support.sas.com/techsup/technote/ts739.pdf.

Questions?