The George Washington University
Department of Statistics

STAT 4197/6197 – Fall 2019
Week 10 – November 1, 2019

Major Topic: Macro Functions, and Working with Macros

Detailed Topics:

1) Compilation vs. Execution of SAS Programs with Macro Variables
2) Macro Variable Functions
3) Macro Quoting Functions
4) Iterative Processing for Generating SAS Code (Sequential vs. Non-Sequential Processing)
5) Conditional Processing Using Macro Facility

Acknowledgements: Portions of SAS' copyrighted SAS course content are reproduced here with permission of SAS Institute Inc., Cary, NC, USA. SAS Institute Inc. makes no warranties with respect to these materials and disclaims all liability therefor. Parts of additional materials are adapted from SAS Documentation, and web-based resources. Please let me know of any errors or typos you have found. Do not copy or circulate the handouts

References:

1. Relevant Chapters/Sections - Delwiche L, and Slaughter S. *The Little SAS Book: A Primer*, Fifth Edition Paperback – November 7, 2012

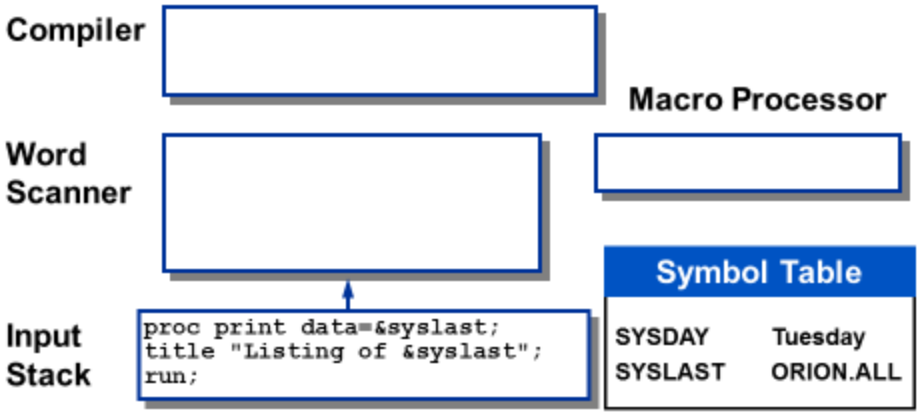
2. Exercises from Relevant Chapters/Sections - Ottesen RA, Delwiche LD, and Slaughter SJ. *Exercises and Projects for The Little SAS Book*, Fifth Edition Paperback – July 1, 2015

3. SAS® 9.4 Macro Language: Reference, Sixth Edition (Online Documentation)

Acknowledgements: Portions of SAS' copyrighted SAS course content are reproduced here with permission of SAS Institute Inc., Cary, NC, USA. SAS Institute Inc. makes no warranties with respect to these materials and disclaims all liability therefor. Parts of additional materials are adapted from SAS Documentation, and web-based resources. Please let me know of any errors or typos you have found. Do not copy or circulate the handouts

Substitution within SAS Code

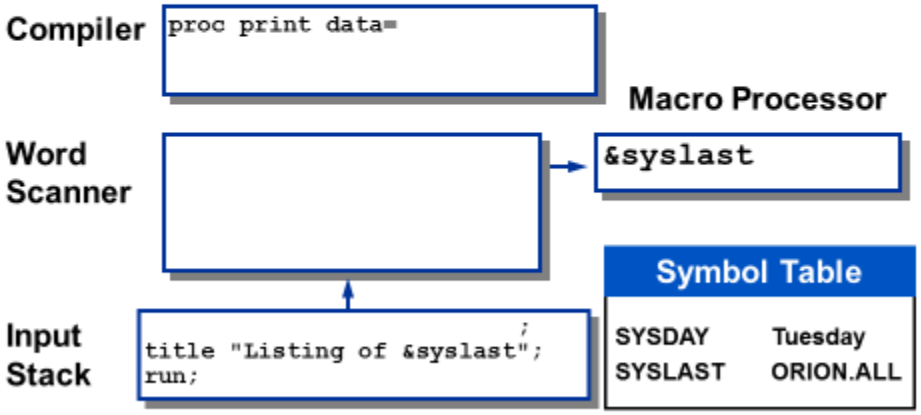
Example: Generalize PROC PRINT to print the last created data set, using the automatic macro variable **SYSLAST**.



51 ...

Substitution within SAS Code

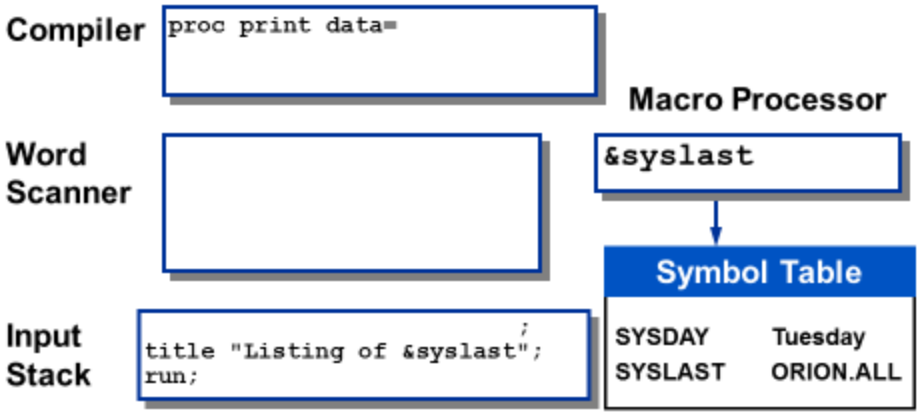
SAS statements are passed to the compiler. When a macro trigger is encountered, it is passed to the macro processor for evaluation.



52 ...

Substitution within SAS Code

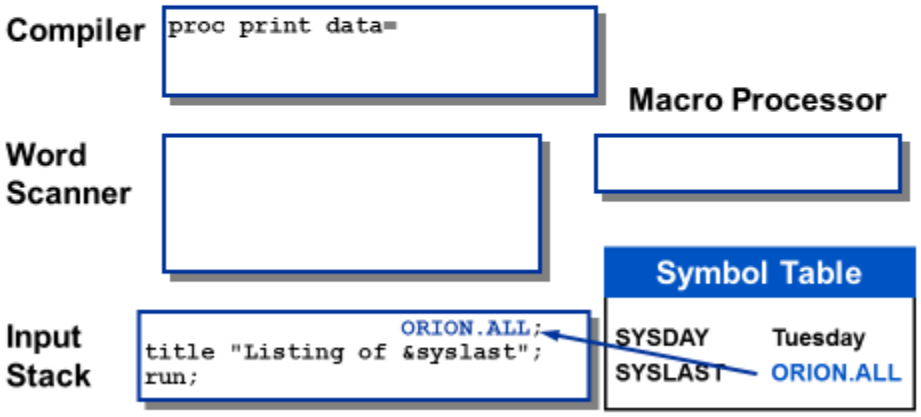
The *macro variable reference* triggers the macro processor to search the symbol table for the reference.



53 ...

Substitution within SAS Code

The macro processor resolves the macro variable reference, passing its resolved value back to the input stack.

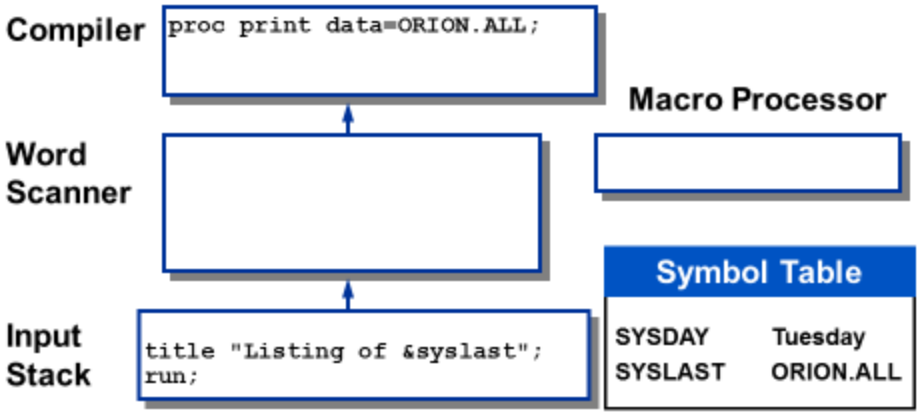


54 ...

Acknowledgements: Portions of SAS' copyrighted SAS course content are reproduced here with permission of SAS Institute Inc., Cary, NC, USA. SAS Institute Inc. makes no warranties with respect to these materials and disclaims all liability therefor. Parts of additional materials are adapted from SAS Documentation, and web-based resources. Please let me know of any errors or typos you have found. Do not copy or circulate the handouts

Substitution within SAS Code

Word scanning continues.

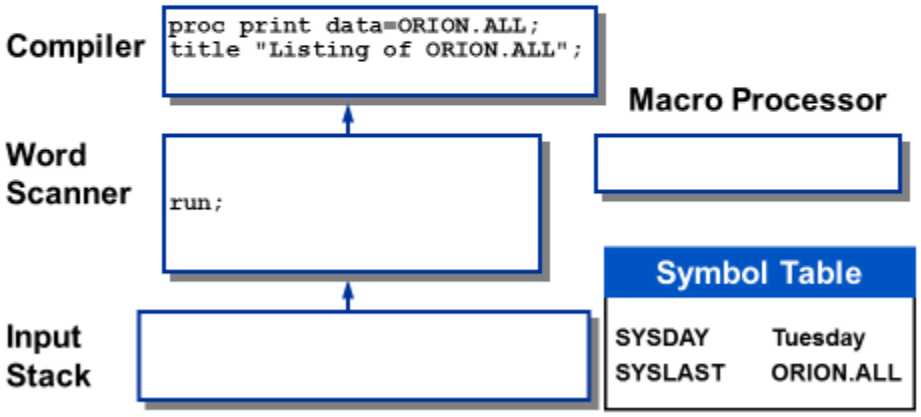


55

...

Substitution within SAS Code

A step boundary is encountered. Compilation ends.
Execution begins.



56

Protecting Function Results

Many macro functions and autocall macros have "Q" equivalents that return masked results.

%CMPRES	%QCMPRES
%LEFT	%QLEFT
%LOWCASE	%QLOWCASE
%SCAN	%QSCAN
%SUBSTR	%QSUBSTR
%SYSFUNC	%QSYSFUNC
%TRIM	%QTRIM
%UPCASE	%QUPCASE

20

Acknowledgements: Portions of SAS' copyrighted SAS course content are reproduced here with permission of SAS Institute Inc., Cary, NC, USA. SAS Institute Inc. makes no warranties with respect to these materials and disclaims all liability therefor. Parts of additional materials are adapted from SAS Documentation, and web-based resources. Please let me know of any errors or typos you have found. Do not copy or circulate the handouts

Macro Functions

Macro functions manipulate macro variables.

SAS function
substr(date,6)

Macro function
%substr(&sysdate9,6)

- Macro functions are executed by the macro processor.

121

Macro Functions

Selected character string manipulation functions:

%UPCASE	Translates letters from lowercase to uppercase.
%SUBSTR	Extracts a substring from a character string.
%SCAN	Extracts a word from a character string.
%INDEX	Searches a character string for specified text.

Other macro functions:

%EVAL	Performs arithmetic and logical operations.
%SYSFUNC	Executes SAS functions.
%STR	Masks special characters.
%NRSTR	Masks special characters, including macro triggers.

122

Acknowledgements: Portions of SAS' copyrighted SAS course content are reproduced here with permission of SAS Institute Inc., Cary, NC, USA. SAS Institute Inc. makes no warranties with respect to these materials and disclaims all liability therefor. Parts of additional materials are adapted from SAS Documentation, and web-based resources. Please let me know of any errors or typos you have found. Do not copy or circulate the handouts

%SUBSTR Function

Use the %SUBSTR function to extract the year portion of the **&SYSDATE9** macro variable.

```
17    %put &=sysdate9;
SYSDATE9=20APR2011
18    %put YEAR=%substr(&sysdate9,6);
YEAR=2011
```

%SUBSTR(*argument*, *position* <,*n*>)

- The %SUBSTR function returns the portion of *argument* beginning at *position* for a length of *n* characters.
- When *n* is not supplied, the %SUBSTR function returns the portion of *argument* beginning at *position* to the end of *argument*.

123

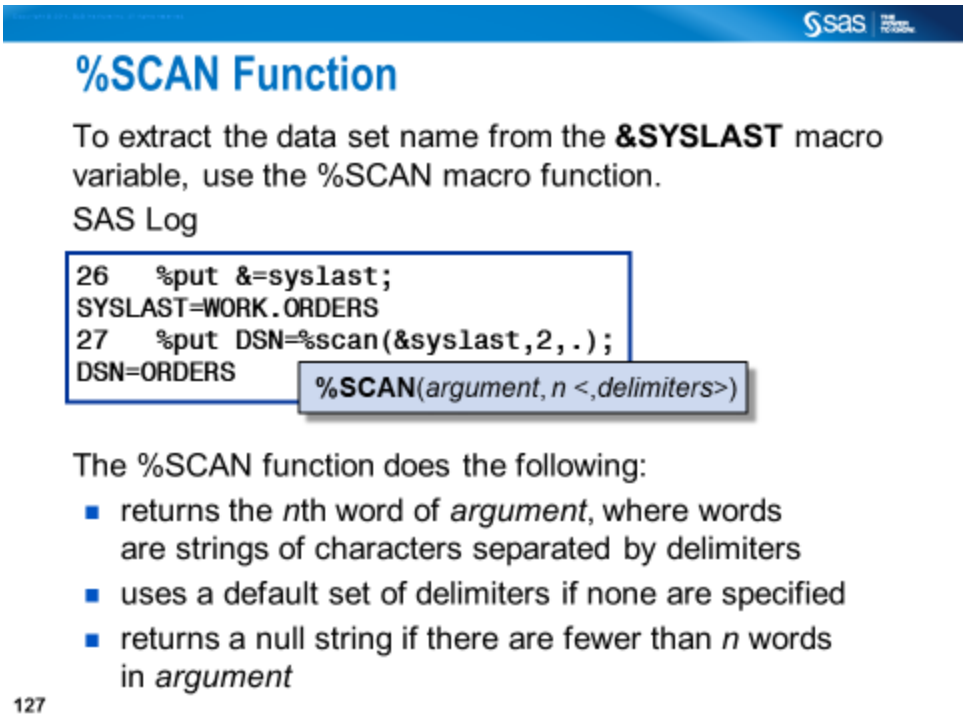
Macro Functions

Arguments to macro string manipulation functions can be any text or macro triggers, or both, including:

Constant text	%put %upcase(angel);	ANGEL
Macro variable references	%let name=angel; %put %upcase(&name);	ANGEL
Macro functions	%put %upcase(%substr(&name,1,1));	A
Macro calls	%put %upcase(%mymacro);	unknown

Constant text arguments do not require quotation marks.

124



Acknowledgements: Portions of SAS' copyrighted SAS course content are reproduced here with permission of SAS Institute Inc., Cary, NC, USA. SAS Institute Inc. makes no warranties with respect to these materials and disclaims all liability therefor. Parts of additional materials are adapted from SAS Documentation, and web-based resources. Please let me know of any errors or typos you have found. Do not copy or circulate the handouts

Business Scenario

Additional macro functions are available to automate the generation of SAS code.

130

%EVAL Function

The %EVAL function performs arithmetic and logical operations.

```
28    %put x=2+2;
x=2+2
29    %put x=%eval(2+2);
x=4
```

%EVAL(*expression*)

The %EVAL function does the following:

- truncates non-integer results
- returns a text result
- returns 1 (true) or 0 (false) for logical operations
- returns a null value and error message when non-integer values are used in arithmetic operations

131

Acknowledgements: Portions of SAS' copyrighted SAS course content are reproduced here with permission of SAS Institute Inc., Cary, NC, USA. SAS Institute Inc. makes no warranties with respect to these materials and disclaims all liability therefor. Parts of additional materials are adapted from SAS Documentation, and web-based resources. Please let me know of any errors or typos you have found. Do not copy or circulate the handouts

%EVAL Function

Example: Compute the first year of a range based on the current date.

```
%let thisyr=%substr(&sysdate9,6);
%let lastyr=%eval(&thisyr-1);
proc means data=orion.order_fact maxdec=2 min max mean;
   class order_type;
   var total_retail_price;
   where year(order_date) between &lastyr and &thisyr;
   title1 "Orders for &lastyr and &thisyr";
   title2 "(as of &sysdate9)";
run;
```

Orders for 2010 and 2011
(as of 31DEC2011)

The MEANS Procedure

Analysis Variable : Total Retail Price for This Product

Order Type	N Obs	Minimum	Maximum	Mean
1	174	3.20	1136.20	126.74
2	62	6.20	1937.20	212.88
3	55	9.60	702.00	172.10

132

m102d06

Acknowledgements: Portions of SAS' copyrighted SAS course content are reproduced here with permission of SAS Institute Inc., Cary, NC, USA. SAS Institute Inc. makes no warranties with respect to these materials and disclaims all liability therefor. Parts of additional materials are adapted from SAS Documentation, and web-based resources. Please let me know of any errors or typos you have found. Do not copy or circulate the handouts

%SYSFUNC Function

The %SYSFUNC macro function executes SAS functions.

```
31    %put &=syslast;
SYSLAST=WORK.ORDERS
32    %put DSN=%sysfunc(propcase(&syslast));
DSN=Work.Orders
```

%SYSFUNC(*SAS function(argument(s))* <,*format*>)

- *SAS function(argument(s))* is the name of a SAS function and its corresponding arguments.
- The second argument is an optional format for the value returned by the first argument.

133

%SYSFUNC Function

Example: Generate titles that contain the current date and time, appropriately formatted.

```
title1 "%sysfunc(today(),weekdate.)";
title2 "%sysfunc(time(),timeAMPM8.)";
```

```
Monday, March 7, 2011
        1:39 PM
```

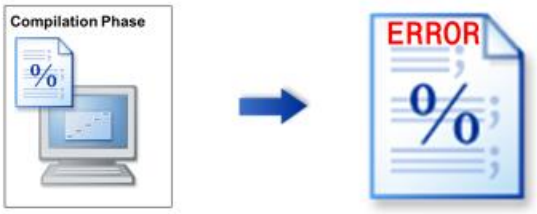
m102d07

135

Acknowledgements: Portions of SAS' copyrighted SAS course content are reproduced here with permission of SAS Institute Inc., Cary, NC, USA. SAS Institute Inc. makes no warranties with respect to these materials and disclaims all liability therefor. Parts of additional materials are adapted from SAS Documentation, and web-based resources. Please let me know of any errors or typos you have found. Do not copy or circulate the handouts

Business Scenario

Macro programs often contain special characters, which can be misinterpreted during macro compilation.



4

Macro Quoting Functions

Special Characters and Macro Quoting Functions by Group and Item
Summary of Macro Quoting Functions and the Characters That They Mask - Online SAS(R) Documentation)

Group	Items	Macro Quoting Functions
A	+ — */<>=¬^|~;, # blank AND OR NOT EQ NE LE LT GE GT IN	All
B	& %	%NRSTR, %NRBQUOTE, %SUPERQ, %NRQUOTE
C	Unmatched: ‘ “ ( )	%BQUOTE, %NRBQUOTE, %SUPERQ, %STR*, %NRSTR*, %QUOTE*, %NRQUOTE*
*Unmatched quotation marks and parentheses must be marked with a percent sign (%) when used with %STR, %NRSTR, %QUOTE, and %NRQUOTE.		

Macro Quoting Functions That Work at Compilation vs. Execution by Group
Summary of Macro Quoting Functions and the Characters That They Mask - Online SAS(R) Documentation)

Function	Works at	Affects Groups	
%STR	Macro compilation	A, C*	
%NRSTR	Macro compilation	A, B, C*	
%BQUOTE	Macro execution	A, C	
%NRBQUOTE	Macro execution	A, B, C	
%SUPERQ	Macro execution (prevents resolution)	A, B, C	
%QUOTE	Macro execution. Requires unmatched quotation marks and parentheses to be marked with a percent sign (%).	A, C*	
%NRQUOTE	Macro execution. Requires unmatched quotation marks and parentheses to be marked with a percent sign (%).	A, B, C*	
*Unmatched quotation marks and parentheses must be marked with a percent sign (%) when used with %STR, %NRSTR, %QUOTE, and %NRQUOTE.			

Acknowledgements: Portions of SAS’ copyrighted SAS course content are reproduced here with permission of SAS Institute Inc., Cary, NC, USA. SAS Institute Inc. makes no warranties with respect to these materials and disclaims all liability therefor. Parts of additional materials are adapted from SAS Documentation, and web-based resources. Please let me know of any errors or typos you have found. Do not copy or circulate the handouts

%STR Function

The %STR function *masks* (removes the normal meaning of) these special tokens:

Special Characters	+ - * / , < > = ; ' " ~ ^ ( ) # blank
Mnemonics	LT EQ GT LE GE NE AND OR NOT IN

139

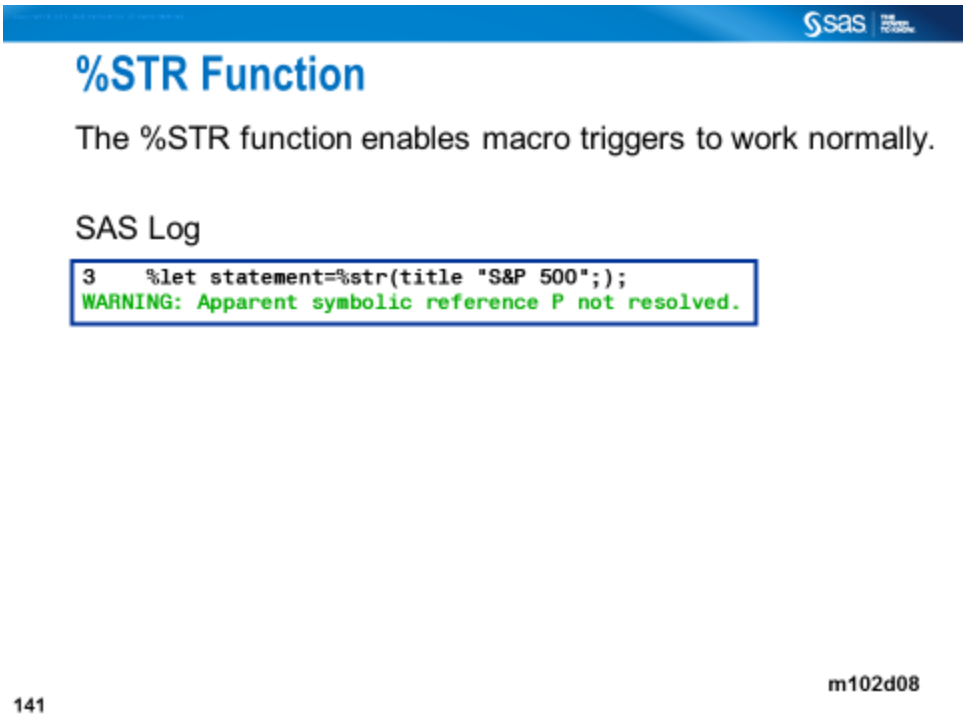
%STR Function

The following are true for the %STR function:

- masks tokens, so the macro processor does not interpret them as macro-level syntax
- enables macro triggers to work normally
- preserves leading and trailing blanks in its argument
- masks an unpaired quotation mark or parenthesis in its argument when the quotation mark or parenthesis is immediately preceded by a percent sign (%)

140

Acknowledgements: Portions of SAS' copyrighted SAS course content are reproduced here with permission of SAS Institute Inc., Cary, NC, USA. SAS Institute Inc. makes no warranties with respect to these materials and disclaims all liability therefor. Parts of additional materials are adapted from SAS Documentation, and web-based resources. Please let me know of any errors or typos you have found. Do not copy or circulate the handouts



Acknowledgements: Portions of SAS' copyrighted SAS course content are reproduced here with permission of SAS Institute Inc., Cary, NC, USA. SAS Institute Inc. makes no warranties with respect to these materials and disclaims all liability therefor. Parts of additional materials are adapted from SAS Documentation, and web-based resources. Please let me know of any errors or typos you have found. Do not copy or circulate the handouts

%NRSTR Function

The %NRSTR function works the same as the %STR function, except it also masks macro triggers.

Example: Use %NRSTR to prevent attempted macro variable resolution.

```
%let statement=%nrstr(title "S&P 500";);
```

%NRSTR(argument)

m102d08

142

3.01 Quiz – Correct Answer

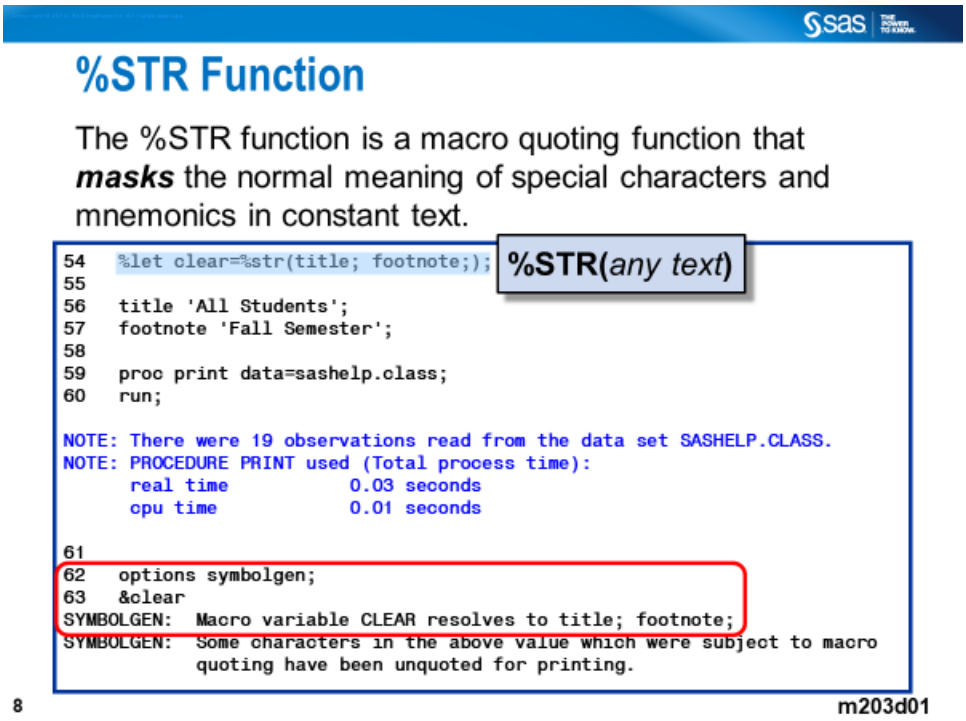
Create a macro variable that, when referenced, clears titles and footnotes.

```
%let clear=title; footnote;
```

What is the problem here?

The first semicolon ends the %LET statement. The macro variable CLEAR stores the text *title* only, without the semicolon.

7



Acknowledgements: Portions of SAS' copyrighted SAS course content are reproduced here with permission of SAS Institute Inc., Cary, NC, USA. SAS Institute Inc. makes no warranties with respect to these materials and disclaims all liability therefor. Parts of additional materials are adapted from SAS Documentation, and web-based resources. Please let me know of any errors or typos you have found. Do not copy or circulate the handouts

%STR Function

The %STR function

- does ***not*** mask macro triggers:
  & %
- requires additional syntax to mask special characters that normally come in pairs:
  " ' ( )

9

Acknowledgements: Portions of SAS' copyrighted SAS course content are reproduced here with permission of SAS Institute Inc., Cary, NC, USA. SAS Institute Inc. makes no warranties with respect to these materials and disclaims all liability therefor. Parts of additional materials are adapted from SAS Documentation, and web-based resources. Please let me know of any errors or typos you have found. Do not copy or circulate the handouts

Protecting Commas

The %STR function masks the comma.

```
%macro name(fullname);
   %let first=%scan(&fullname,2);
   %let last=%scan(&fullname,1);
   %let newname=&first &last;
   %put &newname;
%mend name;

%name(%str(Taylor, Jenna))
```

```
Jenna Taylor
```

m203d02a

12

Protecting Commas

To specify a comma delimiter, use the %STR function.

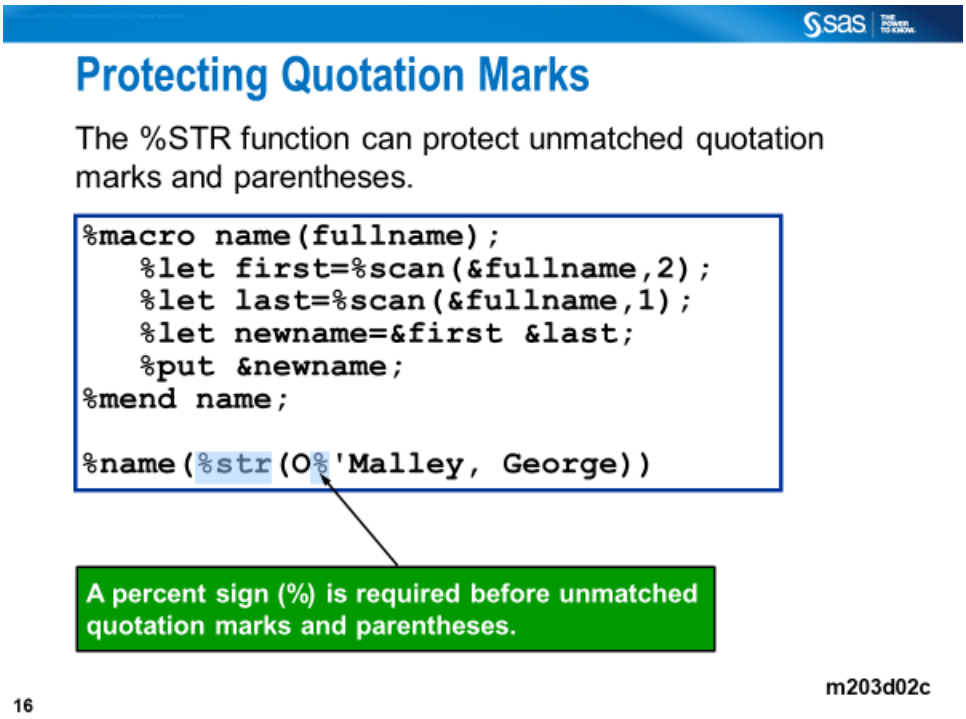
```
%macro name(fullname);
   %let first=%scan(&fullname,2,%str(,));
   %let last=%scan(&fullname,1,%str(,));
   %let newname=&first &last;
   %put &newname;
%mend name;

%name(%str(Taylor, Jenna))
```

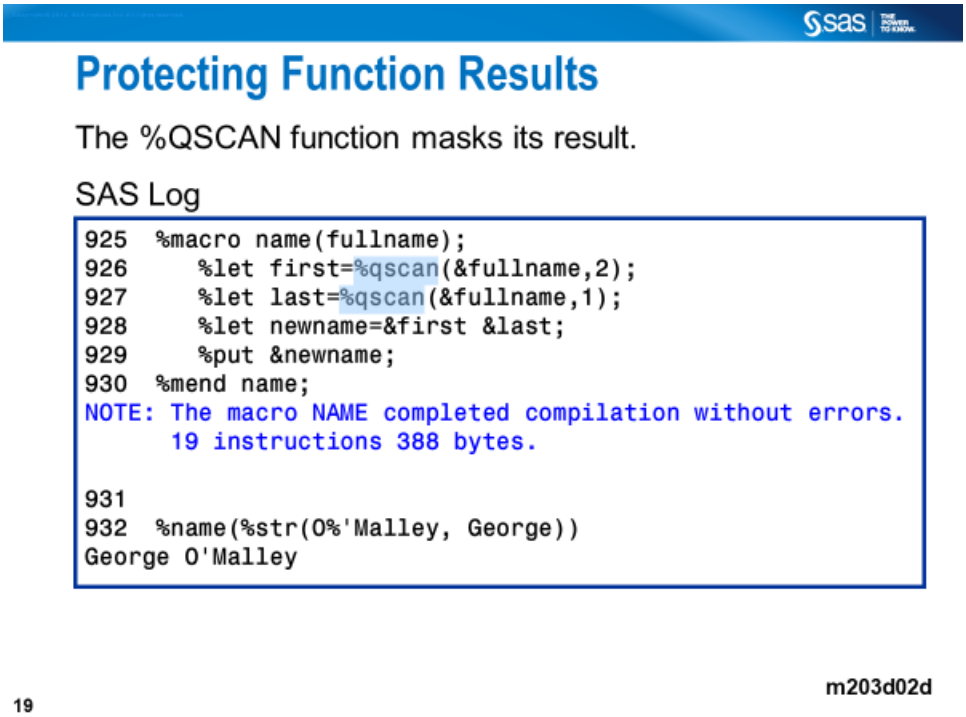
```
Jenna Taylor
```

m203d02b

13

Acknowledgements: Portions of SAS' copyrighted SAS course content are reproduced here with permission of SAS Institute Inc., Cary, NC, USA. SAS Institute Inc. makes no warranties with respect to these materials and disclaims all liability therefor. Parts of additional materials are adapted from SAS Documentation, and web-based resources. Please let me know of any errors or typos you have found. Do not copy or circulate the handouts





Acknowledgements: Portions of SAS' copyrighted SAS course content are reproduced here with permission of SAS Institute Inc., Cary, NC, USA. SAS Institute Inc. makes no warranties with respect to these materials and disclaims all liability therefor. Parts of additional materials are adapted from SAS Documentation, and web-based resources. Please let me know of any errors or typos you have found. Do not copy or circulate the handouts

Business Scenario

Macro programs often contain special characters, which can be misinterpreted during macro execution.

35

3.07 Quiz – Correct Answer

SAS Log

The text **OR** was misinterpreted as a logical operator.

```
1042  %macro where(state);
1043     %if &state=NC %then %put Southeast;
1044     %else %if &state=OR %then %put Northwest;
1045     %else %put Unknown;
1046  %mend where;
NOTE: The macro WHERE completed compilation without errors.
      21 instructions 404 bytes.
1047
1048  %where(NY)
ERROR: A character operand was found in the %EVAL function or %IF
       condition where a numeric operand is required. The condition
       was: &state=OR
ERROR: The macro WHERE will stop executing.
```

m203d04a

37

Acknowledgements: Portions of SAS' copyrighted SAS course content are reproduced here with permission of SAS Institute Inc., Cary, NC, USA. SAS Institute Inc. makes no warranties with respect to these materials and disclaims all liability therefor. Parts of additional materials are adapted from SAS Documentation, and web-based resources. Please let me know of any errors or typos you have found. Do not copy or circulate the handouts

Protecting Constant Text

The %STR function masks mnemonics in constant text.

SAS Log

```
1049  %macro where(state);
1050     %if &state=NC %then %put Southeast;
1051     %else %if &state=%str(OR) %then %put Northwest;
1052     %else %put Unknown;
1053  %mend where;
NOTE: The macro WHERE completed compilation without errors.
      21 instructions 404 bytes.
1054
1055  %where(NY)
Unknown
```

m203d04b

38

Acknowledgements: Portions of SAS' copyrighted SAS course content are reproduced here with permission of SAS Institute Inc., Cary, NC, USA. SAS Institute Inc. makes no warranties with respect to these materials and disclaims all liability therefor. Parts of additional materials are adapted from SAS Documentation, and web-based resources. Please let me know of any errors or typos you have found. Do not copy or circulate the handouts

Protecting Resolved Values

The %STR function masked the resolved value of **state**.

```
1063  %macro where(state);
1064     %if %str(&state)=NC %then %put Southeast;
1065     %else %if %str(&state)=%str(OR) %then %put Northwest;
1066     %else %put Unknown;
1067  %mend where;
NOTE: The macro WHERE completed compilation without errors.
      21 instructions 408 bytes.
1068
1069  %where(OR)
Northwest
```

The %STR function is

- ***not*** recommended for masking resolved values
- recommended for masking ***constant text*** only.

m203d04c

42

%NRSTR Function

The %NRSTR function is similar to the %STR function, but %NRSTR also masks macro triggers.

```
%let company=%nrstr(AT&T);
```

m203d03

29

Acknowledgements: Portions of SAS' copyrighted SAS course content are reproduced here with permission of SAS Institute Inc., Cary, NC, USA. SAS Institute Inc. makes no warranties with respect to these materials and disclaims all liability therefor. Parts of additional materials are adapted from SAS Documentation, and web-based resources. Please let me know of any errors or typos you have found. Do not copy or circulate the handouts

%SUPERQ Function

The %SUPERQ function masks all special characters and mnemonics, including macro triggers, during execution.

```
1071  %macro where(state);
1072     %if %superq(state)=NC %then %put Southeast;
1073     %else %if %superq(state)=%str(OR) %then %put Northwest;
1074     %else %put Unknown;
1075  %mend where;
NOTE: The macro WHERE completed compilation without errors.
      21 instructions 420 bytes.
1076
1077  %where(OR)
Northwest
```

%SUPERQ(*macro-variable*)

macro-variable is a single macro variable name or an expression that resolves to a single macro variable name.

- Do not precede the argument to %SUPERQ with an ampersand (&).

43 m203d04d

%SUPERQ Function

The %SUPERQ function masks the resolved value of **name**.

SAS Log

```
1093  %let name=Taylor, Jenna;
1094  %let initial=%substr(%superq(name),1,1);
1095  %put &=initial;
INITIAL=T
```

- This form of the %PUT statement was introduced in SAS 9.3.

47 m203d05

Acknowledgements: Portions of SAS' copyrighted SAS course content are reproduced here with permission of SAS Institute Inc., Cary, NC, USA. SAS Institute Inc. makes no warranties with respect to these materials and disclaims all liability therefor. Parts of additional materials are adapted from SAS Documentation, and web-based resources. Please let me know of any errors or typos you have found. Do not copy or circulate the handouts

Tip: Protecting Resolved Values

Execution-time quoting functions protect resolved values.

49

Summary: Macro Quoting Functions

	Macro Triggers & % Not Masked	Macro Triggers & % Masked
Constant text (compile time)	%STR	%NRSTR
Resolved values (execution time)		%SUPERQ

50

Acknowledgements: Portions of SAS' copyrighted SAS course content are reproduced here with permission of SAS Institute Inc., Cary, NC, USA. SAS Institute Inc. makes no warranties with respect to these materials and disclaims all liability therefor. Parts of additional materials are adapted from SAS Documentation, and web-based resources. Please let me know of any errors or typos you have found. Do not copy or circulate the handouts

Summary: Macro Quoting Functions

	Macro Triggers & % Not Masked	Macro Triggers & % Masked
Constant text (compile time)	%STR	%NRSTR
Resolved values (execution time)	%BQUOTE	%SUPERQ

59

Acknowledgements: Portions of SAS' copyrighted SAS course content are reproduced here with permission of SAS Institute Inc., Cary, NC, USA. SAS Institute Inc. makes no warranties with respect to these materials and disclaims all liability therefor. Parts of additional materials are adapted from SAS Documentation, and web-based resources. Please let me know of any errors or typos you have found. Do not copy or circulate the handouts

Business Scenario

Macro applications might require iterative processing.

The iterative %DO statement can execute macro language statements and generate SAS code.

46

Simple Loops

Example: Display a series of macro variables in the SAS log by repeatedly executing %PUT within a macro loop.

```
%macro putloop;
   %do i=1 %to &numrows;
      %put Country&i is &&country&i;
   %end;
%mend;
```

%DO *index-variable=start* **%TO** *stop* <**%BY** *increment*>;
text
%END;

SAS Log

```
200  %putloop
Country1 is Australia
Country2 is Canada
Country3 is Germany
Country4 is Israel
Country5 is Turkey
Country6 is United States
Country7 is South Africa
```

48 m105d07

Simple Loops

%DO *index-variable=start* **%TO** *stop* <**%BY** *increment*>;
text
%END;

- %DO and %END are valid only inside a macro definition.
- *index-variable* is a macro variable created in the local symbol table if it does not already exist in another symbol table.
- *start*, *stop*, and *increment* values can be any valid macro expressions that resolve to integers.
- The %BY clause is optional. (The default *increment* is 1.)
- *text* can be any of the following:
  - constant text
  - macro variables or expressions
  - macro statements
  - macro calls

49

Acknowledgements: Portions of SAS' copyrighted SAS course content are reproduced here with permission of SAS Institute Inc., Cary, NC, USA. SAS Institute Inc. makes no warranties with respect to these materials and disclaims all liability therefor. Parts of additional materials are adapted from SAS Documentation, and web-based resources. Please let me know of any errors or typos you have found. Do not copy or circulate the handouts

Generating SAS Code Iteratively with a Macro

Goals:

- To create 8 separate data sets, each containing 1 observation and 2 variables, using a macro with %DO-%END statements.
- To concatenate the data sets (DATA step - non-macro code).
- To get the listing of 8 observations (PROC step – non-macro code)

```
 1  *Ex22_Percent_DoLoop_Macro.sas;
 2  %macro runit (first=, last=);
 3        %local yr;
 4        %do yr=&first %to &last;
 5          data have_20%sysfunc(putn(&yr,z2.));
 6                exp=20%sysfunc(putn(&yr,z2.))/4;
 7              year=20%sysfunc(putn(&yr,z2.));
 8           run;
 9         %end ;
10  %mend runit;
11  %runit(first=08, last=15);
12
13 data all_yrs;
14   retain year;
15    set Have:;
16  run;
17 proc print data=all_yrs noobs split='*';
18  label year= 'Survey Year'
19        exp= 'Mean expenses*per person per year';
20  run;
```

Lines 2-10: Define the macro.
Line 3: Declare index variable of the macro loop as a local macro variable.
Line 2: Specify the keyword parameters in the %macro statement.
Line 11: Invoke the macro to create 8 separate data sets.
Line 13-16: Concatenate them.
Line 17-20: Generate listing.

Here is an example of data-driven dynamic programming that uses the iterative %DO loop within a SAS macro to do the following:

- increment an index macro variable whose value ranges from 2005 to 2015
- produce SAS statements for reading raw data from six data files yob2005 through yob2015 that are stored in a zipped file

```
1  *Ex22_Macro_Read_Zipped.sas;
2  Options nocenter nodate nonumber symbolgen;
3  %Let Path = c:\SASCourse\Week9;
4  Libname mylib "&Path";
5  %macro readraw (first=, last=);
6    Filename ZIPFILE SASZIPAM "&Path\names.zip";
7     %do year=&first %to &last;
8       DATA mylib.DSN&Year;
9          INFILE ZIPFILE(yob&year..txt) DLM=',';
10         INPUT name $ gender $ number;
11      RUN;
12      title "Listing from Data Set (Newborns' Names) for &Year";
13         proc print data=mylib.DSN&Year(obs=5) noobs;
14         format number comma7.;
15         run;
16     %end ;
17 %mend readraw;
18 %readraw(first=2000, last=2005)
```

1. Macro %Do %End Processing with a Non-Sequential List of Values of a Macro Variable
    - Macro call with no parameter list
    - Macro call with keyword parameter
2. Macro with a series of macro variables (common prefix with a numeric suffix)
3. Macro with a list of values of a macro Variable (No keyword or positional parameters)

Acknowledgements: Portions of SAS' copyrighted SAS course content are reproduced here with permission of SAS Institute Inc., Cary, NC, USA. SAS Institute Inc. makes no warranties with respect to these materials and disclaims all liability therefor. Parts of additional materials are adapted from SAS Documentation, and web-based resources. Please let me know of any errors or typos you have found. Do not copy or circulate the handouts

%Do-%End Processing with a Non-Sequential List of Values of a Macro Variable

```
1  *Ex7_%DO_Nonsequential1.sas;
2  options nonumber nocenter nodate symbolgen;
3  %LET list = %str(sashelp.class|
4                   sashelp.iris|
5                   sashelp.retail);
6  /* Count # of values in the string */
7  %LET count=%sysfunc(countw(&list, %STR(|)));
8  %macro loop;
9   /* Loop through the total # of data sets */
10  %local i;
11  %do i = 1 %to &count;
12    title  "%left(%unquote(%SCAN(&list, &i, %STR(|))))";
13    proc print data=%scan(&list,&i, %str(|))
14                (obs=5) noobs;
15     run;
16  %end;
17 %mend loop;
18 %loop
```

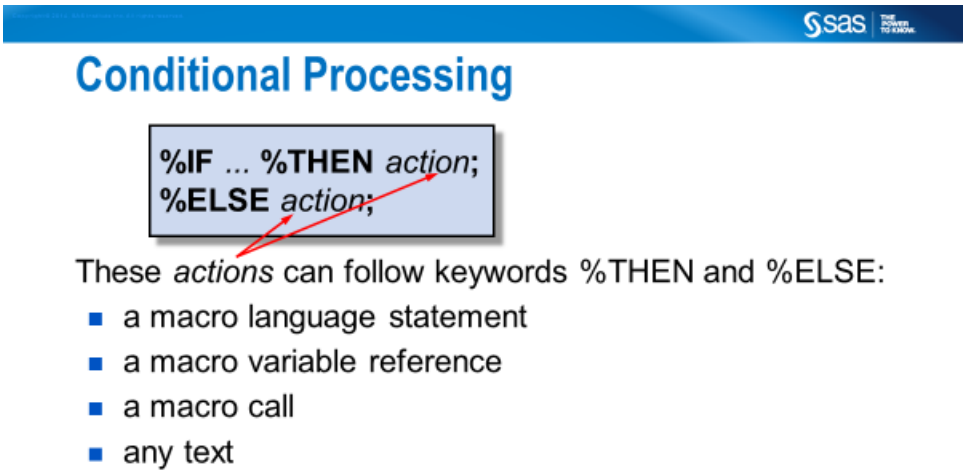
Acknowledgements: Portions of SAS' copyrighted SAS course content are reproduced here with permission of SAS Institute Inc., Cary, NC, USA. SAS Institute Inc. makes no warranties with respect to these materials and disclaims all liability therefor. Parts of additional materials are adapted from SAS Documentation, and web-based resources. Please let me know of any errors or typos you have found. Do not copy or circulate the handouts

```
1  *Ex8_%DO_Nonsequential2.sas;
2  options nocenter nodate nonumber symbolgen;
3  %macro loop(dslist);
4        %local xcount i;
5        /* Count the # of values in the string */
6        %let xcount=%sysfunc(countw(&dslist, %STR(|)));
7        /* Loop through the total # of data sets */
8        %do i = 1 %to &xcount;
9           title  "%left(%scan(&dslist,&i,%str(|)))";
10          proc print data=%scan(&dslist,&i,%str(|))
11                    (obs=5) noobs;
12       run;
13       %end;
14 %mend loop;
15 %loop(%str(sashelp.class|sashelp.iris|sashelp.retail))
```

```
1  *Ex14_VList_HList.sas (Part 1);
2  options nodate nonumber symbolgen;
3  %macro VList;
4  %local ds1 ds2 ds3 j;
5  %let ds1 = class;
6  %let ds2 = revhub2;
7  %let ds3 = iris;
8  %let dscount = 3;
9  %do j = 1 %to &dscount;
10   title1 "%upcase(sashelp.&&ds&j)";
11   proc print data=sashelp.&&ds&j (obs=3) noobs;
12   run;
13 %end;
14 %mend VList;
15 %VList
```

Acknowledgements: Portions of SAS' copyrighted SAS course content are reproduced here with permission of SAS Institute Inc., Cary, NC, USA. SAS Institute Inc. makes no warranties with respect to these materials and disclaims all liability therefor. Parts of additional materials are adapted from SAS Documentation, and web-based resources. Please let me know of any errors or typos you have found. Do not copy or circulate the handouts

```
17  *Ex14_VList_HList.sas (Part 2);
18  options nodate nonumber symbolgen;
19  %macro HList;
20    %local dslist j;
21    %let dslist = class revhub2 iris;
22    %do j =1 %to %sysfunc(countw(&dslist));
23       title1 "sashelp.%scan(&dslist,&j)";
24       proc print data= sashelp.%scan(&dslist, &j)(obs=3) noobs;
25       run;
26    %end;
27  %mend HList;
28  %HList
```



Acknowledgements: Portions of SAS' copyrighted SAS course content are reproduced here with permission of SAS Institute Inc., Cary, NC, USA. SAS Institute Inc. makes no warranties with respect to these materials and disclaims all liability therefor. Parts of additional materials are adapted from SAS Documentation, and web-based resources. Please let me know of any errors or typos you have found. Do not copy or circulate the handouts

Idea Exchange

What is the difference between %IF-%THEN and IF-THEN?

	%IF-%THEN	IF-THEN
Valid in	Macro definition	DATA step
Performs	SAS code (text) processing	Data processing
Passed to	Macro processor	DATA step compiler
Purpose	Determine what SAS code to place on the input stack for tokenization, compilation, and eventual execution	Determine what DATA step statement (or statements) to execute during each execution-time iteration of the DATA step

24

Acknowledgements: Portions of SAS' copyrighted SAS course content are reproduced here with permission of SAS Institute Inc., Cary, NC, USA. SAS Institute Inc. makes no warranties with respect to these materials and disclaims all liability therefor. Parts of additional materials are adapted from SAS Documentation, and web-based resources. Please let me know of any errors or typos you have found. Do not copy or circulate the handouts

Monitoring Macro Execution

The MLOGIC system option displays macro execution messages in the SAS log.

Partial SAS Log

```
494  %macro reports;
495     %daily
496     %if &sysday=Friday %then %weekly;
497  %mend reports;
498
499  options mlogic;
500  %reports
MLOGIC(REPORTS):  Beginning execution.
MLOGIC(DAILY):  Beginning execution.
MLOGIC(DAILY):  Ending execution.
MLOGIC(REPORTS):  %IF condition &sysday=Friday is TRUE
MLOGIC(WEEKLY):  Beginning execution.
MLOGIC(WEEKLY):  Ending execution.
MLOGIC(REPORTS):  Ending execution.
```

OPTIONS MLOGIC;

The default setting is NOMLOGIC.

m105d01a

11

Processing Complete Steps

Method 2 Create a single macro with %DO and %END statements to generate text that contains semicolons.

```
%macro reports;
   proc print data=orion.order_fact;
      where order_date="&sysdate9"d;
      var product_id total_retail_price;
      title "Daily sales: &sysdate9";
   run;
 %if &sysday=Friday %then %do;
   proc means data=orion.order_fact n sum mean;
      where orc
            "&sy                          9"d;
      var quant
      title "We
   run;
 %end;
%mend reports;
```

%IF *expression* **%THEN %DO;**
statement; *statement*;...
%END;
%ELSE %DO;
statement; *statement*;...
%END;

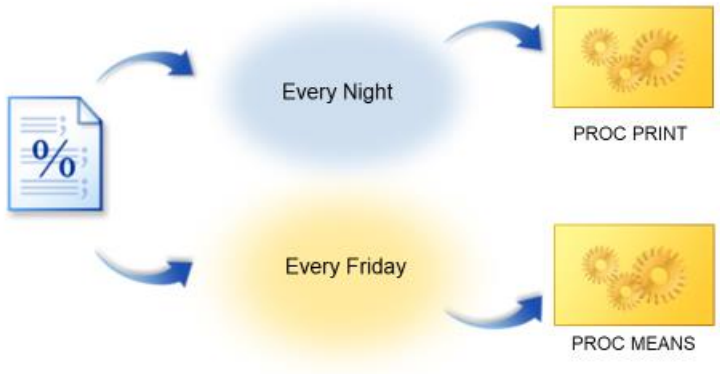
m105d01b

14

Acknowledgements: Portions of SAS' copyrighted SAS course content are reproduced here with permission of SAS Institute Inc., Cary, NC, USA. SAS Institute Inc. makes no warranties with respect to these materials and disclaims all liability therefor. Parts of additional materials are adapted from SAS Documentation, and web-based resources. Please let me know of any errors or typos you have found. Do not copy or circulate the handouts

Business Scenario

A daily sales report is generated every night. Every Friday, a weekly report is generated. Determine the best method to automate these reports.



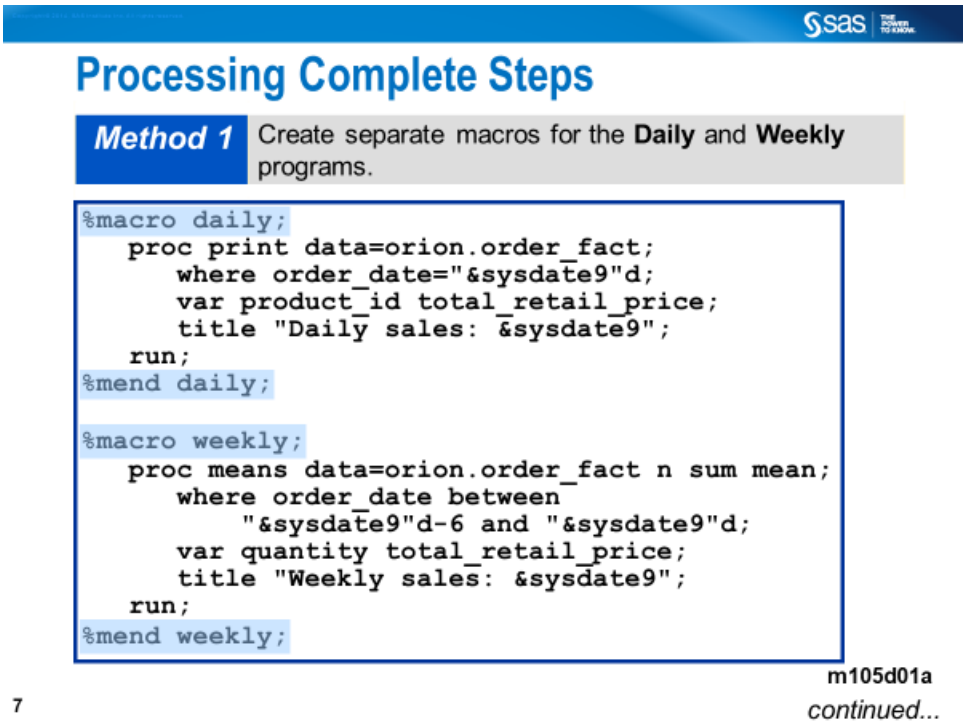
4

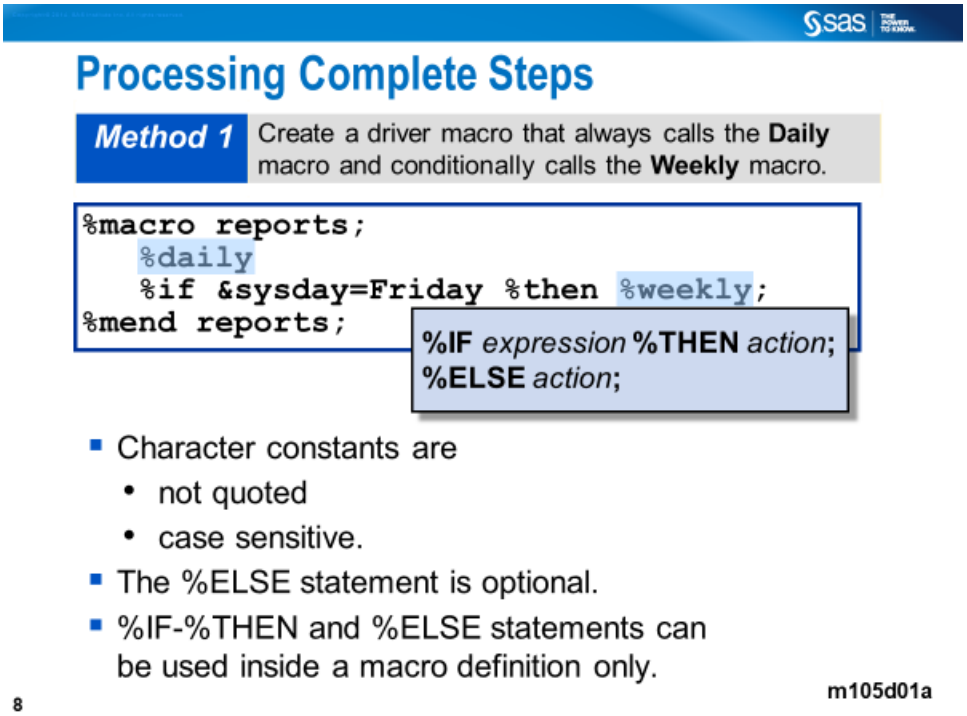
Solutions

Three methods:

Method 1	Create multiple macros, including a driver macro.
Method 2	Create a single macro with %DO and %END statements.
Method 3	Create a single macro with %INCLUDE statements.

6





Acknowledgements: Portions of SAS' copyrighted SAS course content are reproduced here with permission of SAS Institute Inc., Cary, NC, USA. SAS Institute Inc. makes no warranties with respect to these materials and disclaims all liability therefor. Parts of additional materials are adapted from SAS Documentation, and web-based resources. Please let me know of any errors or typos you have found. Do not copy or circulate the handouts

Adding variable names dynamically to the PROC FREQ step using the %IF-%then in a macro

```
 1  *Ex15_macro_part_of_SAS_statement.sas;
 2  options nocenter nodate nonumber;
 3  %macro run_freq(row_var);
 4     proc freq data=sashelp.heart;
 5           tables
 6     %if &row_var ne  %then &row_var *;
 7        smoking_status;
 8     run;
 9  %mend run_freq;
10
11  %run_freq()
12  %run_freq(sex)
13  %run_freq(Weight_Status)
```

Line 6: Here, we conditionally insert text into the middle of a statement to generate one-way or two-way frequency tables.

Acknowledgements: Portions of SAS' copyrighted SAS course content are reproduced here with permission of SAS Institute Inc., Cary, NC, USA. SAS Institute Inc. makes no warranties with respect to these materials and disclaims all liability therefor. Parts of additional materials are adapted from SAS Documentation, and web-based resources. Please let me know of any errors or typos you have found. Do not copy or circulate the handouts

Creating Data-Dependent Values of a Macro Variable for Table Look-Up

```
1  *Ex9_Macro_In_Operator.sas;
2  proc sql ;
3         select quote(strip(Name))
4         INTO  :Starts_withC separated by ','
5       from sashelp.demographics
6         where Name LIKE 'C%';
7  quit;
8  %PUT &Starts_withC;
```

(Partial SAS Log)

```
"CANADA","COSTA RICA","CUBA","CHILE","COLOMBIA","CZECH
REPUBLIC","CROATIA","CAMEROON","CAPE
VERDE","CENTRAL AFRICAN
REP.","CHAD","COMOROS","CONGO","CAMBODIA","CHINA","CYPRUS","COOK
ISLANDS","CORAL SEA ISLANDS"
```

Using a Macro Variable Reference to Select a List of Character Values in the PROC PRINT Step *without a Macro*

```
10  *Ex9_Macro_In_Operator.sas;
11   proc print data=sashelp.demographics;
12    var Name pop;
13    where Name in  (&Starts_withC);
14    run;
```

Acknowledgements: Portions of SAS' copyrighted SAS course content are reproduced here with permission of SAS Institute Inc., Cary, NC, USA. SAS Institute Inc. makes no warranties with respect to these materials and disclaims all liability therefor. Parts of additional materials are adapted from SAS Documentation, and web-based resources. Please let me know of any errors or typos you have found. Do not copy or circulate the handouts

Using a Macro Variable Reference to Select a List of Character Values in the PROC PRINT Step *with a Macro*

```
16  *Ex9_Macro_In_Operator.sas;
17  %macro cn / minoperator mindelimiter=',';
18    proc print data=sashelp.demographics;
19    var Name pop;
20    where Name in  (&Starts_withC);
21    run;
22  %mend cn;
23  %cn
```

The compiled macro is stored in a SAS catalog (WORK.SASMACR).

LINE 15: The MINOPERATOR specifies that the macro processor recognizes and evaluates the mnemonic **IN** and the special character # as logical operators when evaluating arithmetic or logical expressions during the execution of the macro. The MINDELIMETER=',' specifies the character to be used as the delimiter for the macro In operator.

Guideline 5: Comment Macro Applications

Partial SAS Program

```
%macro archive(dsn);

   /*-----------------------------------------------------*/
   /* ARCHIVE - Archive data set, appending current date  */
   /* to build a new (unique by day) name for the copy.   */
   /*-----------------------------------------------------*/
   /* Parameter DSN: Master table to be archived          */
   /*-----------------------------------------------------*/
   /* Requires these other macros:                        */
   /*   %VALIDDSN - validates a SAS data set name         */
   /*-----------------------------------------------------*/

%* %put Activating the MPRINT and MLOGIC options.;

*options mprint mlogic;
```

21

Acknowledgements: Portions of SAS' copyrighted SAS course content are reproduced here with permission of SAS Institute Inc., Cary, NC, USA. SAS Institute Inc. makes no warranties with respect to these materials and disclaims all liability therefor. Parts of additional materials are adapted from SAS Documentation, and web-based resources. Please let me know of any errors or typos you have found. Do not copy or circulate the handouts