```
/*********************************************************
         Week 14: Basic SAS MACRO
          Author: C. Peng

         Topics 1: Introduction
                2: MACRO variable
                3: General MACRO
                4: MACRO with Parameters
**********************************************************/

OPTIONS PS = 26 LS = 72 NODATE NONUMBER;
TITLE "";
FOOTNOTE "";

/** Working Data Set  **/
DATA MODELS;
    INFILE DATALINES;
    INPUT Model $ 1-12 Class $ Price Frame$ 28-38;
    DATALINES;
Black Bora   Track      796 Aluminum
Delta Breeze Road       399 CroMoly
Jet Stream   Track     1130 CroMoly
Mistral      Road      1995 Carbon Comp
Nor'easter   Mountain   899 Aluminum
Santa Ana    Mountain   459 Aluminum
Scirocco     Mountain 2256 Titanium
Trade Wind   Road       759 Aluminum
;
RUN;

/*****************************************************************
    Topic #1: Introduction

   Q. What is SAS MACRO?
   A. A SAS MACRO is a SAS facility of text replacement.

     There are two SAS MACROs: MACRO variable and MACRO program

    A SAS MACRO Variable replaces the small scale of texts in SAS
programs.
    A SAS MACRO Program replaces large scale of texts in SAS programs.

   Macro triggers: & %
         & - Macro variable resolution
         % - Macro invocation

  %LET statement - defining a SAS MACRO variable

  %MACRO macro_program_name(list of parameters);
     text replacement/text filler...
  %MEND macro_program_name;

  Note: macro naming convention - can be up to 32 characters in length.

 *****************************************************************/
```

```
/** Example 1: explicit definition of SAS MACRO variable using %LET
statement. **/
%LET my_macro_var1 = 3+4;   /* plain text value          */
%LET my_macro_var2 = '3+4'; /* string with single quote  */
%LET my_macro_var3 = " 3+4 "; /* string with double quote  */

/**  **/
DATA TEST_MACRO_VAR;
my_var = 3 + 4;
macro_var1 = &my_macro_var1;  /* &my_macro_var1 = 3 + 4 */
macro_var2 = &my_macro_var2;  /* &my_macro_var1 = '3+4' */
macro_var3 = &my_macro_var3;  /* &my_macro_var1 = "3+4" */
RUN;

PROC PRINT DATA = TEST_MACRO_VAR;
RUN;

/**  Example 2: text/string replacement in titles and footnotes     **/
/** Calling a macro in a string such as in title and footnote, double
    quote must be used to resolve the value of the MACRO variable.  **/
PROC PRINT DATA = TEST_MACRO_VAR;
TITLE1 "1-Calling macro_var1 (double-quote): &my_macro_var1";
TITLE2 '2-Calling macro_var1 (single-quote): &my_macro_var1';
TITLE3 "3-Calling macro_var2 (double-quote): &my_macro_var2";
TITLE4 '4-Calling macro_var2 (single-quote): &my_macro_var2';
TITLE5 "5-Calling macro_var3 (double-quote): &my_macro_var3";
TITLE6 '6-Calling macro_var3 (single-quote): &my_macro_var3';
RUN;

/*********************************
     Topic #2: MACRO Variable
************************************/

%LET bikeclass = Mountain;     /* Definition of MACRO variable  */

/** Example 1: Subsetting without using MACRO variable **/
PROC PRINT DATA = models NOOBS;
    WHERE Class = 'Mountain';
    FORMAT Price dollar6.;
    TITLE "Current Models of Mountain Bicycles 01";
RUN;

/** Example 2: Use a macro variable - bikeclass  **/
PROC PRINT DATA = models NOOBS;
    WHERE Class = "&bikeclass";
    FORMAT Price dollar6.;
    TITLE "Current Models of &bikeclass Bicycles 02";
RUN;

/** Example 3. make a title macro variable **/

%LET mytitle = STA311 Week #14 SAS Code;

PROC PRINT DATA = MODELS;
TITLE &mytitle;
RUN;
```

```
/***************************************
   Topic 3: MACRO with no Parameter

This MACRO is useful when a chunk of code
appears repeatedly in the program.

The general form of a macro is

%MACRO macro-name;
     macro-text
%MEND macro-name;

****************************************/

/** Example 1: When we develop a SAS program, we frequently print out
the
    most created data set to check the correctness of the code.
Normally we
    write a PROC PRINT step to see the content of the most recently
created
    data. We can write a SAS MACRO to avoid writing the PROC PRINT
every time.
 **/

%MACRO printit;          /* open the macro statement*/
DM CLEAR OUT;
PROC PRINT;
    TITLE 'Most Currently Created Data';
RUN;
TITLE "";
%MEND printit;           /* close the macro statement*/

/** create a SAS data set and then use %printit to print it out  **/
DATA testdat;
var1 = 123;
var2 = "123";
RUN;

%printit;

PROC CONTENTS DATA = testdat;
RUN;



/** Example 2. set of MACRO statements **/
DATA myDATA;
INPUT var1 var2 var3;
DATALINES;
1 3 6
4 3 9
6 5 2
12 4 5
;
RUN;
```

```
%printit;

/** macro for variable transformation **/
%MACRO comput;
var4 = var1 + var2;
var5 = var1 - var2;
var6 = var1 + var2 + var3;
%MEND comput;

DATA MyNEWData;
SET myDATA;
%comput;
RUN;

%printit;


/***********************************
   Topic 4: MACRO with parameters
************************************/

/** Think about sorting different variables in the
data set and then print out. For each printout, we
want to add titles with the information on sorting
variables or variables.

To avoid writing multiple procedures due to sorting
the data with different variables, we can SAS MACRO.

Two types of MACRO parameters: Positional and Keyword

(1) Positional:  %MACRO macro_name(sortseq, sortvar)
(2) Keyword:     %MACRO macro_name(sortseq =, sortvar =)
**/

/** Example 1: MACRO with positional parameter  **/
%MACRO PositionalSortPrint(sortseq, sortvar, out);
    PROC SORT data = models out = models&out;
        by &sortseq &sortvar;
    RUN;

    PROC PRINT data = models&out  noobs;
        title1  'Current Models:';
        title2 "Sorted by &sortseq &sortvar: OUTPUT&out";
        var Model Class Frame Price;
        format price dollar6.;
    RUN;
%MEND PositionalSortPrint;

%PositionalSortPrint(DESCENDING, Price, 01)
%PositionalSortPrint(Price, DESCENDING, 02) /* order of values were
messed up!!!! */

%PositionalSortPrint( , Price, 03)
%PositionalSortPrint( , Class, 04)
%PositionalSortPrint( DESCENDING, Class, 05)
```

```
/** Example 2: MACRO with keyword parameter  **/
%MACRO KeywordSortPrint(sortseq =, sortvar =, out =  );
    PROC SORT data = models out = models&out;
        by &sortseq &sortvar;
    RUN;

    PROC PRINT data = models&out noobs;
        title  'Current Models';
        title2 "Sorted by &sortseq &sortvar: OUTPUT&out";
        var Model Class Frame Price;
        format price dollar6.;
    RUN;
%MEND KeywordSortPrint;

%KeywordSortPrint(sortseq = Descending, sortvar = Price, out = 06)

%KeywordSortPrint(sortseq =, sortvar = Price, out = 07)
```