\* How to call SAS Macro from External Location

\* % INCLUDE 'MACRO-PATH';

%include "/home/u1048896/test.sas";

%test(cust, product);

\* Suppose your macro is stored in a location and you need to call it from the location. You

can accomplish this task using %Include.;

/\* SAS : DIFFERENCE BETWEEN SYMPUT AND SYMGET;

SYMPUT : To create macro variables in a data step.

SYMGET : To get macro variable value in a data step.

\*/

\* Example 1 : Creating a single macro variable;

data \_null\_;

set sashelp.class;

if \_N\_ = 1 then do;

call symput('nvar', name);

end;

run;

%put &nvar;

\* Get macro variable value in a dataset;

data want;

var1=symget('nvar');

run;

proc print data=want;

run;

\* Example 2 : Creating multiple macro variables;

data \_null\_;

set sashelp.class;

call symput('nvars' || strip(\_n\_), name);

run;

%put &nvars1 &nvars2 &nvars3;

\* Number of rows;

data \_null\_;

if 0 then set sashelp.class nobs=n;

call symput ('nrows',n);

run;

%put &nrows;

\* Get macro variable value in a dataset;

data want (drop = i);

do i=1 to &nrows.;

var1=symget(cats('nvars',i));

output;

end;

run;

proc print data=want;

run;

\* Multiple Ampersand Macro Variables;

\* SINGLE, DOUBLE, TRIPLE AMPERSAND SAS MACRO VARIABLES;

%let x=temp;

%let n=3;

%let x3=result;

%let temp3 = result2;

%put &x&n;

%put &&x&n;

%put &&&x&n;

/\* Rule :

The scanner reads from left to right.

&x&n : Macro variable X resolves first to temp and then N resolves to 3. Output : temp3

&&x&n : Two ampersands (&&) resolves to one ampersand (&) and scanner continues and then N resolves

to 3 and then &x3 resolves to result. Output : result

&&&x&n : First two ampersands (&&) resolves to & and then X resolves to temp and then N resolves to 3.

In last, &temp3 resolves to result2. Output : result2

\*/

\* SAS : CALL EXECUTE - To interact with the SAS macro facility;

/\* Example 1 : Suppose you have two data sets named "temp" and "temp2". You are asked to form a group based

on the logical conditions given in the other dataset "temp2" and apply the conditions in dataset temp.

if x less than or equal to 30 then Groups=3;

if x less than or equal to 20 then Groups=2;

if x less than or equal to 10 then Groups=1;

\*/

Data temp;

input x;

cards;

5

10

15

20

25

30

;

run;

Proc print data=temp;

run;

Data temp2;

input var $ Score rank Section $;

cards;

x 30 3 C

x 20 2 B

x 10 1 A

;

run;

Proc print data=temp;

run;

Data \_null\_;

set temp2 end=last;

if \_n\_=1 then call execute ('Data output; set temp;');

call execute ("if " ||strip(Var)|| " LE " ||strip(Score) || " then " || "Groups" || "=" ||strip(rank) ||";");

if last then call execute (' run;');

run;

Proc print data=output;

run;

/\* Form a new Character variable using the following conditions

If x LE 30 then Class="C";

If x LE 20 then Class="B";

If x LE 10 then Class="A";

\*/

Data \_null\_;

set temp2 end=last;

if \_n\_=1 then call execute ('Data output; set temp; length Class $10.;');

call execute ("if " ||strip(Var)|| " LE " ||strip(Score) || " then " || "Class" || "=" || """" || strip(Section) ||""""|| ";");

if last then call execute (' run;');

run;

Proc print data=output;

run;

\* Example 2 : Print Multiple Datasets;

data \_null\_ ;

input mydata $char50. ;

call execute

("proc print" || " data = " || strip(mydata) || ";" || "run ;");

cards;

temp

temp2

output

;

run;

\* Example 3 : Call a Macro;

%macro mymacro(k);

data want;

set temp;

%do i = 1 %to &k;

if \_N\_ = &i then y = %eval(&i.\* 10);

%end;

run;

%mend;

data \_null\_;

call execute ('%mymacro(6)');

run;

Proc print data=want;

run;

\* Example 4 : Dynamically Call Macro;

%macro mymacro(i,j);

%put first = &i. second = &j.;

%mend;

DATA example;

input x y $32. ;

datalines;

1 temp

2 temp2

;

run;

data \_null\_;

set example;

call execute('%MyMacro('||x||','||y||')');

run;

\* STOP SAS MACRO ON ERROR;

\* 1. Stop Macro Processing on Error;

\* In the following program, we are telling SAS to stop sas code if user does not specify

parameters and notifying them what they have missed. The %abort cancel; statement tells

SAS to abort execution immediately.;

%macro explore(inputdata= ,var=);

options notes;

%if %length(&inputdata) = 0 %then %do;

%put ERROR: INPUTDATA= must be specified;

%put ERROR: The macro ended abnormally.;

%abort cancel;

%end;

%if %length(&var) = 0 %then %do;

%put ERROR: VAR= must be specified;

%put ERROR: The macro ended abnormally.;

%abort cancel;

%end;

proc sort data = &inputdata.;

by &var.;

run;

%mend;

%explore(inputdata = , var = age );

\* Logic - If the length of string of a macro parameter is 0, it means the macro parameter

is blank.;

\* 2. Go to End of Program If Error;

\* In the following program, we are telling SAS to go to end of the code if error comes,

The %goto statement is used to jump to end of the program.;

%macro explore(inputdata= ,var=);

options notes;

%if %length(&inputdata) = 0 %then %do;

%put ERROR: INPUTDATA= must be specified;

%put ERROR: The macro ended abnormally.;

%goto exit;

%end;

%if %length(&var) = 0 %then %do;

%put ERROR: VAR= must be specified;

%put ERROR: The macro ended abnormally.;

%goto exit;

%end;

proc sort data = &inputdata.;

by &var.;

run;

%exit:

%mend;

%explore(inputdata = , var = age );

\* 3. Check for Error after each step of SAS Code;

\* Sometimes we make typo while entering dataset or variable name. It is important to

handle these kinds of errors as well so we need to check for error(s) after each step

of SAS Code (Data Step, PROCs). %if &syserr. ne 0 %then %do; works for it.;

%macro explore(inputdata= ,var=);

options notes;

%if %length(&inputdata) = 0 %then %do;

%put ERROR: INPUTDATA= must be specified;

%put ERROR: The macro ended abnormally.;

%abort cancel;

%end;

%if %length(&var) = 0 %then %do;

%put ERROR: VAR= must be specified;

%put ERROR: The macro ended abnormally.;

%abort cancel;

%end;

proc sort data = &inputdata.;

by &var.;

run;

%if &syserr. ne 0 %then %do;

%abort cancel;

%end;

%mend;

%explore(inputdata = sashelp.clss , var = age );

\* SAS MACRO : COUNT NUMBER OF VARIABLES ASSIGNED IN A MACRO VARIABLE;

\* Option I;

%macro nvars (ivars);

%let n=%sysfunc(countw(&ivars));

%put &n;

%mend;

%nvars (X1 X2 X3 X4);

\* Option II;

%macro nvars (ivars);

%let n=1;

%do %until ( %scan(&ivars,&n)= );

%let n=%EVAL(&n + 1);

%end;

%let n=%eval(&n-1);

%put &n;

%mend;

%nvars ( X1 X2 X3 X4);

\* SAS MACRO : A DYNAMIC %DO LOOP;

\* Suppose you need to pass a variable in loop based on the input defined in a macro.;

data test;

input a b c;

cards;

1 6 7

1 8 9

1 7 9

1 5 0

2 1 0

2 5 6

2 7 9

2 4 5

;

run;

%macro report (input=, var = , class=);

%let n=%sysfunc(countw(&var));

%do i=1 %to &n;

%let val = %scan(&var,&i);

proc means data = &input noprint nway;

class &class;

vars &val;

output out=out&i mean= median= / autoname;

run;

%end;

%mend;

options mprint;

%report(input= test, var = b c, class=a);

/\* When you execute the above sas program, it generates the following statements :

proc means data = test noprint nway;

class a;

vars b;

output out=out1 mean= median= / autoname;

run;

proc means data = test noprint nway;

class a;

vars c;

output out=out2 mean= median= / autoname;

run;

\*/

\* SAS MACRO : GET VARIABLE NAMES FROM A DATASET;

\* 1. Selecting all the variables;

proc sql noprint;

select name into : vars separated by " " from dictionary.columns

where LIBNAME = upcase("sashelp")

and MEMNAME = upcase("class");

quit;

\* LIBNAME : Library Name;

\* MEMNAME : Dataset Name;

\* Dictionary.Columns - It returns information about the columns in one or more data sets.

It is similar to the results of the CONTENTS procedure.;

%put variables = &vars.;

\* 2. Get all the numeric variable names from a data set;

\* Selecting numeric variables;

proc sql noprint;

select name into : numvar separated by " " from dictionary.columns

where LIBNAME = upcase("sashelp")

and MEMNAME = upcase("class")

and type = 'num';

quit;

%put numvariables = &numvar.;

\* 3. Get all the character variable names from a data set;

\* Selecting character variables;

proc sql noprint;

select name into : charvar separated by " " from dictionary.columns

where LIBNAME = upcase("sashelp")

and MEMNAME = upcase("class")

and type = 'char';

quit;

%put charvariables = &charvar.;

\* 4. Get all the variable names except Name variable;

proc sql noprint;

select name into : vars separated by " " from dictionary.columns

where LIBNAME = upcase("sashelp")

and MEMNAME = upcase("class")

and upcase(name) ne upcase("name");

quit;

%put variables = &vars.;

/\* SAS MACRO : RUN SAS PROCEDURE ON MULTIPLE DATASETS

Dictionary.Columns - It returns information about the columns in one or more data sets.

It is similar to the results of the CONTENTS procedure.

Dictionary.Tables - It returns information about names of SAS files and type, date created

and last modified, number of observations, observation length, number of variables etc.

\*/

\* Task : Export all SAS data sets of a library in CSV format;

\* Count Number of Datasets in a library;

%let lib = sashelp;

proc sql noprint;

select count(\*) into :n from dictionary.tables where libname=%upcase("&lib");

quit;

%put &n;

\* List name of all datasets in a library;

proc sql noprint;

select memname into :data1 - :data%LEFT(&n) from dictionary.tables

where libname=%upcase("&lib");

quit;

%put &data1;

%put &data100;

%put &data238;

%macro temp;

%do i=1 %to &n.;

proc export data = &lib..&&data&i

outfile = "/home/u1048896/&&data&i...csv" DBMS = CSV;

run;

%end;

%mend;

%temp;

\* SAS : DROPPING VARIABLES ENDING WITH A SPECIFIC STRING;

\* Suppose you need to drop all the variables ending with a specific string from your dataset.

For example. you have a dataset named 'Have' and you want to remove all the variables ending

with '\_d' or '\_D'.;

data have;

set sashelp.class;

r\_d = ranuni(9);

b\_d = ranuni(10);

c\_DD = ranuni(11);

run;

\* Assign library and filename;

%let libname= work.have;

\* SAS Code : Dropping variables ending with '\_d' or '\_D';

\* Extracting library name;

\* Extracting dataset name;

\* Upcase all macro variables to have consistency;

data \_null\_;

call symput ("library", put(upcase(substr("&libname",1,index("&libname",'.')-1)), $8.));

call symput ("datset", put(upcase(substr("&libname",index("&libname",'.')+1,length("&libname"))), $32.));

%put &library &datset;

run;

\* Get variable list;

proc sql noprint;

select name into : var\_list separated by " " from dictionary.columns

where LIBNAME = "&library" and MEMNAME = "&datset"

and upcase(substr(name,length(name)-1,2)) = '\_D';

quit;

%put &var\_list;

\* Dropping all the variables ending with '\_D';

data &libname.1;

set &libname. (drop=&var\_list);

run;