Statistical Analysis System: Class 33

Dated: 07/07/2018

::

7th way to create a Macro variable:

Call Symput: It is a call routine. It creates a macro variable in a datastep or assigns a data step value to an existing macro variable

Syntax: call symput(arg1,arg2);

arg1: specifies a character expression which is also the macro variable that is assigned a value, if macro variable doesn't exist the routine creates it.

arg2: specifes a character constant, variable or expression that contains the value that is assigned.

Prog 1:

```
    data _null_;                                        /* NULL is used here as the
     dataset name so that only  the macro variable is created without the dataset
                                                    being created unnecessarily */
name=99;
call symput ('x',name);
call symputx('y',name);
run;

%put ***&x***;
%put **&y***;
```

Differece between symput and symputx:

Symput	Symputx
if variable value is character then symput applies.	if variable value is numeric then symputx applies.

%eval and %sysevalf: These are two macro evaluation functions.

%eval: evaluates using integer arithematic. syntax: %eval(arithematic or logical expression)

%sysevalf: evaluates using floating point arithematic. %sysevalf(arithematic or logical expression)

Prog 2:

```
%let a=1;
%let b=2;

%let c= &a + &b;
%put ***&c**;
```

```
%let d= %eval(&a+&b);
%put ***&d**;
```

Prog 3:

```
%let a=1.2;
%let b=1.2;

%let c= %eval(&a + &b);
%put ***&c**;

%let d= %sysevalf(&a+&b);
%put ***&d**;
```

Scope of macro variable

SCOPE / Preference of a macro variable created: local first and then global. Every macro variable created is stored in one of two symbol tables (LOCAL / GLOBAL).

Symbol Table: Stores value of macro variable. The symbol table, lists the macro variable name and its value and determines its scope.

Global macro variables or those stored in the global symbol table, exist for the duration of the SAS session and can be referenced anywhere except in the CARDS or DATALINES statements.

Local macro variables or those stored in a local symbol table, exist only during the excecution of the macro in which the variable is created.

Macro variables included under Global symbol table:	**Macro variables included under Local symbol table:**
all automatic macro variable	macro parameters
macro variable created outside of any macro definition.	macro variables created on %LOCAL statement
macro variables created on a %GLOBAL statement	macro statements that define macro variables within a macro defintion, such as %LET and the iterative %DO statement (only if the variable is not global already)
most macro variables created by CALL SYMPUT / CALL SYMPUTX (some exception).	

Note: Call SYMPUT always creats macro variable in the nearest non-empty symbol tables.

Debugging Macros:

1. **%PUT statement:** To determine the symbol table where a macro variable is stored in, following are the ways:

• %PUT _ALL_;	lists all macro variable
• %PUT _GLOBAL_;	lists only global one's
• %PUT _LOCAL_;	lists only local variable

- %PUT _AUTOMATIC_; lists only automatic, which are by default in GLOBAL symbol table
- %PUT _user_;   lists user defined global and user defined local variable

Prog 4:

```
%put _all_;
Proc print data = sasuser.admit;
title "report created on &sysdate9. and &sysday by &sysuserid";
run;

%put _global_
```

2. **Macro debugging options:** Mprint, Mlogic, Symbolgen etc.. Like other system options you can turn these options on for debugging by using options statement:

   Syntax: Options mprint mlogic symbolgen;

   MPRINT: Displays all SAS statements of the macro code in the log or you may say, it converts a macrotized code into simple sas code.

   MLOGIC: This option identifies and displays the macro logic and in the macro execution pattern. Helpful when dealing with nested macros or %DO loops or %IF-%THEN-%ELSE statements.

   SYMBOLGEN: It prints a message in the LOG whenever a macro variable gets resolved.

Prog 5:

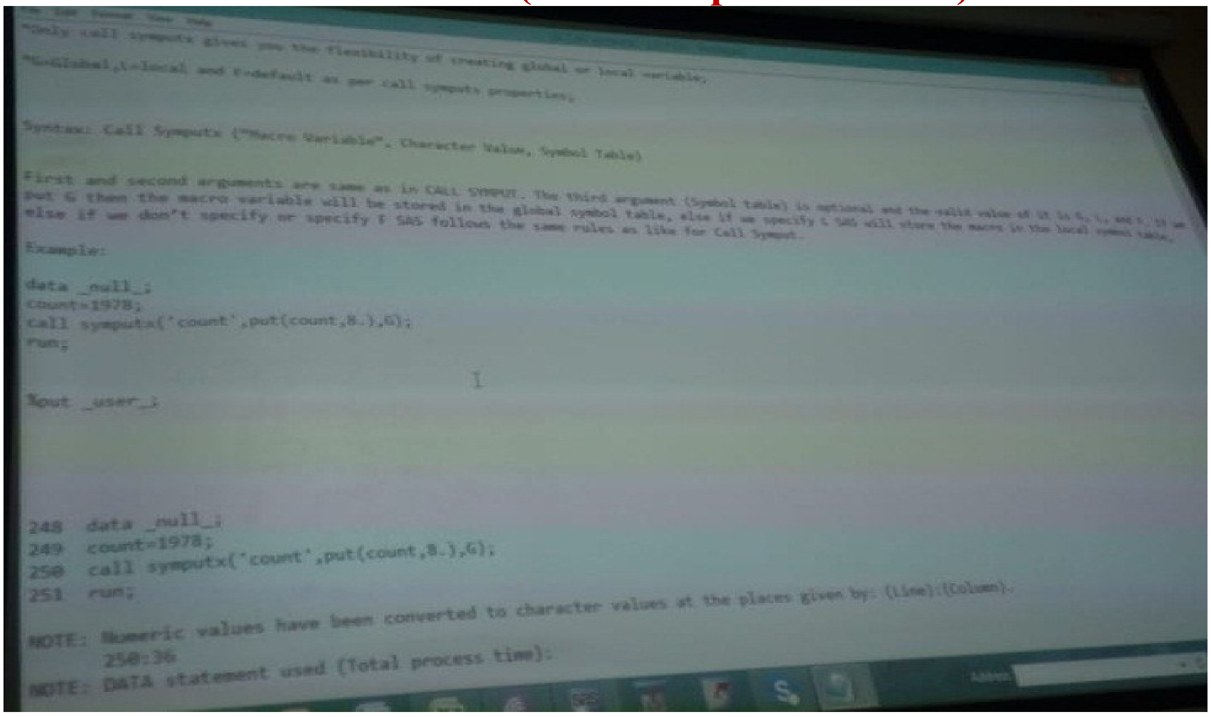
```
Options mprint mlogic symbolgen;

%macro a;
%do i=1 %to 5;

data a&i;
set sasuser.admit;
run;

%end;
%mend a;

%a;
```

Prog 6:

```
Options mprint mlogic symbolgen;

%macro a;

%let j=9;
%do i=1 %to 5;

data a&i&j;
set sasuser.admit;
```

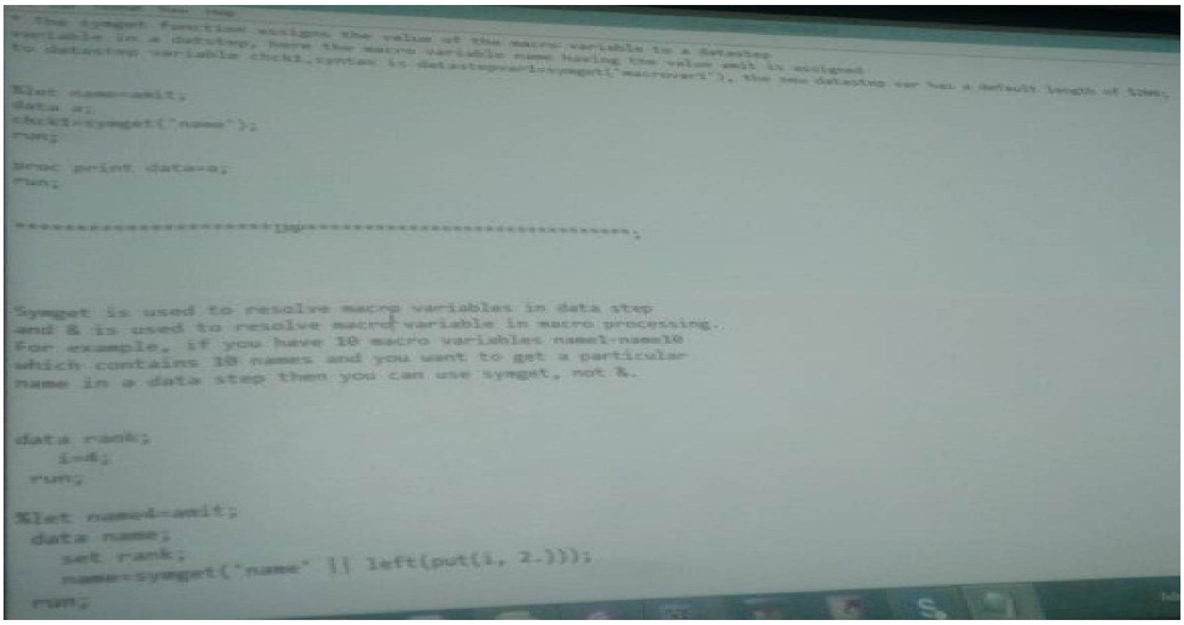
```
run;

%end;
%mend a;

%a;
```

Call SYMPUTX elaborated (see in the picture below):



```
*Only call symputx gives you the flexibility of creating global or local variable;
*G=Global,L=local and F=default as per call symputx properties;

Syntax: Call Symputx ("Macro Variable", Character Value, Symbol Table)

First and second arguments are same as in CALL SYMPUT. The third argument (Symbol table) is optional and the valid value of it is G, L, and F. If we
put G then the macro variable will be stored in the global symbol table, else if we specify L SAS will store the macro in the local symbol table,
else if we don't specify or specify F SAS follows the same rules as like for Call Symput.

Example:

data _null_;
count=1978;
call symputx('count',put(count,8.),G);
run;


%put _user_;



248  data _null_;
249  count=1978;
250  call symputx('count',put(count,8.),G);
251  run;

NOTE: Numeric values have been converted to character values at the places given by: (Line):(Column).
      250:36
NOTE: DATA statement used (Total process time):
```

Symget Function:



```
* The symget function assigns the value of the macro variable to a datastep
variable in a datstep, here the macro variable name having the value amit is assigned
to datastep variable chck1,syntax is datastepvar1=symget('macrovar1'), the new datastep var has a default length of [illegible];

%let name=amit;
data a;
chck1=symget('name');
run;

proc print data=a;
run;


***********************imp***********************************;



Symget is used to resolve macro variables in data step
and & is used to resolve macro variable in macro processing.
For example, if you have 10 macro variables name1-name10
which contains 10 names and you want to get a particular
name in a data step then you can use symget, not &.


data rank;
   i=4;
 run;

%let name4=amit;
 data name;
    set rank;
    name=symget('name' || left(put(i, 2.)));
 run;
```