Lec2

SAS Macro Language

- SAS Macro: **Substitute, Condition, Loop**

- Define Macro Variables- **3 methods**:

(1) **%Let Statement**: Straightforward and easy to understand. Use period to separate out the macro variable you would like to resolve.

(2) **Call symput**: With CALL SYMPUT, you can create macro variables that contain the values of variables from SAS data sets. You cannot resolve them in the same data step.

(3) **SQL Method**: Create macro variables in PROC SQL

(1) %Let Statement

```
%let data=work.overview;     *to define a macro variable;
proc print data=&data;              *use & to resolve it;
*double quotes "" - resolve macro variable;
  title "A listing of the &data SAS data set";
*single quotes '' - not resolve macro variable;
  title2 'A listing of the &data SAS data set';
run;
```

```
%let i=10; *what if I want resolve "i" in "ith";
proc print data=work.overview(obs=&i firstobs=&i);
title "&i.th Record in work.overview"; *resolved as 10th;
title2 "&ith Record in work.overview"; *can not resolve, since ith is
not a macro variable;
run;
```

- Use **%put statement** to put your resolution into LOG window.

Example 19:

The following SAS program is submitted:
%let product=merchandise;
[_insert_%put_statement_]
and the following message is written to the SAS log: the value is "merchandise"

Which macro statement wrote this message?
A. %put the value is &product;
B. %put the value is %quote(&product.);
C. %put the value is "&product.";
D. %put the value is ""&product."";

Example 20:

Given the following program and data:

```
data WORK.BDAYINFO;
infile datalines;
input Name $ Birthday : mmddyy10.;
datalines;
Alan  11/15/1950
Barb  08/23/1966
Carl  09/01/1963
;
run;

%let Want=23AUG1966;
proc print data=WORK.BDAYINFO;
  [_insert_statement_]
run;
```

What is the WHERE statement that successfully completes the PROC PRINT and selects the observation for Barb?

A. where Birthday=&Want;
B. where Birthday="&Want";
C. where Birthday="&Want"d;
D. where Birthday='&Want'd;

(2) Call symput(var1,var2)

```
data _null_;
  set overview;
  call symput(Record_No,gdp);
run;
```

(3) Proc SQL Method

```
PROC SQL;
select count(*) into: n from work.overview;
select distinct country into: country_list separated by ","
                      from work.overview;
quit;
```

Example 21:

Given the SAS data set SASUSER.HIGHWAY:

Steering	Seatbelt	Speed	Status	Count
absent	No	0-29	serious	31
absent	No	0-29	not	1419
absent	No	30-49	serious	191
absent	No	30-49	not	2004

absent No 50+ serious 216

The following SAS program is submitted:

```
proc sql noprint;
  select distinct
    Speed [_insert_SQL_clause_]
  from SASUSER.HIGHWAY
  ;
quit;

title1 "Speed values represented are: &GROUPS";
proc print data=SASUSER.HIGHWAY;
run;
```

Which SQL clause stores the text **0-29, 30-49, 50+** in the macro variable GROUPS?
A. into &GROUPS
B. into :GROUPS
C. into :GROUPS separated by ','
D. into &GROUPS separated by ','

Consecutive &

```
%let a=alpha;
%let b=beta;
%let abeta=pair;

%put &a&b;     *resolve to alphabeta;
%put &&a&b;    *resolve to: &abeta which resolves to pair;
```

- Method to understand how consecutive & resolves
  - Rule 1: SAS read left to right
  - Rule 2: &&→&
  - Rule 3: In each read, SAS only resolve each & once
  - Rule 4: SAS will read again while there is & unresolved

Exercise:

```
%let attri=name;
%let name=shan;
%let shan=chinese;
%put &name is a &&&&name;
%put &name is a &&&&&name;
%put &name is a &&&&&&name;
```

Example 22:
Given the following program and desired results:

```
%let Thing1=gift;
%let Thing2=surprise;
%let Gift1=book;
%let Gift2=jewelry;
%let Surprise1=dinner;
%let Surprise2=movie;

%let Pick=2;
%let Choice=surprise;
```

Desired %PUT Results in LOG:
My favorite surprise is a movie

What is the correct %PUT statement that generates the desired results?
A. %put My favorite &Thing&Pick is a &&Choice&Pick;
B. %put My favorite &&Thing&pick is a &&&Choice&Pick;
C. %put My favorite &Choice&pick is a &&Thing&Pick;
D. %put My favorite &&Choice&pick is a &&&Thing&Pick;

Example 23:
The following SAS program is submitted:

```
%let Name1=Shoes;
%let Name2=Clothes;
%let Root=name;
%let Suffix=2;
%put &&&Root&Suffix;
```

What is written to the SAS log?
A. &Name2
B. Clothes
C. &&&Root&Suffix
D. WARNING: Apparent symbolic reference ROOT2 not resolved.

Macro Functions

- **%STR** and **%NRSTR** functions

```
%let a=   Shan's favorite mathematicians;
%let b=&a are; Newton&Gauss;
```

- %STR is important when the string contains:
  - Unmatched quotes/parenthesis – **use %** before them
  - Semicolon
  - Significant blanks

- %NRSTR function does the same as %STR, but also masks the macro characters & and % so they don't resolve

```
%let a=%str(  Shan%'s favorite mathematicians);
%let b=%str(&a %nrstr(are; Newton&Gauss; &a));
%put &a &b;
```

- **%eval (expression):** allows integer arithmetic of macro variables (decimal points are not allowed in expression)
- **%sysevalf (expression):** returns the numeric results of a mathematical expression, which contains operators and numeric constants only.
- **%sysfunc (data step function, <format>):** allows us to use some data step functions outside of the data step.

```
%let express1=0.25**0.5;
%let express2=%eval(5/3);
%let express3=%sysevalf(0.25**0.5);
%let express4=%sysfunc(sqrt(0.25),4.1);

%put &express1;
%put &express2 &express3 &express4;
```

Example 24:

The following SAS program is submitted:

```
%let Value=9;
%let Add=5;
%let Newval=%eval(&Value/&Add);
%put &Newval;
```

What is the value of the macro variable Newval when the %PUT statement executes?

A. 0.555
B. 2
C. 1.8
D. 1

Example 25:

The following SAS program is submitted:

```
%let Num1=7;
%let Num2=3;
%let Result=%eval(&Num1/&Num2);
%put &Result;
```

What is the value of the macro variable Result when the %PUT statement executes?

A. 2.3
B. 2
C. . (missing value)
D. 2.33333333333333

- **%substr (string, position<,length>):** returns the substring of a string starting from position and ending length characters later.
- **%length (string):** returns length of string,
- **%upcase (string):** returns upper case
- **%scan (string, n <,delimiters>):** returns the nth word of the string where words are delineated by the specified delimiter(s)
- **%index (string, substring):** Returns the position of the first occurrence of the substring found in the string. If the string is not found, then returns 0.

```
%let ss=Canadians are bad at math;
%let ss1=%substr(&ss,11,3);
%let ss2=%scan(&ss,4);
%let ss3=%index(&ss,American);   *no quotes are needed for American;

%put &ss1 &ss2 &ss3 %length(&ss3);
```

Example 26:

Given the dataset WORK.STUDENTS:

Name	Age
Mary	15
Philip	16
Robert	12
Ronald	15

The following SAS program is submitted:

```
%let Value=Philip;
proc print data=WORK.STUDENTS;
  [_insert_WHERE_statement_]
run;
```

Which WHERE statement successfully completes the program and produces a report?

A. where upcase(Name)=upcase(&Value);
B. where upcase(Name)=%upcase(&Value);
C. where upcase(Name)="upcase(&Value)";
D. where upcase(Name)="%upcase(&Value)";

Macros

- **Macros** – a block of SAS steps that is resolved when the macro is revoked.

%MACRO *name*(*positional parameters, keyword parameters*=);
<contents of macro>
%MEND *name*;

%name;

Note: Positional parameters must be defined before keyword parameters

```
%macro print(data,var=);
  proc print data=&data noobs label;
    var &var;
  run;
%mend;

%print(work.overview,var=country year gdp);
```

Example 27:
Macros That Include Keyword Parameters:
Given the SAS data set SASUSER.HIGHWAY:

Steering	Seatbelt	Speed	Status	Count
absent	No	0-29	serious	31
absent	No	0-29	not	1419
absent	No	30-49	serious	191
absent	no	30-49	not	2004
absent	no	50+	serious	216

The following SAS program is submitted:

```
%macro HIGHWAY(Belt=no);
  proc print data=SASUSER.HIGHWAY;
    where Seatbelt="&Belt" ;
  run;
%mend;

%HIGHWAY(Belt=No)
```

How many observations appear in the generated report?
A. 0
B. 2
C. 3
D. 5

Example 28:
The following SAS program is submitted:

```
%macro CHECK(Num=4);
  %let Result=%eval(&Num gt 5);
  %put Result is &result;
%mend;
%check(Num=10)
```

What is written to the SAS log?

A. Result is 0
B. Result is 1
C. Result is 10 gt 5
D. Result is true

Example 29:
Given the data set SASHELP.CLASS:

Name	Age
Mary	15
Philip	16
Robert	12
Ronald	15

The following SAS program is submitted:

```
%macro MP_ONE(pname=means);
  proc &pname data=SASHELP.CLASS;
  run;
%mend;
%MP_ONE(print)
%MP_ONE()
```

Which PROC steps execute successfully?
A. PROC MEANS only
B. PROC PRINT only
C. PROC MEANS and PROC PRINT
D. No PROC steps execute successfully

Example 30:
The following SAS program is submitted:

```
%macro COLS1;
  Name Age;
%mend;
%macro COLS2;
  Height Weight;
%mend;
proc print data=SASHELP.CLASS;
  [_insert_VAR_statement_here_]
run;
```

Which VAR statement successfully completes the program to produce a report containing four variables?

A. var %COLS1 %COLS2;
B. var %COLS1-%COLS2;
C. var %COLS1 Weight Height;
D. var Weight Height %COLS1;

Macros related Global Options

- **SYMBOLGEN:** The SYMBOLGEN options shows when **each macro variable reference resolution** occurs; use SYMBOLGEN to help locate problems with macro variable reference resolution. To Turn it off, use NOSYMBOLGEN.
- **MPRINT:** The MPRINT options shows all **SAS statements generated by macro code**. Use MPRINT to help locate problems from errors generated by the SAS compiler. To turn it off, use NOMPRINT.
- **MLOGIC:** The MLOGIC option prints messages that indicate **macro actions** that were taken during macro execution.

Example 31:

Given the following macro program:

```
%macro MAKEPGM(NEWNAME, SETNAME, PRINT);
  data &NEWNAME;
    set &SETNAME;
  run;
  %if &PRINT=YES %then %do;
    proc print data=&NEWNAME.(obs=10);
    run ;
  %end;
%mend;
```

Which option would provide feedback in the log about the parameter values passed into this macro when invoked?

A. MPRINT
B. MDEBUG
C. MLOGIC
D. MPARAM

Conditional Processing and Iteration

- **Macro IF-ELSE-THEN**

%IF *expression* %THEN %DO;
SAS code;
%END;
%ELSE %DO;
SAS code;
%END;

- **Macro Iterative Do (Do Loop)**

%DO *macro variable* = *start* %TO *stop* <%BY *increment*>;
SAS code;
%END;

```
%DO %WHILE(expression);
SAS code;
%END;
%DO %UNTIL(expression);
SAS code;
%END;
```

- Important: Macro loops and conditional processing as shown previously can not be use in open code. They must only be used within macros.

Example 32:

Given the SAS data set SASUSER.HIGHWAY:

Steering	Seatbelt	Speed	Status	Count
absent	No	0-29	serious	31
absent	No	0-29	not	1419
absent	No	30-49	serious	191
absent	no	30-49	not	2004
absent	no	50+	serious	216

The following SAS program is submitted:

```
%macro SPLIT;
 proc sort data=SASUSER.HIGHWAY
         out=WORK.UNIQUES(keep=Status)  nodupkey;
     by Status;
 run;

 data _null_;
     set uniques end=Lastobs;
     call symputx('Status'||left(_n_),Status);
     if Lastobs then call symputx('Count',_n_);
 run;

 %local i;

  data
 %do i=1 %to &count;
 [_insert_reference_]
 %end;
 ;
     set SASUSER.HIGHWAY;

     select (Status);
       %do i=1 %to &Count;
          when("[_insert_reference_]") output [_insert_reference_];
       %end;
       otherwise;
     end;
```

```
run;

%mend;

%SPLIT
```

What macro variable reference completes the program to create the WORK.NOT and WORK.SERIOUS data sets?

A. &Status&i
B. &&Status&i
C. &Status&Count
D. &&Status&Count

Scope of macro variables – Global and Local

- Global macro variables exist for the duration of the SAS session and can be referenced anywhere.
- Local macro variables exist only during the execution of the macro in which the variables are created and have no meaning outside the defining macro.
- Use %local or %global to force it to be a local or global macro
  %global *macro_name*;
  %local *macro_name*;

- You can create a global macro variable with the following:

  - a %LET statement (used outside a macro definition)
  - a DATA step that contains a SYMPUT routine
  - a DATA step that contains a SYMPUTX routine
  - a SELECT statement that contains an INTO clause in PROC SQL
  - a %GLOBAL statement.

- The %GLOBAL statement

  - creates one or more macro variables in the global symbol table and assigns null values to them
  - can be used either inside or outside a macro definition has no effect on variables that are already in the global symbol table.

- You can create local macro variables with the following:

  - parameters in a macro definition
  - a %LET statement within a macro definition
  - a DATA step that contains a SYMPUT routine within a macro definition
  - a DATA step that contains a SYMPUTX routine within a macro definition
  - a SELECT statement that contains an INTO clause in PROC SQL within a macro definition
  - a %LOCAL statement.

Note: The SYMPUT routine can create a local macro variable if a local symbol table already exists.
A local symbol table is created when a macro that includes a parameter list is called or when a request is made to create a local variable during macro execution. The local symbol table is deleted when the macro finishes execution. That is, the local symbol table exists only while the macro executes.
If no local symbol table exists when the SYMPUT routine executes, it creates a global macro variable.

You have already learned about using parameters in macro definitions. You should also already be familiar with the %LET statement, the SYMPUT routine, and the INTO clause. Let's examine the %LOCAL statement.

- The %LOCAL Statement

- can appear only inside a macro definition
- creates one or more macro variables in the local symbol table and assigns null values to them
- has no effect on variables that are already in the local symbol table.

A local symbol table is not created until a request is made to create a local variable. Macros that do not create local variables do not have a local table. Remember, the SYMPUT routine can create local variables only if the local table already exists.
Since local symbol tables exist separately from the global symbol table, it is possible to have a local macro variable and a global macro variable that have the same name and different values.

- Writing the macro variables into the SAS Log

ALL
AUTOMATIC
GLOBAL
LOCAL: can only display when defined within macro
USER

Example 33:
The following SAS program is submitted:

```
%let Mv=shoes;

%macro PRODUCT(Mv=bicycles);
   %let Mv=clothes;
%mend;

%PRODUCT(Mv=tents)
%put Mv is &Mv;
```

What is written to the SAS log?
A. Mv is bicycles
B. Mv is clothes
C. Mv is shoes
D. Mv is tents

Example 34:
Given the following macro program and invocation:

```
%macro MAKEPGM(NEWNAME, SETNAME);
  data &NEWNAME;
    set &SETNAME;
  run;
  %put ---> inside macro &NEWNAME &SETNAME;
%mend;

%MAKEPGM(WORK.NEW, SASHELP.CLASS)
%put ---> outside invocation &NEWNAME &SETNAME;
```

Which of these choices shows the correct %PUT statement output if the program is submitted at the beginning of a new SAS session? Note that other lines may be written to the SAS log by the program, but only the %PUT output is shown here.

A. ---> inside macro WORK.NEW SASHELP.CLASS
---> outside invocation WORK.NEW SASHELP.CLASS
B. ---> inside macro WORK.NEW SASHELP.CLASS
---> outside invocation &NEWNAME &SETNAME
C. ---> inside macro &NEWNAME &SETNAME
---> outside invocation WORK.NEW SASHELP.CLASS
D. ---> inside macro &NEWNAME &SETNAME
---> outside invocation &NEWNAME &SETNAME

SAS Automatic Macro Variables

- SAS creates and defines several automatic macro variables.
- Automatic macro variables contain information about your computing environment, such as the date and time of the session, and the version of SAS that you are running.

- Created when SAS is invoked
- Global (always available)
- Assigned values by SAS/User

- Some automatic macro variables have fixed values that are set when SAS is invoked.

Name	Value
SYSDATE	the date of the SAS invocation (DATE7.)
SYSDATE9	the date of the SAS invocation (DATE9.)
SYSDAY	the day of the week of the SAS invocation
SYSTIME	the time of the SAS invocation
SYSENV	FORE (interactive execution) or BACK (noninteractive or batch execution)
SYSSCP	an abbreviation for the operating system that is being used, such as WIN or LINUX
SYSVER	the release of SAS that is being used
SYSJOBID	an identifier for the current SAS session or for the current batch job (the user ID or job name for mainframe systems, the process ID (PID) for other systems)

```
footnote1 "Created &systime &sysday, &sysdate9";
footnote2 "on the &sysscp system using Release &sysver";
title "REVENUES FOR DALLAS TRAINING CENTER";
proc tabulate data=sasuser.all(keep=location course_title fee);
where upcase(location)="DALLAS";
class course_title;
var fee;
table course_title=" " all="TOTALS",
fee=" "*(n*f=3. sum*f=dollar10.)
/ rts=30 box="COURSE";
run;
```

COURSE	N	Sum
Artificial Intelligence	25	$10,000
Basic Telecommunications	18	$14,310
Computer Aided Design	19	$30,400
Database Design	23	$8,625
Local Area Networks	24	$15,600
Structured Query Language	24	$27,600
TOTALS	133	$106,535

Created 08:37[1] Wednesday[2], 20APR2011[3]
on the WIN[4] system using Release 9.3[5]

Example 35:

Which title statement would always display the current date?

A. title "Today is: &sysdate.";
B. title "Today is: &sysdate9.";
C. title "Today is: &today.";
D. title "Today is: %sysfunc(today(),worddate.)";

Example 36:

The following SAS program is submitted:

```
%macro execute;
  [_insert_statement_here_]
    proc print data=SASUSER.HOUSES;
    run;
  %end;
%mend execute;
%execute
```

Which statement completes the program so that the PROC PRINT step executes on Thursday?

A. if &sysday = Thursday then %do;
B. %if &sysday = Thursday %then %do;
C. %if "&sysday" = Thursday %then %do;
D. %if &sysday = "Thursday" %then %do;

- Some automatic macro variables have values that automatically change based on submitted SAS statements.

Name	Value
SYSLAST	the name of the most recently created SAS data set, in the form LIBREF.NAME. This value is always stored in all capital letters. If no data set has been created, the value is _NULL_; Note:Throughout this book, the keyword _NULL_ is often used in place of the data set name in sample programs. Using _NULL_ suppresses the creation of an output data set. Using _NULL_ when benchmarking enables you to determine what resources are used to read a SAS data set.
SYSPARM	text that is specified when SAS is invoked
SYSERR	contains a return code status that is set by the DATA step and some SAS procedures to indicate whether the step or procedure executed successfully

- Automatic macro variables are stored in the global symbol table.
- User-defined macro variables that you create with a %LET statement in open code (code that is outside of a macro definition) are also stored in the global symbol table.

Global Symbol Table	
SYSDATE	*04APR11*
SYSDAY	*Monday*
SYSVER	*9.2*
uservar1	*value1*
uservar2	*value2*

- The global symbol table is created during the initialization of a SAS session and is deleted at the end of the session.

- Macro variables in the global symbol table
  - are available anytime during the session
  - can be created by a user
  - have values that can be changed during the session (except for some automatic macro variables).

- The %SYMDEL statement

- You use the %SYMDEL statement to delete a macro variable from the global symbol table during a SAS session.
- To remove the macro variable **var** from the global symbol table, you submit the following statement:

```
%symdel var;
```

Example 37:
Which macro statement would remove the macro variable Mv_Info from the symbol table?
A. %mdelete &Mv_Info;
B. %symerase Mv_Info;
C. %symdel &Mv_Info;
D. %symdel Mv_Info;