Various Macro Loops

- Iterative %DO statement
- %DO %UNTIL statement
- %DO %WHILE statement

**Warning:** Do not confuse macro loops with DATA step loops. Macro loops will generate SAS codes (combination of code fragments, DATA steps, PROC steps, and stand-alone statements) repetitively.
Iterative %DO Statement

%DO macro-variable=start %TO stop <%BY increment>;
   text and macro statements
%END;

- **macro-variable**: macro variable name or macro expression that generates a macro variable
- **start**: integer or macro expression that generates integer
- **stop**: integer or macro expression that generates integer
- **increment**: integer or macro expression that generates integer (other than 0, and default is 1)

**Warning:**
1. **start**, **stop**, and **increment** are calculated before loop starts, and you cannot change them during loop execution
2. if you need value of index variable after last iteration, do
   \[
   \%EVAL(start+increment*((stop-start)/increment+1))
   \]
3. %UNTIL and %WHILE clauses not allowed.

%DO %UNTIL Statement

%DO %UNTIL(expression);
   text and macro statements
%END;

where **expression** can be any macro expression.
The text and macro statements are first executed then the condition is checked to determine to continue (if false) / terminate (if true) the loop
%DO %WHILE Statement

%DO %WHILE(expression);
    text and macro statements
%END;

where expression can be any macro expression.
The condition is checked first to continue (if true) / terminate (if false) the execution of the text and macro statements.

%DO Statement

%DO;
    text and macro statements
%END;

Not a genuine %DO loop, it works similarly as DATA step DO-group.
%GOTO or %GO TO Statement

%GOTO | %GO TO label;
where label is a label or a macro expression that generates a label. Examples:

- %GOTO special;
- %GOTO &this_label;
- %GO TO %look();
- in a macro:
  %MACRO mymacro(parameters);
    text and macro statements
  %IF &code=2 %THEN %GOTO out;
    text and macro statements
  %out:  %MEND mymacro;

%label Statement

%label:  macro-text
where

- label: any SAS name
- macro-text: macro program statement or macro expression.
Conditional Execution
by using %IF - %THEN / %ELSE statements

%IF expression1 %THEN expression2;
<%ELSE expression2;>
where

- expression1: macro expression that yields a logical expression with nonzero numeric value = true, zero numeric value = false, and character (non-null or null) value = expression2 not executed with error message
- expression2: macro expression (text or macro program statement) that will be executed if expression1 has true value.

An Example

%MACRO info(data=&SYSLAST,type=long,obs=10);
%IF %UPCASE(&type)=SHORT %THEN %GOTO peek;
PROC CONTENTS DATA=&data;
RUN;
PROC FREQ DATA=&data;
   TABLES _NUMERIC_;  
RUN;
%peek: PROC PRINT DATA=&data(obs=&obs);
RUN;
%MEND info;
Execute System Command
Under Windows

%SYSEXEC <system-command>;
E.g.,
%SYSEXEC time
where time is a command in WINDOWS

EXECUTE Call Routine

EXECUTE is a DATA step call routine that is used to resolve its argument and executes the resolved value at the next step boundary.

CALL EXECUTE(argument)
where argument can be one of:

▶ quoted string (single quotes used: resolves during execution; double quotes used: resolves during compilation)

▶ unquoted DATA step character variable whose values are SAS statement (so this is unrelated to macro)

▶ character expression to be resolved to a macro expression or a SAS statement
CALL EXECUTE Examples

▶ call execute('%aov');
▶ call execute(do_sort);
▶ call execute('%aov(' || varlist || '))';

A Complete CALL EXECUTE Example

%macro overdue;
   proc print data=late;
      title "Overdue Accounts As of &sysdate";
      run;
%mend overdue;

data late;
   set sasuser.billed end=final;
   if datedue<=today()-30 then do;
      n+1;
      output;
   end;
   if final and n then call execute('%overdue');
run;