

Five Ways to Create Macro Variables: A Short Introduction to the Macro Language

Arthur L. Carpenter

**California Occidental Consultants
Oceanside, CA**

Overview

- **There are a number of ways to create macro variables.**
- **%LET statement**
- **Macro parameters**
- **Iterative %DO statement**
- **Using the INTO in PROC SQL**
- **Using the CALL SYMPUT routine**

%LET

- The %LET statement is used to define a macro variable
- Similar to the DATA step assignment statement

```
%let macro_var_name = value;
```

```
%let dsn = clinics;
```

%LET

- The macro language does not support the concept of missing values

```
%let nada =;  
%let nada =          ;
```

```
%let dset =clinics;  
%let dset =          clinics      ;
```

EXERCISE # 1

- Create a macro variable named &DSN that contains SASHELP.CLASS

```
%let xxxx = xxxx;
```

```
%let dsn = sashelp.class;
```

USING MACRO VARIABLES

- When used, macro variables are preceded with an ampersand (&).
- When used in titles, macro variables are enclosed in double quotes.

```
proc means data=&dset;  
var ht wt;  
title1 "Means of &dset";  
run;
```

EXERCISE # 2

- Use the macro variable named &DSN in a PROC PRINT and a title.

```
proc print data=xxxx;  
title 'the data is xxxx';
```

```
proc print data=&dsn;  
title1 "the data is &dsn";  
run;
```

USING %PUT

- %PUT writes to the LOG.

```
%put the data set is &dset;
```

```
%put _user_;
```

```
%put _global_;
```

```
%put _automatic_;
```

```
%put _all_;
```

```
%put WARNING: data set is &dset.
```

```
%put ERROR: xxxx;
```

```
%put NOTE: xxxxxx;
```


EXERCISES # 3 and 4

- Use the %PUT statement to:
 - Write: The data set is SASHELP.CLASS
 - A list of all user defined macro variables

```
%put The data set is xxxx;  
%put xxxx;
```

```
%put The data set is &dsn;  
%put _user_;
```

POSITIONAL PARAMETERS

- Used to pass information into a macro.

```
%macro printit(dset, varlist);  
    PROC PRINT DATA=&dset;  
        var &varlist;  
        TITLE1 "Listing of &dset";  
    RUN;  
%mend printit;  
%printit(clinics, lname ht wt)
```

- Order of parameters is crucial.

EXERCISE # 5

- Define macro variables so that %LOOK:
 - Uses the data set SASHELP.CLASS
 - Prints out 10 observations

```
%MACRO LOOK (xxxx,yyyy) ;  
    PROC CONTENTS DATA=xxxx ;  
        TITLE "DATA SET xxxx" ;  
    RUN ;  
    PROC PRINT DATA=xxxx (OBS=yyyy) ;  
        TITLE2 "FIRST yyyy OBSERVATIONS" ;  
    RUN ;  
%MEND LOOK ;  
%look (aaaa,bbbb)
```

EXERCISE # 5

- Define macro parameters

```
%MACRO LOOK (dsn, obs) ;  
  PROC CONTENTS DATA=&dsn ;  
    TITLE "DATA SET &dsn" ;  
  RUN ;  
  PROC PRINT DATA=&dsn (OBS=&obs) ;  
    TITLE2 "FIRST &obs OBSERVATIONS" ;  
  RUN ;  
%MEND LOOK ;  
%look(sashelp.class, 10)
```

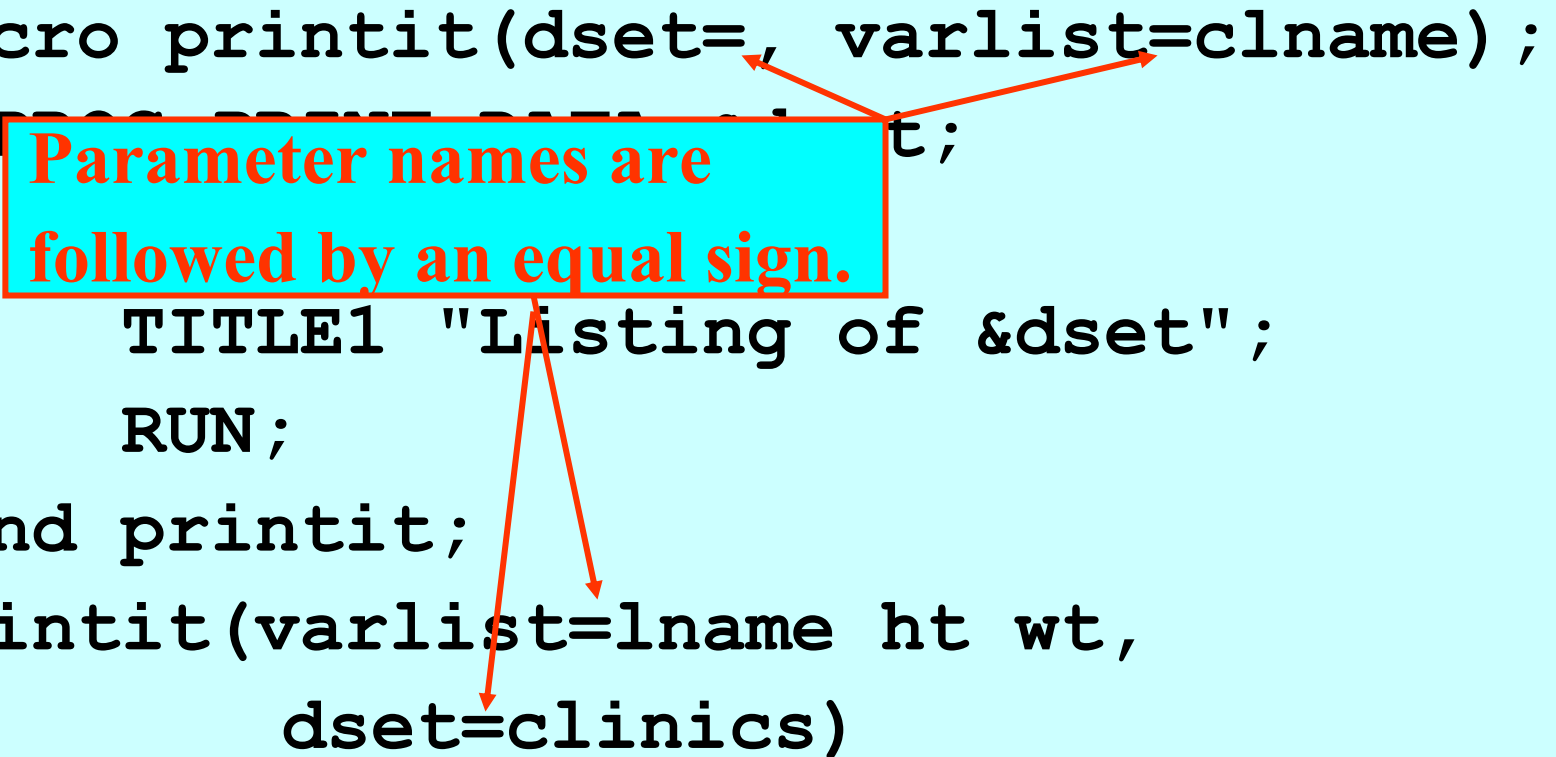
The diagram illustrates the macro definition and its usage. Red arrows show the substitution of the macro parameter `dsn` with `&dsn` in the `PROC CONTENTS` and `PROC PRINT` statements, and the substitution of the macro parameter `obs` with `&obs` in the `PROC PRINT` statement. The macro call `%look(sashelp.class, 10)` is shown at the bottom, with `sashelp.class` circled in red.

KEYWORD PARAMETERS

- Parameters are named in the %MACRO statement.

```
%macro printit(dset=, varlist=cname) ;  
  proc print data=dset ;  
    TITLE1 "Listing of &dset" ;  
    RUN ;  
%mend printit ;  
%printit(varlist=lname ht wt,  
        dset=clinics)
```

Parameter names are followed by an equal sign.



EXERCISE # 6

- Replace Exercise #5 positional parameters with Keyword= parms.

```
%MACRO LOOK (xxxx,yyyy) ;  
    PROC CONTENTS DATA=xxxx ;  
        TITLE "DATA SET xxxx" ;  
    RUN ;  
    PROC PRINT DATA=xxxx (OBS=yyyy) ;  
        TITLE2 "FIRST yyyy OBSERVATIONS" ;  
    RUN ;  
%MEND LOOK ;  
%look (aaaa,bbbb)
```

EXERCISE # 6

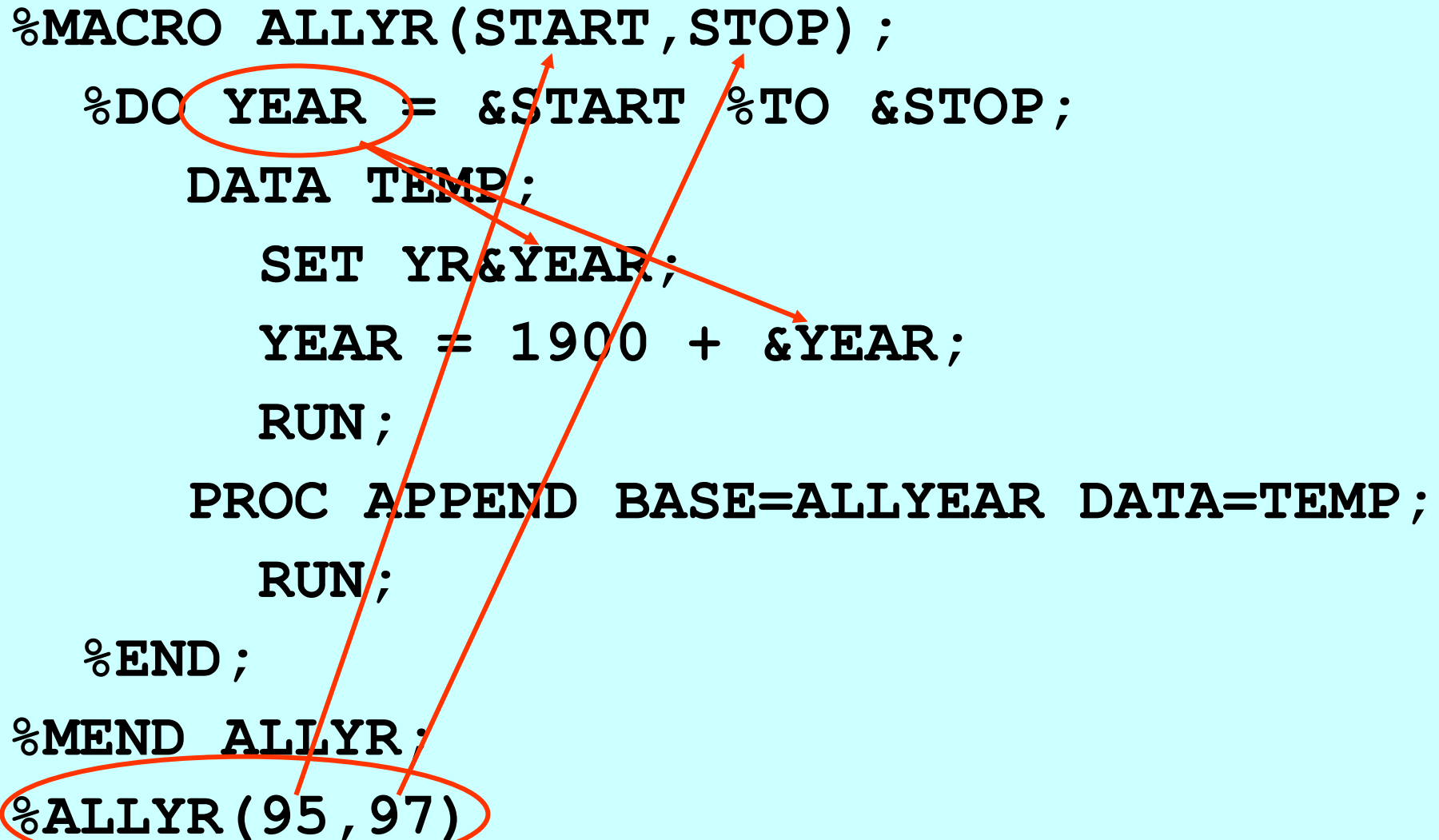
```
%MACRO LOOK (dsn=, obs=5) ;  
    PROC CONTENTS DATA=&dsn ;  
        TITLE "DATA SET &dsn" ;  
    RUN ;  
    PROC PRINT DATA=&dsn (OBS=&obs) ;  
        TITLE2 "FIRST &obs OBSERVATIONS" ;  
    RUN ;  
%MEND LOOK ;  
  
%look (dsn=sashelp.class, obs=10)
```

ITERATIVE %DO LOOP

- The %DO loop is similar to the DO in the DATA step.
- The index variable is a macro variable.
- The %DO is always completed with an %END.

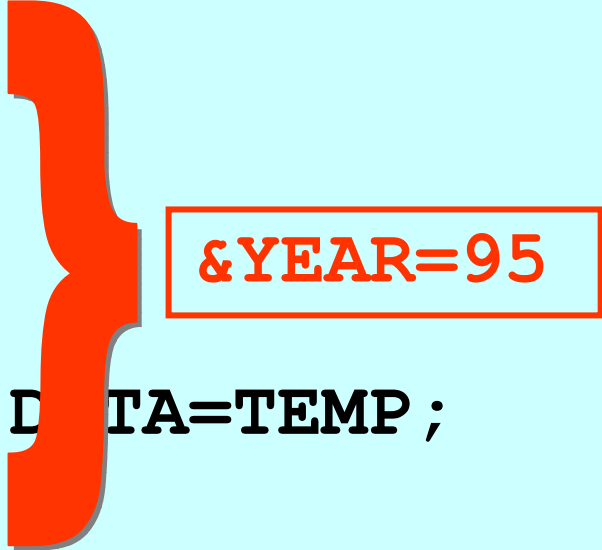
ITERATIVE %DO LOOP

```
%MACRO ALLYR (START, STOP) ;  
  %DO YEAR = &START %TO &STOP;  
    DATA TEMP;  
      SET YR&YEAR;  
      YEAR = 1900 + &YEAR;  
      RUN;  
    PROC APPEND BASE=ALLYEAR DATA=TEMP;  
      RUN;  
  %END;  
%MEND ALLYR;  
%ALLYR (95, 97)
```



ITERATIVE %DO LOOP

```
DATA TEMP;  
  SET YR95;  
  YEAR = 1900 + 95;  
  RUN;  
PROC APPEND BASE=ALLYEAR DATA=TEMP;  
  RUN;
```



```
DATA TEMP;  
  SET YR96;  
  YEAR = 1900 + 96;  
  RUN;
```

EXERCISE # 7

Write a macro that will break up the SASHELP.CLASS data set into one data set for each age. Pass the lowest and highest ages as macro parameters.

```
%macro dsbyage(start=10, stop=18);  
    %do xxxx = xxxx %to xxxx;  
        data agexxxx;  
            set sashelp.class;  
            where age = xxxx;  
            run;  
    %end;  
%mend dsbyage;  
%dsbyage(start=12, stop=15)
```

EXERCISE # 7

Write a macro that will break up the SASHELP.CLASS data set into one data set for each age. Pass the lowest and highest ages as macro parameters.

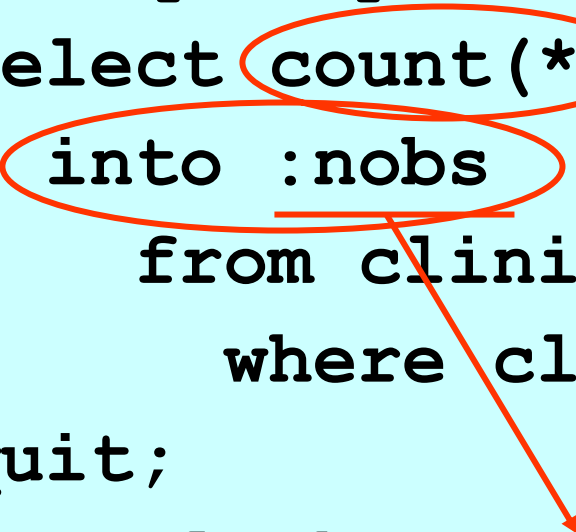
```
%macro dsbyage(start=10, stop=18);  
  %do age = &start %to &stop;  
    data age&age;  
      set sashelp.class;  
      where age = &age;  
    run;  
  %end;  
%mend dsbyage;  
%dsbyage(start=12, stop=15)
```

```
data age12;  
  set sashelp.class;  
  where age = 12;  
run;
```

USING INTO IN PROC SQL

- INTO : creates a macro variable

```
%let cln = Beth;  
proc sql noprint;  
    select count(*)  
        into :nobs  
        from clinics  
        where clinname="&cln";  
quit;  
%put &cln has &nobs clinics;
```



EXERCISE # 8

Write a macro that counts the number of students of a given age and places the count in a macro variable.

```
%macro agecnt(age=12) ;  
  proc sql noprint;  
    select count(****)  
      into : ****  
        from sashelp.class  
        where age=****;  
  quit;  
  %put The count for age **** is ****;  
%mend agecnt;  
%agecnt(****=14)
```

EXERCISE # 8

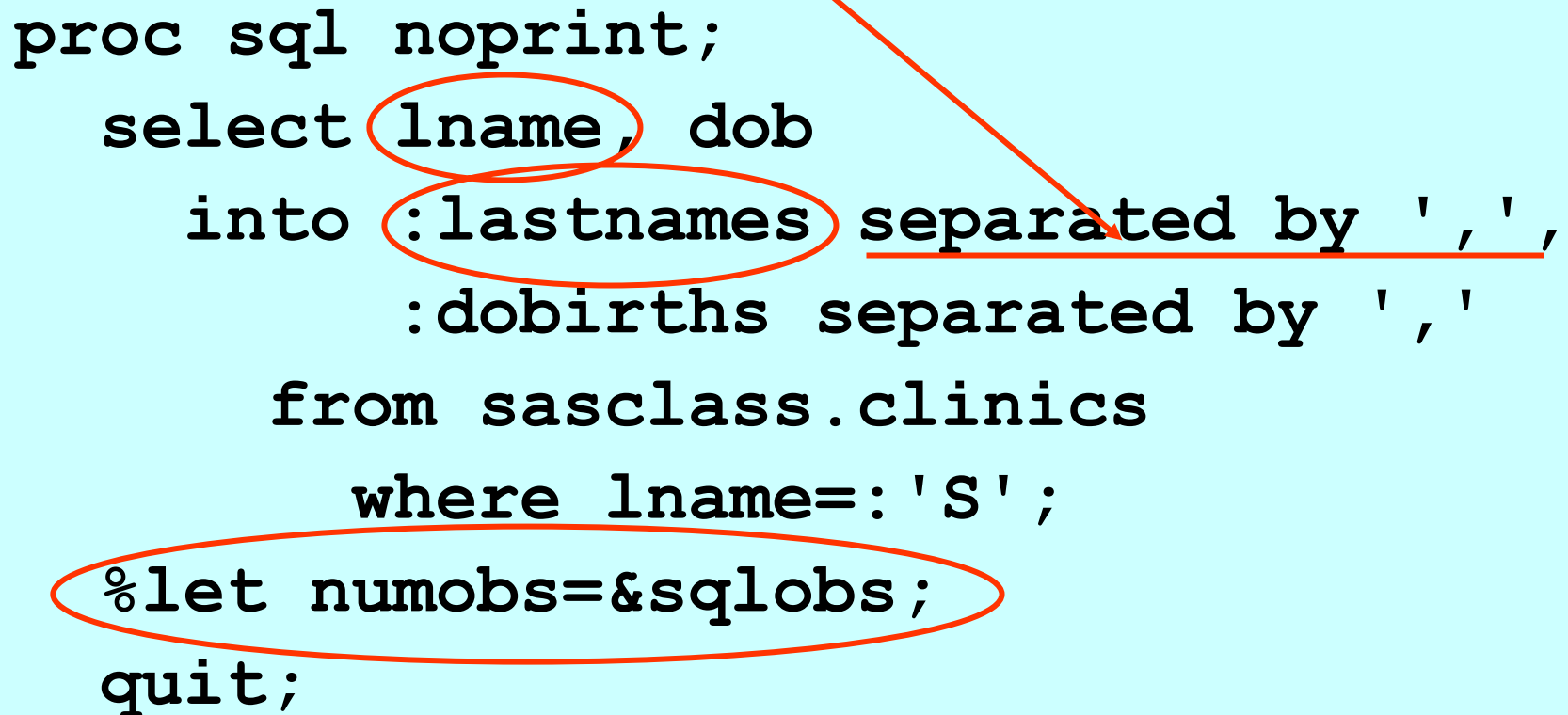
Using PROC SQL write a macro that counts the number of students of a given age and places the count in a macro variable.

```
%macro agecnt(age=12) ;  
  proc sql noprint;  
    select count(age)  
      into : agecnt  
        from sashelp.class  
        where age=&age;  
  quit;  
  %put The count for age &age is &agecnt;  
%mend agecnt;  
%agecnt(age=14)
```

CREATING LISTS WITH SQL

- The phrase *separated by* creates a list

```
proc sql noprint;  
  select lname, dob  
    into :lastnames separated by ',',  
         :dobirths separated by ','  
    from sasclass.clinics  
    where lname=: 'S';  
%let numobs=&sqlobs;  
quit;
```



EXERCISE # 9

Write a macro that places a list of the unique values of a variable, whose name is passed as a parameter, into a single macro variable.

```
%macro vlist(vname=name) ;  
  proc sql noprint;  
    select distinct(!!!!)  
      into : !!!! separated by ' '  
    from sashelp.class;  
  quit;  
  %put !!!! takes on the values of !!!!;  
%mend vlist;  
%vlist(vname=age)
```

EXERCISE # 9

Create a macro variable that contains a list.

```
%macro agecnt(age=12) ;  
  proc sql noprint;  
    select count(age)  
      into : agecnt  
      from sashelp.class  
      where age=&age;  
  quit;  
  %put The count for age &age is &agecnt;  
%mend agecnt;  
%agecnt(age=14)
```

USING THE SYMPUT ROUTINE

- The %LET cannot be used within a DATA step.
- The %LET is executed before the DATA step is compiled.

```
data new;  
  set old(where=name='FRED') ;  
  %let fredage = age;  
run;
```

- &FREDAGE contains the letters a-g-e

USING THE SYMPUT ROUTINE

- A DATA step routine executes when the DATA step executes.
- The DATA step writes to the macro variable SYMPUT.

```
data new;  
  set old(where=name='FRED') ;  
  call symput('fredage',age) ;  
run;
```

- &FREDAGE contains the AGE.

Value for the macro variable

EXERCISE # 10

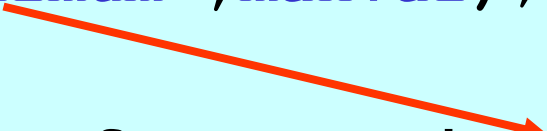
Use a DATA step to store the maximum value of a numeric DATA step variable in a macro variable.

```
%macro maxval(vname=weight) ;  
  data maxvalue;  
    set sashelp.class;  
    retain maxval .;  
    maxval = max(maxval, &vname) ;  
    call symput(****, ****) ;  
  run;  
  %put The maximum value of **** is ****;  
%mend maxval;  
%maxval(vname=age)
```

EXERCISE # 10

Use a DATA step to store the maximum value of a numeric DATA step variable in a macro variable.

```
%macro maxval(vname=weight) ;  
  data maxvalue;  
    set sashelp.class;  
    retain maxval .;  
    maxval = max(maxval,&vname) ;  
    call symput('maximum',maxval) ;  
  run;  
  %put Maximum value of &vname is &maximum;  
%mend maxval;  
%maxval(vname=age)
```



CREATING A LIST USING SYMPUT

- A character variable acts as an index.
- Concatenate variable name with index value

```
data _;
```

```
se
```

```
cnt = left(put(_n_,6.));
```

```
call symput('name' || cnt, lname);
```

```
call symput('namecnt', cnt);
```

```
run;
```

Root portion of
the macro
variable name

Index portion
of the macro
variable name

Total number of
names

- Creates: &NAME E3,

EXERCISE # 11

```
%macro namelist;  
  data _null_;  
    set sashelp.class;  
    by name;  
    if first.name then do;  
      cnt+1;  
      call symput('xxxx' || left(put(cnt,5.)), name) ;  
      call symput('xxxx', left(put(cnt,5.))) ;  
    end;  
  run;  
  
  %put num   Name;  
  %do xxxx = 1 %to &xxxx;  
    %put &xxxx      &&xxxx&xxxx;  
  %end;  
%mend namelist;  
%namelist
```


EXERCISE # 11

```
%macro namelist;  
  data _null_;  
    set sashelp.class;  
  by name;  
  if first.name then do;  
    cnt+1;  
    call symput('name' ||  
      left(put(cnt,5.)),name) ;  
    call symput('namecnt',  
      5.))) ;  
    &name1, &name2, &name3, ...  
  run;
```

EXERCISE # 11

```
%put num Name;
%do namenum = 1 %to &namecnt;
    %put &namenum &&name&namenum;
%end;
```

```
%put The list of u
%put Macro vars is
%put _user_;
%mend namelist;
%namelist
```

&&name&namenum

&&name &namenum

& name 3

&name3

Barbara

Summary

- There are a number of ways to create macro variables.
- %LET assigns values directly.
- Macro parameters create local variables.
- The index of a %DO loop is a macro variable.
- The INTO : can be used in PROC SQL
- CALL SYMPUT writes to the symbol table from within a DATA step.

Five Ways to Create Macro Variables: A Short Introduction to the Macro Language

Arthur L. Carpenter

California Occidental Consultants

P.O. Box 430

Oceanside, CA 92085-0430

(760) 945-0613

art@caloxy.com www.caloxy.com

SAS is a registered trademark of SAS Institute Inc. in the USA and other countries.

® indicates USA registration.