Creating Macro Variables in the DATA Step
Creating Macro Variables

• In some applications it is desirable to be able to assign macro variable values based on values stored in a SAS data set.

• We have used the `%let` statement to assign values to macro variables previously, but it is important to remember how macro statements are processed...
Creating Macro Variables

• Consider:

```sas
%let crsnum=3;
data revenue;
set mysas.register end=final;
where course_number=&crsnum;
total+1;
if paid='Y' then paidup+1;
if final then do;
  put total= paidup=;
  if paidup < total then do;
    %let foot=Some Fees Due;
    end;
  else do;
    %let foot=All Students Paid;
    end;
end;
run;
```
Creating Macro Variables

```
proc print data=revenue noobs;
    var student_name paid;
    title "Paid Status for Course &crsnum";
    footnote "&foot";
run;
```
Creating Macro Variables

```sas
proc print data=revenue noobs;
  var student_name paid;
  title "Paid Status for Course &crsnum";
  footnote "&foot";
run;
```

- Output (partial):

```
Paid Status for Course 3

<table>
<thead>
<tr>
<th>Student_Name</th>
<th>Paid</th>
</tr>
</thead>
<tbody>
<tr>
<td>Bills, Ms. Paulette</td>
<td>Y</td>
</tr>
<tr>
<td>Chevarley, Ms. Arlene</td>
<td>N</td>
</tr>
<tr>
<td>Clough, Ms. Patti</td>
<td>N</td>
</tr>
<tr>
<td>...</td>
<td></td>
</tr>
<tr>
<td>Sulzbach, Mr. Bill</td>
<td>Y</td>
</tr>
<tr>
<td>Williams, Mr. Gene</td>
<td>Y</td>
</tr>
</tbody>
</table>

All Students Paid
```
Creating Macro Variables

• Obviously, this did not work as intended...
• What form of the if-then-else structure is sent to the SAS compiler?
Creating Macro Variables

- Obviously, this did not work as intended...
- What form of the if-then-else structure is sent to the SAS compiler?
  - This:
    ```sas
    if paidup < total then do;
    end;
    else do;
    end;
    ```
Creating Macro Variables

• Obviously, this did not work as intended...
• What form of the if-then-else structure is sent to the SAS compiler?
  ▫ This:
    ```
    if paidup < total then do;
    end;
    else do;
    end;
    ```
• The `%let` statements are executed by the macro processor (at the word scanning phase) before the data step is compiled and executes.
The SYMPUT Routine

• The SYMPUT routine can be used to create a macro variable and assign its value

  \texttt{call symput(macro-variable, value);}
The SYMPUT Routine

• The SYMPUT routine can be used to create a macro variable and assign its value
  
  call syput(macro-variable, value);

  ▫ The *macro-variable* can be specified as a:
    • Literal (quoted string)
    • DATA step variable
    • DATA step expression
    • It must form a legal macro variable name
The SYMPUT Routine

- Using literals in the SYMPUT routine you can
  - Specify the exact name for the macro variable as the first argument.
  - Specify the exact value of the macro variable as the second argument.
The SYMPUT Routine

- Using literals in the SYMPUT routine you can
  - Specify the exact name for the macro variable as the first argument.
  - Specify the exact value of the macro variable as the second argument.
  - A re-write of our first attempt:
    ```java
    if paidup < total then do;
        call symput('foot','Some Fees Due');
    end;
    else do;
        call symput('foot','All Students Paid');
    end;
    ```
The SYMPUT Routine

- This version uses SYMPUT with a literal for the variable name and a data step variable as the value.

```plaintext
%let crsnum=3;

data revenue;
...
  if final then do;
    call symput('paid',paidup);
    call symput('total',total);
  end;
run;

proc print data=revenue noobs;
  var student_name paid;
  title "Paid Status for Course &crsnum";
  footnote "Note: &paid paid out of &total students";
run;
```
The SYMPUT Routine

- We can also use a data step expression to set the value...
  - A better looking footnote can be had with this:
    ```
    if final then do;
    call symput('paid',trim(left(paidup)));
    call symput('total',trim(left(total)));
    end;
    ```
  - Default storage of numeric values is done with a conversion to BEST12. format.
The SYMPUT Routine

- The PUT function can be useful in defining a specific format for storage.

\[
\text{PUT}(source, \text{format})
\]

- \textit{Source} is a constant, variable or expression
- \textit{Format} is any SAS format or user-defined format.
- E.G.:
The SYMPUT Routine

- The PUT function can be useful in defining a specific format for storage.

**PUT(source, format)**

- *Source* is a constant, variable or expression
- *Format* is any SAS format or user-defined format.
- E.G.:

```sas
data revenue;
  set mysas.register end=final;
  ...
  if final then do;
    ...
    call symput('owed', put(650*(total-paidup),dollar8.));
  end;
run;
proc print data=revenue noobs;
  ...
  footnote2 "Note: &owed in unpaid fees";
run;
```
Multiple Macro Variables with SYMPUT

• Consider the following code:

```sas
%let crsid=C005;

data _null_;  
  set mysas.courses;
  where course_code="&crsid";
  call symput('title',trim(course_title));
run;

proc print data=mysas.schedule noobs label;
  where course_code="&crsid";
  var location begin_date teacher;
  title1 "Schedule for &title";
run;
```
Multiple Macro Variables with SYMPUT

• In this case, a separate run of the DATA step are not necessary for each course

• Consider the following:

  data _null_;  
  set mysas.courses;  
  call symput(course_code, trim(course_title));  
  run;  
  %put _user_;
Multiple Macro Variables with SYMPUT

- The first argument of SYMPUT is now a data step variable, whose values are legal macro variable names.
- If I wish to use these macro variables to assign my titles, how do I accomplish it?
Indirect References

• Suppose we have:

```sas
%let crsid=C005;
proc print data=mysas.schedule noobs label;
  where course_code="&crsid";
  var location begin_date teacher;
  title1 "Schedule for   ";
run;
```
Indirect References

• Suppose we have:

```sas
%let crsid=C005;

proc print data=mysas.schedule noobs label;
  where course_code="&crsid";
  var location begin_date teacher;
  title1 "Schedule for ";
run;
```

What goes here in order to get the course title to display?
Indirect References

- To get the course title to appear in place of the course code, logic dictates
  - we resolve &crsid to “C005”
  - we would like to have an additional & in front so we can resolve &C005 to “Artificial Intelligence”.
  - It seems that the appropriate reference would thus be &&crsid
  - but, it’s wrong...
Indirect References

- Two important rules in resolving macro variable references
  - Multiple ampersands or percent signs in front of a name token cause the reference to be rescanned
  - && resolves to &
Indirect References

• Two important rules in resolving macro variable references
  ▫ Multiple ampersands or percent signs in front of a name token cause the reference to be rescanned
  ▫ && resolves to &

• The correct solution is to use the reference &&&crsid
Indirect References

• Behind the scenes...

Artificial Intelligence
Indirect References

• Is a && ever useful?
  ▫ Consider:

```sas
data _null_
  set mysas.schedule;
  call symput(cats("teach",put(course_number,2.),trim(teacher))); run;
```
Indirect References

• Is a && ever useful?
  ▫ Consider:

```sas
data _null_;  
  set mysas.schedule;  
  call symput(cats("teach",put(course_number,2.),trim(teacher)));  
run;

%let crs=3;
proc print data=mysas.register noobs;
  where course_number=&crs;
  var student_name paid;
  title1 "Roster for Course &crs";
  title2 "Taught by ";
run;
```
Indirect References

- Is a && ever useful?
  - Consider:

```sas
data _null_
set mysas.schedule;
call symput(cats("teach",put(course_number,2.),trim(teacher)));
run;

%let crs=3;
proc print data=mysas.register noobs;
  where course_number=&crs;
  var student_name paid;
  title1 "Roster for Course &crs";
  title2 "Taught by ";
run;
```

What reference should be here?
Indirect References

• By the previous rules
  &teach&crs
Indirect References

• By the previous rules
  \&\&teach\&crs

\&teach_3
Indirect References

• By the previous rules

&teach&crs

&teach3

Forest, Mr. Peter