The George Washington University
Department of Statistics

STAT 4197/6197

Week 2: Reading Data into SAS and Creating Reports

1. Reading Raw Data into SAS Data Sets Using
    a. Column Input
    b. Formatted Input
    c. List Input
    d. Named Input
2. Reading Microsoft Excel Data into SAS Data Sets
3. Handling Missing Values
4. SAS Processing
5. Creating Raw Tabular Data Files or Reports Using DATA _NULL_
6. Creating a SAS List Report

Acknowlwdgements: Portions of SAS' copyrighted SAS course content are reproduced here with permission of SAS Institute Inc., Cary, NC, USA. SAS Institute Inc. makes no warranties with respect to these materials and disclaims Neither the GW nor the instructor shall be held liable or responsible to any person or entity with respect to any loss or incidental or consequential damages caused by anyone's use of the information or SAS codes contained herein.

Reading Raw Data with the Input Statement

Use the above link to study the SAS codes for various input styles.

Input Styles

Column input, formatted input, list input, and named input are all styles of writing INPUT statement specifications.

Style	Use for Reading
Column input	Standard data in fixed columns
Formatted input	Standard and nonstandard data in fixed columns
List input	Standard and nonstandard data separated by blanks or some other delimiter
Named input	Standard data that is preceded by the name of the variable and an equal sign (=)

7

The list input style that handles standard data and character variables whose values are no longer than 8 bytes is called *simple list input*.

The list input style that additionally handles nonstandard data and character variables whose values are more than 8 bytes long is called *modified list input*.

Modified List Input = List input + some features of Formatted Input.

Acknowledgments: Portions of SAS' copyrighted SAS course content are reproduced here with permission of SAS Institute Inc., Cary, NC, USA. SAS Institute Inc. makes no warranties with respect to these materials and disclaims all liability therefor. Neither the GW nor the instructor shall be held liable or responsible to any person or entity with respect to any loss or incidental or consequential damages caused by anyone's use of the information or SAS codes contained herein. Circulating the handouts is strictly forbidden.

When to Use Column Input

Reading Fixed Column Files

Column input is used to read standard values from fixed column data files.

Partial **salesemp.dat**

```
          1    1    2    2    3    3    4    4    5    5    6
1---5----0----5----0----5----0----5----0----5----0----5----0--
120102 Tom        Zhou           Sales Manager       108255 AU
120103 Wilson     Dawes          Sales Manager       87975  AU
120121 Irenie     Elvish         Sales Rep. II       26600  AU
120122 Christina  Ngan           Sales Rep. II       27475  AU
120123 Kimiko     Hotstone       Sales Rep. I        26190  AU
```

Formatted input is used to read standard and nonstandard values from fixed column data files.

16

Using Column Input

Column input requires a variable name, type, starting column, and ending column for each field to be read.

Partial **salesemp.dat**

```
          1    1    2    2    3    3    4    4    5    5    6
1---5----0----5----0----5----0----5----0----5----0----5----0--
120102 Tom        Zhou           Sales Manager       108255 AU
120103 Wilson     Dawes          Sales Manager       87975  AU
```

Name: EmpID
Type: Numeric
Columns: 1-7

Field	Type	Starting Column	Ending Column
Employee ID	Numeric	1	7
First Name	Character	8	17
Last Name	Numeric	19	32
Job Title	Character	34	52
Salary	Numeric	54	59
Country	Character	61	62

17

Acknowledgments: Portions of SAS' copyrighted SAS course content are reproduced here with permission of SAS Institute Inc., Cary, NC, USA. SAS Institute Inc. makes no warranties with respect to these materials and disclaims all liability therefor. Neither the GW nor the instructor shall be held liable or responsible to any person or entity with respect to any loss or incidental or consequential damages caused by anyone's use of the information or SAS codes contained herein. Circulating the handouts is strictly forbidden.

Using Column Input

Use INFILE and INPUT statements in a DATA step to read a raw data file.

```
data work.sales;
   infile "&path\salesemp.dat";
   input EmpID 1-7 First_Name $ 8-17
         Last_Name $ 19-32 Job_Title $ 34-52
         Salary 54-59 Country $ 61-62;
run;
```

DATA *output-data-set*;
  INFILE "*raw-data-file*";
  INPUT *variable* <$> *start-end* …;
RUN;

p2aad02

18

INFILE Statement

The INFILE statement identifies the raw data file to be read.

```
infile "&path\salesemp.dat";
```

INFILE "*raw-data-file*";

- A full pathname is recommended.
- Using the **&path** macro variable reference makes the program more flexible.

Be sure to use double quotation marks when referencing a macro variable within a quoted string.

19

In the INFILE statement, you can specify either the external file's name or point to the external file with a fileref defined with the FILENAME statement, unless you have instream data following the DATALINES statement. You may also need to specify one or more options for this statement.

Acknowledgments: Portions of SAS' copyrighted SAS course content are reproduced here with permission of SAS Institute Inc., Cary, NC, USA. SAS Institute Inc. makes no warranties with respect to these materials and disclaims all liability therefor. Neither the GW nor the instructor shall be held liable or responsible to any person or entity with respect to any loss or incidental or consequential damages caused by anyone's use of the information or SAS codes contained herein. Circulating the handouts is strictly forbidden.

Advantages of Column Input

- Data fields can be read in any order. The same field can be reread.
- Parts of a data field can be read.
- Both leading and trailing blanks within the field are ignored.
- Character values can contain embedded blanks.
- Placeholders (e.g., period) are not required for missing data.

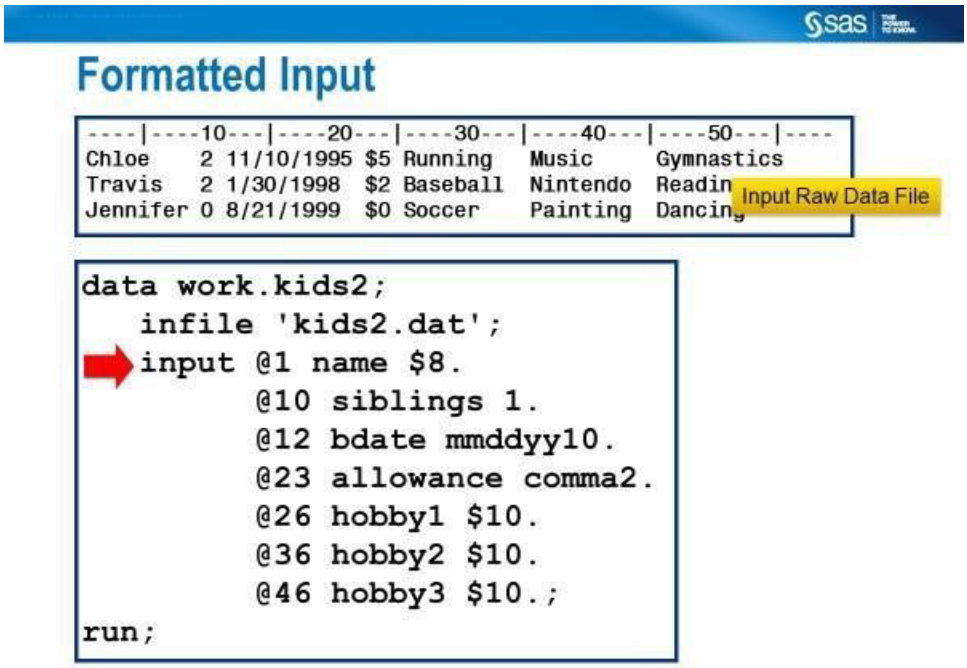
Disadvantage: The *Column Input* cannot read nonstandard data values

When to Use Formatted Input

Records with standard numeric, nonstandard numeric, or character data fields aligned in columns

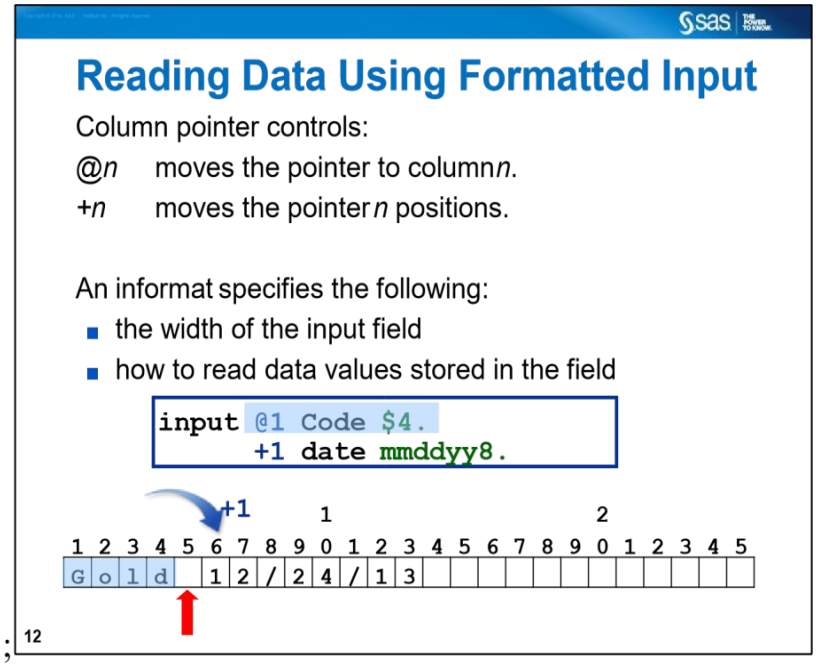
Features of the formatted input style

- Starting position
- Variable
- INFORMAT
  The informat name must contain a period (.) at the end for SAS to distinguish it (i.e., informat name) from a variable name, as seen in the code below.



Acknowledgments: Portions of SAS' copyrighted SAS course content are reproduced here with permission of SAS Institute Inc., Cary, NC, USA. SAS Institute Inc. makes no warranties with respect to these materials and disclaims all liability therefor. Neither the GW nor the instructor shall be held liable or responsible to any person or entity with respect to any loss or incidental or consequential damages caused by anyone's use of the information or SAS codes contained herein. Circulating the handouts is strictly forbidden.

Formatted Input: Use of Absolute Pointer Control vs. Relative Pointer Control

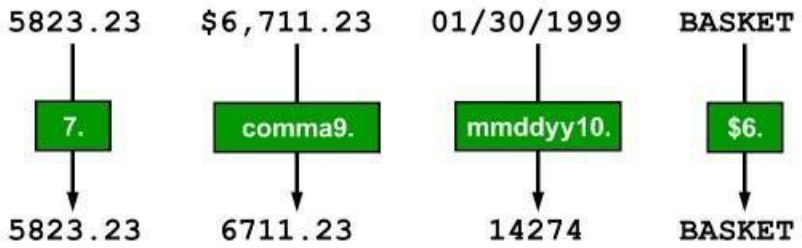


Remember that when you use formatted input, the column pointer control moves to the first column following the field that was just read.

Acknowledgments: Portions of SAS' copyrighted SAS course content are reproduced here with permission of SAS Institute Inc., Cary, NC, USA. SAS Institute Inc. makes no warranties with respect to these materials and disclaims all liability therefor. Neither the GW nor the instructor shall be held liable or responsible to any person or entity with respect to any loss or incidental or consequential damages caused by anyone's use of the information or SAS codes contained herein. Circulating the handouts is strictly forbidden.

Informats

An *informat* is an instruction that SAS uses to read data values into a variable.



SAS uses the informat to determine the following:

- whether the variable is numeric or character
- the length of character variables

48

a

Informats

Raw Data Value	Informat	SAS Data Value
$12,345	COMMA7. DOLLAR7.	12345
$12.345	COMMAX7. DOLLARX7.	12345
€12.345	EUROX7.	12345
Australia	$11.	Australia
Australia	$CHAR11.	Australia
au	$UPCASE2.	AU
01/01/1960	MMDDYY10.	0
31/12/60	DDMMYY8.	365
31DEC1959	DATE9.	-1

53

The ANYDTDTE*w.* informat can be used to read data that has a variety of date forms.

Informats always contain a *w* value to indicate the width of the raw data field. A perod (.) ends the informat or separates the *w* value from the optional *d* value, which specifies the number of decimal places.

Acknowledgments: Portions of SAS' copyrighted SAS course content are reproduced here with permission of SAS Institute Inc., Cary, NC, USA. SAS Institute Inc. makes no warranties with respect to these materials and disclaims all liability therefor. Neither the GW nor the instructor shall be held liable or responsible to any person or entity with respect to any loss or incidental or consequential damages caused by anyone's use of the information or SAS codes contained herein. Circulating the handouts is strictly forbidden.

Column Input: Reading Data in Fixed Columns with Some Shorter Records from an external file

```
001 Tim Dyson    74 8724
002 Sam Larson   96 8224
003 Jane Miller  91
004 Bikas Das    90 8783
```

The third record above is shorter than expected.

```
Data HAVE;
   infile 'C:\SASCourse\Week2\short_records.txt'   lrecl=25 PAD;
   input id 1-3 name $ 5-16  score 18-19 @21 some_value 5.2;
 run;
```

With the PAD option on the INFILE statement, SAS pads the records in fixed columns read from an external file with blanks to the length specified in the LRECL= option or implied by the column position.

Column Input: Reading Data in Fixed Columns with Some Values in Fewer than the Number of Columns Specified from an external file

```
001 Tim Dyson    74 8724
002 Sam Larson   96  224
003 Jane Miller  91   24
004 Bikas Das    90    4
```

The last field holds values in fewer than four columns among the last three records above.

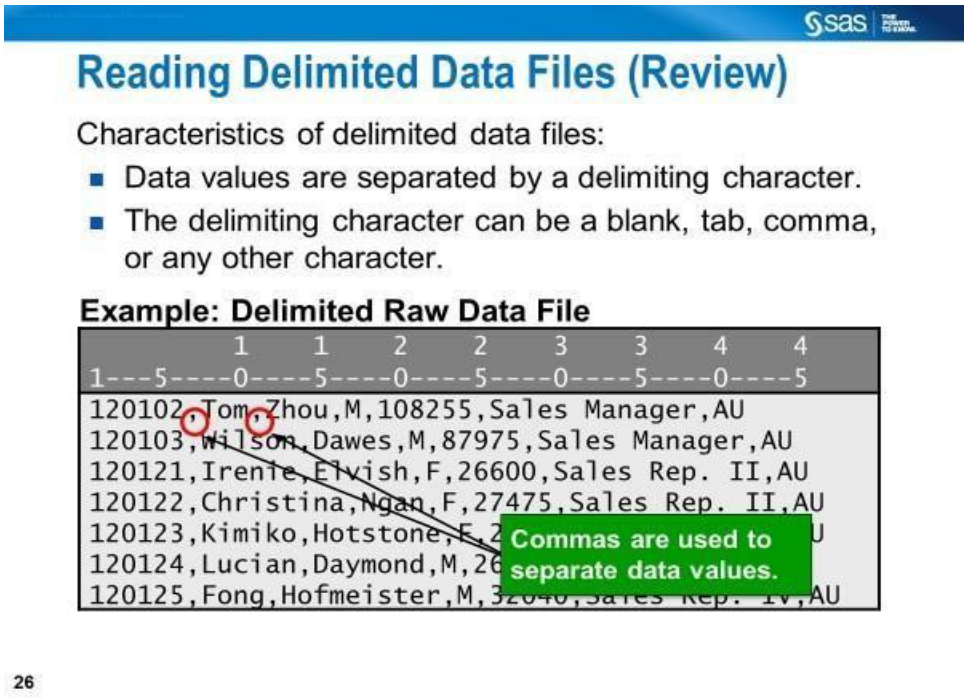
```
data HAVE;
  infile 'C:\SASCourse\Week2\short_values.txt' TRUNCOVER;
  input id 1-3 name $ 5-16 score 18-19 @21 some_value 5.2;
run;
```

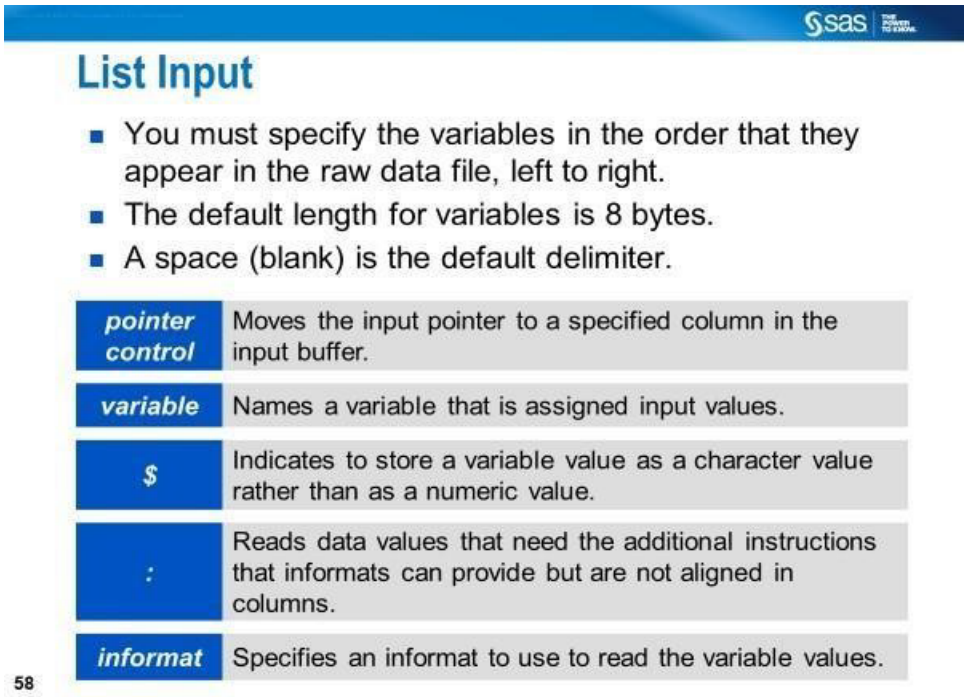
The TRUNCOVER option on the INFILE statement correctly assigns the contents of the input buffer to the last field.

Acknowledgments: Portions of SAS' copyrighted SAS course content are reproduced here with permission of SAS Institute Inc., Cary, NC, USA. SAS Institute Inc. makes no warranties with respect to these materials and disclaims all liability therefor. Neither the GW nor the instructor shall be held liable or responsible to any person or entity with respect to any loss or incidental or consequential damages caused by anyone's use of the information or SAS codes contained herein. Circulating the handouts is strictly forbidden.

When to Use List Input

-





Acknowledgments: Portions of SAS' copyrighted SAS course content are reproduced here with permission of SAS Institute Inc., Cary, NC, USA. SAS Institute Inc. makes no warranties with respect to these materials and disclaims all liability therefor. Neither the GW nor the instructor shall be held liable or responsible to any person or entity with respect to any loss or incidental or consequential damages caused by anyone's use of the information or SAS codes contained herein. Circulating the handouts is strictly forbidden.

Reading a Delimited Raw Data File

Use *INFILE* and *INPUT statements* in a DATA step to read a raw data file.

```
data work.subset;
   infile "&path\sales.csv" dlm=',';
   input Employee_ID First_Name $
         Last_Name $ Gender $ Salary
         Job_Title $ Country $;
run;
```

DATA *output-data-set*;
INFILE "*raw-data-file*" <**DLM=**'*delimiter*'>;
INPUT *variable* <$> *variable* <$> … ;
RUN;

11 p108d01

INPUT Statement

The INPUT statement reads the data fields sequentially, left to right. Standard data fields require only a variable name and type.

Partial **sales.csv**

```
120102,Tom,Zhou,M,108255,Sales Manager,AU,11AUG1969,06/01/1989
120103,Wilson,Dawes,M,87975,Sales Manager,AU,22JAN1949,01/01/1974
120121,Irenie,Elvish,F,26600,Sales Rep. II,AU,02AUG1944,01/01/1974
```

```
input Employee_ID First_Name $ Last_Name $
      Gender $ Salary Job_Title $ Country $;
```

INPUT *variable* <$> *variable* <$> …;

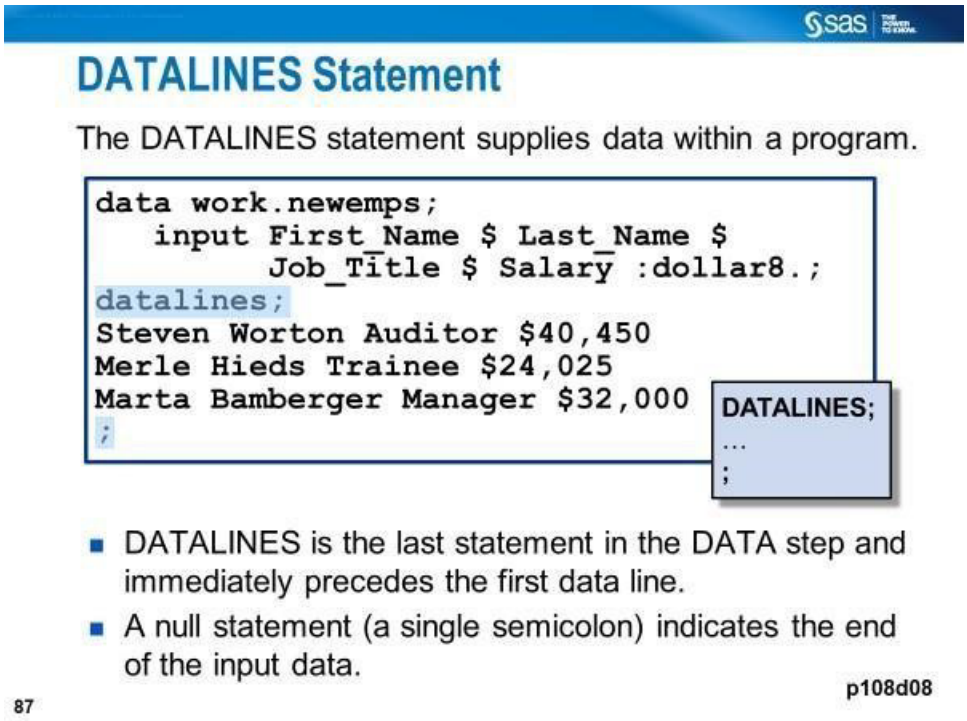
- The optional dollar sign indicates a character variable.
- Default length for ***all*** variables is eight bytes, regardless of type.

13

Acknowledgments: Portions of SAS' copyrighted SAS course content are reproduced here with permission of SAS Institute Inc., Cary, NC, USA. SAS Institute Inc. makes no warranties with respect to these materials and disclaims all liability therefor. Neither the GW nor the instructor shall be held liable or responsible to any person or entity with respect to any loss or incidental or consequential damages caused by anyone's use of the information or SAS codes contained herein. Circulating the handouts is strictly forbidden.

Modified List Input: Using the Colon Modifier in the Input Statement

- Free-Format Data
- Data fields separated by 1+ blanks or other delimiters
- Data fields have standard numeric values or characters more than 8 bytes
- Data fields have non-standard numeric values



The colon (:) format modifier enables you to use list input but also to specify an informat after a variable name, whether character or numeric. SAS reads until it encounters a blank column, the defined length of the variable (character only), or the end of the data line, whichever comes first.

Here, the INFORMAT specifies the length for each character variable, not the number of columns that are read.

Acknowledgments: Portions of SAS' copyrighted SAS course content are reproduced here with permission of SAS Institute Inc., Cary, NC, USA. SAS Institute Inc. makes no warranties with respect to these materials and disclaims all liability therefor. Neither the GW nor the instructor shall be held liable or responsible to any person or entity with respect to any loss or incidental or consequential damages caused by anyone's use of the information or SAS codes contained herein. Circulating the handouts is strictly forbidden.

The LENGTH Statement

A LENGTH statement contains the names of the variables followed by the number of bytes to be used for storage.

```
LENGTH  RECTYPE  3 HHX  $ 6 INTV_QRT 3 INTV_MON 3;
```

The length of a variable is the number of bytes SAS allocates for storing the variable. It is not necessarily the same as the number of characters in the variable. The LENGTH statement is used in the data step to reduce the size (in terms of disk space) of SAS data sets. By default, SAS uses 8 bytes to store numeric variables. Variables containing integer values can be stored using less than 8 bytes.

Largest integer represented exactly by length for SAS variables under Windows	
Length in bytes	Largest integer represented exactly
3	8,192
4	2,097,152
5	536,870,912
6	137,438,953,472
7	35,184,372,088,832
8	9,007,199,254,740,990

Data sets containing many integer variables (such are common with data collected by questionnaire) or indicator variables can be reduced in size by more than 50%.

Variables containing real numbers should be left with the default length of 8. Note that specifying a length less than that required will result in a loss of precision without any warning being given.

Acknowledgments: Portions of SAS' copyrighted SAS course content are reproduced here with permission of SAS Institute Inc., Cary, NC, USA. SAS Institute Inc. makes no warranties with respect to these materials and disclaims all liability therefor. Neither the GW nor the instructor shall be held liable or responsible to any person or entity with respect to any loss or incidental or consequential damages caused by anyone's use of the information or SAS codes contained herein. Circulating the handouts is strictly forbidden.

LENGTH Statement to Define the Length of Both Numeric and Character Variables

Using a LENGTH Statement

The LENGTH statement identifies the character variables, so dollar signs can be omitted from the INPUT statement.

```
data work.subset;
   length Employee_ID 8 First_Name $ 12
          Last_Name $ 18 Gender $ 1
          Salary 8 Job_Title $ 25
          Country $ 2;
   infile "&path\sales.csv" dlm=',';
   input Employee_ID First_Name Last_Name
         Gender Salary Job_Title Country;
run;
```

p108d03

49

Viewing the Output

Display the variables in creation order.

```
proc contents data=work.subset varnum;
run;
```

Partial PROC CONTENTS Output

Variables in Creation Order

#	Variable	Type	Len
1	Employee_ID	Num	8
2	First_Name	Char	12
3	Last_Name	Char	18
4	Gender	Char	1
5	Salary	Num	8
6	Job_Title	Char	25
7	Country	Char	2

50

Acknowledgments: Portions of SAS' copyrighted SAS course content are reproduced here with permission of SAS Institute Inc., Cary, NC, USA. SAS Institute Inc. makes no warranties with respect to these materials and disclaims all liability therefor. Neither the GW nor the instructor shall be held liable or responsible to any person or entity with respect to any loss or incidental or consequential damages caused by anyone's use of the information or SAS codes contained herein. Circulating the handouts is strictly forbidden.

Reading Non-Standard Data: Modifier with an Informat instead of the Informat Statement

Considerations

Use modified list input to read all the fields from **sales.csv**. Store the date fields as SAS dates.

Partial **sales.csv**

```
120102,Tom,Zhou,M,108255,Sales Manager,AU,11AUG1973,06/01/1993
120103,Wilson,Dawes,M,87975,Sales Manager,AU,22JAN1953,01/01/1978
120121,Irenie,Elvish,F,26600,Sales Rep. II,AU,02AUG1948,01/01/1978
120122,Christina,Ngan,F,27475,Sales Rep. II,AU,27JUL1958,07/01/1982
120123,Kimiko,Hotstone,F,26190,Sales Rep. I,AU,28SEP1968,10/01/1989
```

63

Modified List Input

This DATA step uses *modified list input.* Instead of a LENGTH statement, an informat specifies the length for each character variable.

input *variable* <:*informat.*> …;

```
data work.subset;
   infile "&path\sales.csv" dlm=',';
   input Employee_ID First_Name :$12.
         Last_Name :$18. Gender :$1. Salary
         Job_Title :$25. Country :$2.;
run;
```

- The **$12.** informat defines a length of 12 for **First_Name** and allows up to 12 characters to be read.
- The **:** format modifier tells SAS to read until it encounters a delimiter.

64

p108d05

Acknowledgments: Portions of SAS' copyrighted SAS course content are reproduced here with permission of SAS Institute Inc., Cary, NC, USA. SAS Institute Inc. makes no warranties with respect to these materials and disclaims all liability therefor. Neither the GW nor the instructor shall be held liable or responsible to any person or entity with respect to any loss or incidental or consequential damages caused by anyone's use of the information or SAS codes contained herein. Circulating the handouts is strictly forbidden.

INFILE Options

INFILE "*raw-data-file***" <DLM=> <DSD> <MISSOVER>;**

Option	Description
DLM=	Specifies an alternate delimiter.
DSD	Sets the default delimiter to a comma, treats consecutive delimiters as missing values, and allows embedded delimiters when the data value is enclosed in quotation marks.
MISSOVER	Sets variables to missing if the end of the record is reached before finding values for all fields.

100

Acknowledgments: Portions of SAS' copyrighted SAS course content are reproduced here with permission of SAS Institute Inc., Cary, NC, USA. SAS Institute Inc. makes no warranties with respect to these materials and disclaims all liability therefor. Neither the GW nor the instructor shall be held liable or responsible to any person or entity with respect to any loss or incidental or consequential damages caused by anyone's use of the information or SAS codes contained herein. Circulating the handouts is strictly forbidden.

DLM= Option

The DLM= option specifies a delimiter to be used for list input. Blank is the default delimiter.

```
Chloe,2,11/10/1995,$5,Running,Music,Gymnastics
Travis,2,1/30/1998,$2,Baseball,Nintendo,Reading
Jennifer,0,8/21/1999,$0,Soccer,Painting,Dancing
```

Input Raw Data File

```
data work.kids4;
   infile 'kids4.dat' dlm=',';
   input name $
         siblings
         bdate : mmddyy10.
         allowance : comma2.
         hobby1 : $10.
         hobby2 : $10.
         hobby3 : $10.;
run;
```

69

Modified List Input and DSD Option on the Infile Statement

DSD Option

The DSD option can do the following:

- treat two consecutive delimiters as a missing value
- remove quotation marks from strings and treat any delimiter inside the quotation marks as a valid character
- set the default delimiter to a comma

```
data work.kids5;
   infile 'kids5.dat' dsd;
   input name $
         siblings
         bdate : mmddyy10.
         allowance : comma2.
         hobby1 : $10.
         hobby2 : $10.
         hobby3 : $10.;
run;
```

72

Acknowledgments: Portions of SAS' copyrighted SAS course content are reproduced here with permission of SAS Institute Inc., Cary, NC, USA. SAS Institute Inc. makes no warranties with respect to these materials and disclaims all liability therefor. Neither the GW nor the instructor shall be held liable or responsible to any person or entity with respect to any loss or incidental or consequential damages caused by anyone's use of the information or SAS codes contained herein. Circulating the handouts is strictly forbidden.

Missing Values at the End of a Record

The raw data file **phone.csv** contains missing values at the end of some records.

phone.csv

```
          1    1    2          missing values   4    4
1---5----0----5----0----5----0----5----0----5----0----5
James Kvarniq,(704) 293-8126,(701) 281-8923
Sandrina Stephano,(919) 871-7830
Cornelia Krahl,(212) 891-3241,(212) 233-5413
Karen Ballinger,(714) 344-4321
Elke Wallstab,(910) 763-5561,(910) 545-3421
```

The DSD option is not appropriate because the missing data is not marked by consecutive delimiters.

97

MISSOVER Option

The *MISSOVER option* prevents SAS from loading a new record when the end of the current record is reached.

```
data contacts;
   length Name $ 20 Phone Mobile $ 14;
   infile "&path\phone.csv" dlm=',' missover;
   input Name $ Phone $ Mobile $;
run;

proc print data=contacts noobs;
run;
```

INFILE "*raw-data-file*" <DLM=> MISSOVER;

If SAS reaches the end of a record without finding values for all fields, variables without values are set to missing.

p108d10

98

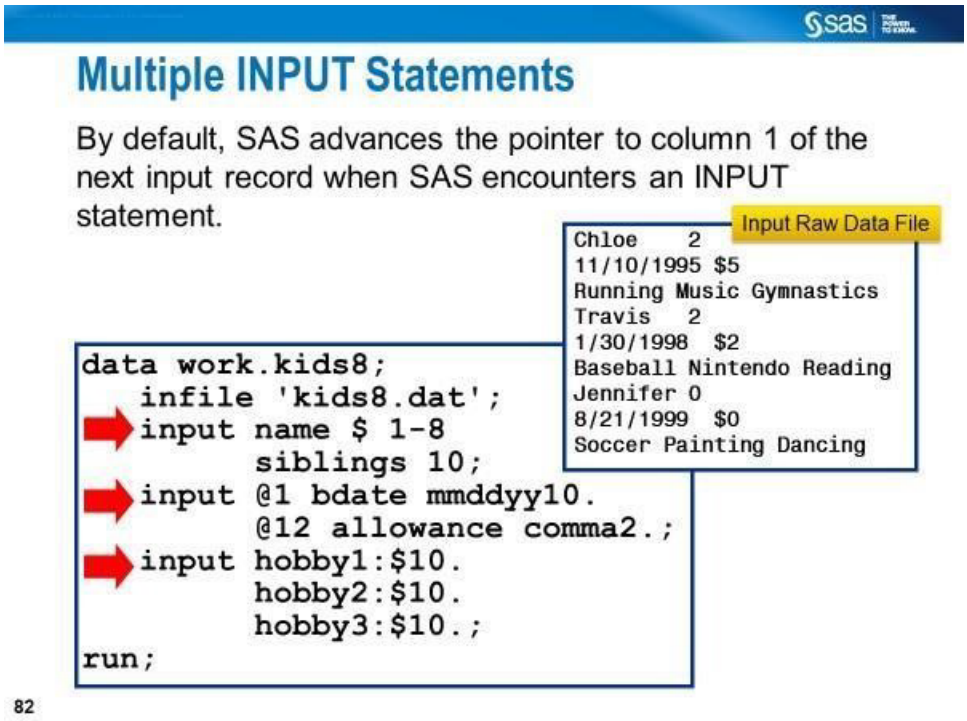
Use MISSOVER if the last field or fields might be missing and you want SAS to assign missing values to the corresponding variable.

Acknowledgments: Portions of SAS' copyrighted SAS course content are reproduced here with permission of SAS Institute Inc., Cary, NC, USA. SAS Institute Inc. makes no warranties with respect to these materials and disclaims all liability therefor. Neither the GW nor the instructor shall be held liable or responsible to any person or entity with respect to any loss or incidental or consequential damages caused by anyone's use of the information or SAS codes contained herein. Circulating the handouts is strictly forbidden.

The Ampersand (&) Modifier

The ampersand (&) format modifier enables you to read character values that contain embedded blanks with list input and specify a character informat. SAS reads until it encounters **two consecutive blanks**, the defined variable length, or the end of the input line, whichever comes first.

```
*Ex6_Formated_Input_Dates.sas (Part 7);
options yearcutoff=1720; data yc;
   INPUT state_name  & $22. date_entry :mmddyy.;
   FORMAT date_entry :mmddyy10.;
DATALINES;
Delaware  12/07/87
Pennsylvania  12/12/87
New Jersey  12/18/87
South Carolina  05/23/88
;
proc print data=yc noobs;
run;
```



Acknowledgments: Portions of SAS' copyrighted SAS course content are reproduced here with permission of SAS Institute Inc., Cary, NC, USA. SAS Institute Inc. makes no warranties with respect to these materials and disclaims all liability therefor. Neither the GW nor the instructor shall be held liable or responsible to any person or entity with respect to any loss or incidental or consequential damages caused by anyone's use of the information or SAS codes contained herein. Circulating the handouts is strictly forbidden.

Line-Pointer Controls

The / line-pointer control advances the pointer to column 1 of the next input record.

```
data work.kids8;
   infile 'kids8.dat';
   input name $ 1-8
         siblings 10
       / @1 bdate mmddyy10.
         @12 allowance comma2.
       / hobby1:$10.
         hobby2:$10.
         hobby3:$10.;
run;
```

86

Line-Pointer Controls

The #*n* line-pointer control advances the pointer to column 1 of record *n*.

```
data work.kids8;
   infile 'kids8.dat';
   input #1 name $ 1-8
            siblings 10
         #2 @1 bdate mmddyy10.
            @12 allowance comma2.
         #3 hobby1:$10.
            hobby2:$10.
            hobby3:$10.;
run;
```

87

Acknowledgments: Portions of SAS' copyrighted SAS course content are reproduced here with permission of SAS Institute Inc., Cary, NC, USA. SAS Institute Inc. makes no warranties with respect to these materials and disclaims all liability therefor. Neither the GW nor the instructor shall be held liable or responsible to any person or entity with respect to any loss or incidental or consequential damages caused by anyone's use of the information or SAS codes contained herein. Circulating the handouts is strictly forbidden.

Line-Pointer Controls

The second record is skipped in each iteration of the DATA step.

```
input name $ 1-8 siblings 10 //
      hobby1:$10. hobby2:$10. hobby3:$10.;
```

```
input #1 name $ 1-8 siblings 10
      #3 hobby1:$10. hobby2:$10. hobby3:$10.;
```

```
input name $ 1-8 siblings 10;
input;
input hobby1:$10. hobby2:$10. hobby3:$10.;
```

Output Data Set

VIEWTABLE: Work.Kids8

	name	siblings	hobby1	hobby2	hobby3
1	Chloe	2	Running	Music	Gymnastics
2	Travis	2	Baseball	Nintendo	Reading
3	Jennifer	0	Soccer	Painting	Dancing

90

When to Use the Forward Slash (/) Line Pointer Control

You have data elements that need to be read sequentially from multiple records to create a single observation.

Acknowledgments: Portions of SAS' copyrighted SAS course content are reproduced here with permission of SAS Institute Inc., Cary, NC, USA. SAS Institute Inc. makes no warranties with respect to these materials and disclaims all liability therefor. Neither the GW nor the instructor shall be held liable or responsible to any person or entity with respect to any loss or incidental or consequential damages caused by anyone's use of the information or SAS codes contained herein. Circulating the handouts is strictly forbidden.

The Use of Multiple Input Statements

```
*Ex29_Multiple_Input_Statements.sas;
data work.HAVE(drop=i);
 input date: Anydtdte9. @;
 do i = 1 to 4;
 input name $ study_hours @;
 output;
 end;
datalines;
27Aug2018 Doris 5.5 Alice 4.0 Mike 2.0 James 1.0
28Jun2018 Doris 3.0 Alice 3.0 Mike 3.0 James 1.0
;
proc print data=work.HAVE;
Format date worddate.;
run;
```

Obs	date	name	study_ hours
1	August 27, 2018	Doris	5.5
2	August 27, 2018	Alice	4.0
3	August 27, 2018	Mike	2.0
4	August 27, 2018	James	1.0
5	June 28, 2018	Doris	3.0
6	June 28, 2018	Alice	3.0
7	June 28, 2018	Mike	3.0
8	June 28, 2018	James	1.0

Normally, each INPUT statement in a DATA step reads a new data record into the input buffer. However, when you use a trailing @, the following things occur:

- The pointer position does not change.
- No new record is read into the input buffer.
- The next INPUT statement for the same iteration of the DATA step continues to read the same record rather than a new one.
  SAS releases a record held by a trailing @ when a null INPUT statement executes (**input;**).
- An INPUT statement without a trailing @ executes the next iteration of the DATA step.

List Input with a Double Trailing @ at the End of the Input Statement

Acknowledgments: Portions of SAS' copyrighted SAS course content are reproduced here with permission of SAS Institute Inc., Cary, NC, USA. SAS Institute Inc. makes no warranties with respect to these materials and disclaims all liability therefor. Neither the GW nor the instructor shall be held liable or responsible to any person or entity with respect to any loss or incidental or consequential damages caused by anyone's use of the information or SAS codes contained herein. Circulating the handouts is strictly forbidden.

Double Trailing @

Input Raw Data File

```
Chloe IN Travis IL Jennifer IN
Brian IL Mark IN Kurt IN Hannah IL
```

```
data work.kids10;
   infile 'kids10.dat';
   input name $ state $ @@;
run;
```

Output Data Set

VIEWTABLE: Work.Kids10

	name	state
1	Chloe	IN
2	Travis	IL
3	Jennifer	IN
4	Brian	IL
5	Mark	IN
6	Kurt	IN
7	Hannah	IL

```
NOTE: 2 records were read from the infile 'kids10.dat'.
      The minimum record length was 30.
      The maximum record length was 34.
NOTE: SAS went to a new line when INPUT statement reached past the
      end of a line.
NOTE: The data set WORK.KIDS10 has 7 observations and 2 variables.
```

96

Acknowledgments: Portions of SAS' copyrighted SAS course content are reproduced here with permission of SAS Institute Inc., Cary, NC, USA. SAS Institute Inc. makes no warranties with respect to these materials and disclaims all liability therefor. Neither the GW nor the instructor shall be held liable or responsible to any person or entity with respect to any loss or incidental or consequential damages caused by anyone's use of the information or SAS codes contained herein. Circulating the handouts is strictly forbidden.

Modified List Input with a double @ at the End of the Input Statement

```
1  *Ex30_@@.sas;
2  data work.HAVE;
3   input date: Anydtdte. name $ study_hours @@;
4  datalines;
5  27Aug2018 Doris 5.5 28Aug2018 Alice 4.0
6  29Aug2018 Mike 2.0 29Aug2018 James 1.0
7  30Jun2018 Doris 3.0 31Aug2018 Alice 3.0
8  01Sep2018 Mike 3.0
9  02Sep2018 James 1.0
10 ;
11 proc print data=work.HAVE;
12 Format date mmddyy10.;
13 run;
```

In the above program, we are reading 5 lines of data with the **Datalines** statement.
Lines 5, 6, and 7 each have two records, and lines 8 and 9 each have one record. Here, our goal is to create a SAS data file that will have one observation from each record – a total of 8 observations.

Notice that the Informat for the variable **date** does not specify a *w* value because we are reading data using List Input with a colon modifier; here, SAS reads data fields until it encounters a blank.

Acknowledgments: Portions of SAS' copyrighted SAS course content are reproduced here with permission of SAS Institute Inc., Cary, NC, USA. SAS Institute Inc. makes no warranties with respect to these materials and disclaims all liability therefor. Neither the GW nor the instructor shall be held liable or responsible to any person or entity with respect to any loss or incidental or consequential damages caused by anyone's use of the information or SAS codes contained herein. Circulating the handouts is strictly forbidden.

When to Use Named Input

- Data records contain both variable names and values

What to Specify on the Input Statement (Named Input)

- Each variable name must be followed by an equal sign on the INPUT statement
- Each variable name must be followed by an equal sign on the data as well
- Character variables must be indicated by a "$" following the equals sign on the INPUT statement
- Appropriate format modifiers and informats must be specified
- The"/" at the end of the line must be used to read the next data line to complete the observation

```
*Ex16_Named_Input.sas;
options nocenter nodate nonumber ls=132;
DATA TEST;
input name = & $ 30. address = & $ 30.
      city_zip  = & $ 30. phone= $ 14.
      Num_employees = ;
      FORMAT Num_employees comma7.;
DATALINES;
name=Air Force Personnel Center /
address=550 C Street West /
city_zip=Randolph AFB, TX 78150 /
phone=1-800-525-0102 /
Num_employees=5876
name= Navy Personnel Command /
address= 5720 Integrity Drive /
city_zip= Millington, TN 38055 /
phone= 901-874-4885 /
Num_employees=3987
;
proc print noobs; run;
```

name	address	city_zip	phone	Num_ employees
Air Force Personnel Center	550 C Street West	Randolph AFB, TX 78150	1-800-525-0102	5,876
Navy Personnel Command	5720 Integrity Drive	Millington, TN 38055	901-874-4885	3,987

Acknowledgments: Portions of SAS' copyrighted SAS course content are reproduced here with permission of SAS Institute Inc., Cary, NC, USA. SAS Institute Inc. makes no warranties with respect to these materials and disclaims all liability therefor. Neither the GW nor the instructor shall be held liable or responsible to any person or entity with respect to any loss or incidental or consequential damages caused by anyone's use of the information or SAS codes contained herein. Circulating the handouts is strictly forbidden.

Processing a DATA Step: A Walk-through

Exploring SAS Data Step Processing

- After the SAS program is submitted, the codes are copied into a memory area called the **input stack**
- The **word scanner** reads the text from the input stack and breaks it into fundamental units called tokens, which are of four types:
    - **Literal** - a string of characters enclosed in quotation marks (e.g., "GWU")
    - **Number** - digits, date values, time values, and hexadecimal numbers (e.g., 1234 '13mar2016'dv 20e4v 14.5)
    - **Name** - a string of characters beginning with an underscore or letter (_N_ , Descending, means)
    - **Special characters** – other than a letter, number, or underscore that have a special meaning to the SAS system (e.g., * / + - % &)
- The word scanner **passes tokens**, one at a time, to the appropriate compiler (regular SAS compiler, or macro compiler) as appropriate. The compiler **requests tokens until it receives a semicolon.** The word scanner repeats this process for each SAS statement.
- The compilation is suspended after the step boundary is encountered.
- SAS statements are **scanned for syntax errors** ○ missing semicolons ○ misspelled keywords ○ unmatched quotation marks ○ invalid options
- If there is **no compilation error** for the step, SAS **executes the compiled code**.

Compilation Phase

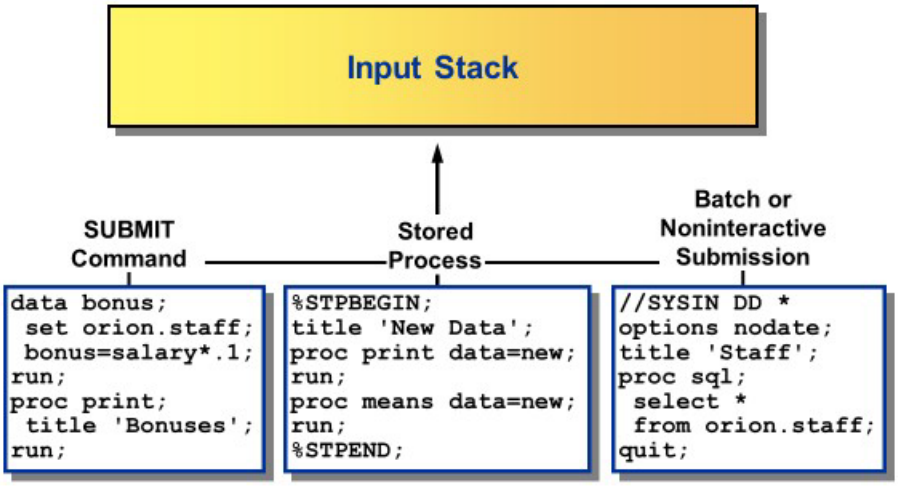
- The **input buffer** is created to hold a data record from the raw data file
- The **program data vector** (PDV) is created for two automatic variables (_N_ and _ERROR_)
- The **descriptor portion** of the SAS data file is created: data file name, # of observations, and the number, names, and attributes of variables

Execution Phase • **Variables in the PDV are initialized to missing** before each execution of the data step

- The DATA step **executes once for each record in the input file**, unless otherwise directed by additional statements
- Each record in the raw data file is **read into the input buffer**, **copied to the PDV**, and then **written to the SAS data file** at the end of the DATA step

Acknowledgments: Portions of SAS' copyrighted SAS course content are reproduced here with permission of SAS Institute Inc., Cary, NC, USA. SAS Institute Inc. makes no warranties with respect to these materials and disclaims all liability therefor. Neither the GW nor the instructor shall be held liable or responsible to any person or entity with respect to any loss or incidental or consequential damages caused by anyone's use of the information or SAS codes contained herein. Circulating the handouts is strictly forbidden.

Program Flow

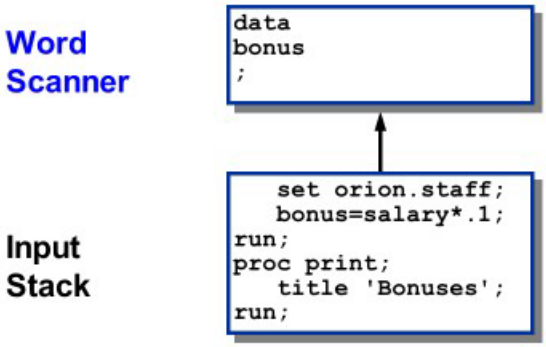


39

Program Flow

When SAS code is in the input stack, a component of SAS called the *word scanner* does the following:

- reads the text in the input stack, character by character, left to right, top to bottom
- breaks the text into fundamental units called *tokens*



40

Acknowledgments: Portions of SAS' copyrighted SAS course content are reproduced here with permission of SAS Institute Inc., Cary, NC, USA. SAS Institute Inc. makes no warranties with respect to these materials and disclaims all liability therefor. Neither the GW nor the instructor shall be held liable or responsible to any person or entity with respect to any loss or incidental or consequential damages caused by anyone's use of the information or SAS codes contained herein. Circulating the handouts is strictly forbidden.

Program Flow

The word scanner recognizes four classes of tokens:

- name tokens

Word Scanner

```
data
bonus
;
```

Input Stack

```
   set orion.staff;
   bonus=salary*.1;
run;
proc print;
   title 'Bonuses';
run;
```

42

...

Program Flow

The word scanner recognizes four classes of tokens:

- name tokens
- special tokens

Word Scanner

```
data
bonus
;
```

Input Stack

```
   set orion.staff;
   bonus=salary*.1;
run;
proc print;
   title 'Bonuses';
run;
```

43

...

Acknowledgments: Portions of SAS' copyrighted SAS course content are reproduced here with permission of SAS Institute Inc., Cary, NC, USA. SAS Institute Inc. makes no warranties with respect to these materials and disclaims all liability therefor. Neither the GW nor the instructor shall be held liable or responsible to any person or entity with respect to any loss or incidental or consequential damages caused by anyone's use of the information or SAS codes contained herein. Circulating the handouts is strictly forbidden.

Program Flow

The word scanner recognizes four classes of tokens:

- name tokens
- special tokens
- literal tokens

Word Scanner

```
data
bonus
;
```

Input Stack

```
   set orion.staff;
   bonus=salary*.1;
run;
proc print;
   title 'Bonuses';
run;
```

44

...

Program Flow

The word scanner recognizes four classes of tokens:

- name tokens
- special tokens
- literal tokens
- number tokens

Word Scanner

```
data
bonus
;
```

Input Stack

```
   set orion.staff;
   bonus=salary*.1;
run;
proc print;
   title 'Bonuses';
run;
```

45

Acknowledgments: Portions of SAS' copyrighted SAS course content are reproduced here with permission of SAS Institute Inc., Cary, NC, USA. SAS Institute Inc. makes no warranties with respect to these materials and disclaims all liability therefor. Neither the GW nor the instructor shall be held liable or responsible to any person or entity with respect to any loss or incidental or consequential damages caused by anyone's use of the information or SAS codes contained herein. Circulating the handouts is strictly forbidden.

Tokenization

A token ends when the word scanner detects

- a blank
- the beginning of another token.

Word Scanner

```
data
bonus
;
```

Input Stack

```
   set orion.staff;
   bonus=salary*.1;
run;
proc print;
   title 'Bonuses';
run;
```

The maximum length of a token is 32,767 characters.

50

Program Flow

The word scanner passes tokens, one at a time, to the appropriate compiler, as the compiler demands.

Compiler

```
data bonus;
```

Word Scanner

```
set
orion
.
staff
;
```

Input Stack

```
   bonus=salary*.1;
run;
proc print;
   title 'Bonuses';
run;
```

52

Acknowledgments: Portions of SAS' copyrighted SAS course content are reproduced here with permission of SAS Institute Inc., Cary, NC, USA. SAS Institute Inc. makes no warranties with respect to these materials and disclaims all liability therefor. Neither the GW nor the instructor shall be held liable or responsible to any person or entity with respect to any loss or incidental or consequential damages caused by anyone's use of the information or SAS codes contained herein. Circulating the handouts is strictly forbidden.

How to represent missing values

When you have a space-delimited file, and you have a missing value, you have to represent the missing value as a period when reading data using list or modified list input.

When you have a comma-delimited file or any type of delimited file, you do not have to have a period as a missing value because you normally have multiple delimiters in a row to indicate a missing value.

When you have a space-delimited file that is column-aligned, then you do not have to have a period to represent a missing value because you are using column input to read the data.

Handling Missing Values

Missing Data Values

Missing values are valid values in a SAS data set.

Partial **work.newsalesemps**

First_Name	Last_Name	Job_Title	Salary
Monica	Kletschkus	Sales Rep. IV	.
Kevin	Lyon	Sales Rep. I	26955
Petrea	Soltau		27440

A blank represents a missing character value.

A period represents a missing numeric value.

- A value must exist for every variable in every observation.

10

In addition to the above two ordinary missing values, 26 special missing values only apply to the numeric variable.

The MISSING= system option allows specifying a character to be printed when numeric variables contain ordinary missing values (.).

If your data contains characters that represent special missing values, such as a or z, do not use the MISSING= system option to define them; simply define these values in a MISSING statement.

Acknowledgments: Portions of SAS' copyrighted SAS course content are reproduced here with permission of SAS Institute Inc., Cary, NC, USA. SAS Institute Inc. makes no warranties with respect to these materials and disclaims all liability therefor. Neither the GW nor the instructor shall be held liable or responsible to any person or entity with respect to any loss or incidental or consequential damages caused by anyone's use of the information or SAS codes contained herein. Circulating the handouts is strictly forbidden.

Reading Microsoft Excel Data into SAS Data Sets

Creating Raw Tabular Data Files or Reports Using DATA _NULL_ PUT and PUTLOG Statements

In general, the PUT statement writes to the LOG or to an External File with a FILE statement, but the PUTLOG statement always writes to the LOG.

```
1 *Ex21_put.sas;
2 * List all DATA step variables and their values;
3 data _null_;
4   set sashelp.class(obs=2);
5   put _all_;
6 run;
```

Line 3: The keyword _NULL_ on the DATA statement is used to execute the data step without creating a data set.

Line 5: The PUT statement is used to create output records to the LOG window. The special SAS name list _ALL_ refers to all variables on the data step and program data vector, including _N_ and _ERROR_.

```
Name=Alfred Sex=M Age=14 Height=69 Weight=112.5 _ERROR_=0 _N_=1
Name=Alice Sex=F Age=13 Height=56.5 Weight=84 _ERROR_=0 _N_=2
NOTE: There were 2 observations read from the data set SASHELP.CLASS.
```

```
 7 * Exclude automatic variables;
 8 data _null_;
 9   set sashelp.class(obs=2);
10   put (_all_)(=);
11 run;
```

Acknowledgments: Portions of SAS' copyrighted SAS course content are reproduced here with permission of SAS Institute Inc., Cary, NC, USA. SAS Institute Inc. makes no warranties with respect to these materials and disclaims all liability therefor. Neither the GW nor the instructor shall be held liable or responsible to any person or entity with respect to any loss or incidental or consequential damages caused by anyone's use of the information or SAS codes contained herein. Circulating the handouts is strictly forbidden.

Line 10: In the PUT statement, the variable list argument is _ALL_ [i.e., all variables on the data step and program data vector excluding _N_ and _ERROR_]; the format argument is the equal sign so that the variable name precedes its value.

```
Name=Alfred Sex=M Age=14 Height=69 Weight=112.5
Name=Alice Sex=F Age=13 Height=56.5 Weight=84
NOTE: There were 2 observations read from the data set SASHELP.CLASS.
17 /* Put each value on a new line and apply
18 a common format to all numeric variables*/
19 data _null_;
20   set sashelp.class(obs=2);
21   put (_all_)(=/12.2);
22 run;
```

Line 21: The PUT statement is used to tell SAS to write each of the variables in the data step and program data vector (except _N_ and _ERROR_) that must precede its value. An additional format argument / is used so that each variable and its value, separated by an equal sign, is output to a separate line. Note the **FORMAT***w.d* (i.e., 12.2) that applies to all numeric variables.

```
Name=Alfred
Sex=M
Age=14.00
Height=69.00
Weight=112.50

Name=Alice
Sex=F
Age=13.00
Height=56.50
Weight=84.00
NOTE: There were 2 observations read from the data set SASHELP.CLASS.
```

Acknowledgments: Portions of SAS' copyrighted SAS course content are reproduced here with permission of SAS Institute Inc., Cary, NC, USA. SAS Institute Inc. makes no warranties with respect to these materials and disclaims all liability therefor. Neither the GW nor the instructor shall be held liable or responsible to any person or entity with respect to any loss or incidental or consequential damages caused by anyone's use of the information or SAS codes contained herein. Circulating the handouts is strictly forbidden.

```
23 /* List values as a table and apply formats
24 to groups of variables*/
25 data _null_;
26   set sashelp.class(obs=2);
27   if _n_=1 then put @1 'NAME' @19 'SEX' @23 'AGE'
28                     @30 'HEIGHT' @38 'WEIGHT';
29   put (_all_)(1*$20.,1*$2.,1*3.,2*8.2);
30 run;
```

Line 27: During the first iteration, the variable names are printed, each starting with the column position specified, to the LOG window by default.

Line 29: This is an example of the FORMATTED PUT statement. There is no equal sign or slash as a format-list argument. Formats specified as format-list arguments. For example, a character format **$20.** is applied to the variable NAME. Another character format **$2.** is applied to the variable SEX. A numeric format **3.** is applied to the variable AGE. Another numeric format, 8.2, is applied to the variables HEIGHT and WEIGHT.

```
NAME                 SEX AGE    HEIGHT  WEIGHT
Alfred                 M  14   69.00  112.50
Alice                  F  13   56.50   84.00
NOTE: There were 2 observations read from the data set SASHELP.CLASS.
```

Acknowledgments: Portions of SAS' copyrighted SAS course content are reproduced here with permission of SAS Institute Inc., Cary, NC, USA. SAS Institute Inc. makes no warranties with respect to these materials and disclaims all liability therefor. Neither the GW nor the instructor shall be held liable or responsible to any person or entity with respect to any loss or incidental or consequential damages caused by anyone's use of the information or SAS codes contained herein. Circulating the handouts is strictly forbidden.

```
33 /* List values as a table and apply formats
34 to groups of variables. Route output to the
35 standard SAS output window. */
36 options nodate nonumber;
37 title;
38 data _null_;
39   set sashelp.class(obs=2);
40   file print;
41   if _n_=1 then put @1 'NAME' @19 'SEX' @23 'AGE'
42                     @30 'HEIGHT' @38 'WEIGHT';
43   put (_all_)(1*$20.,1*$2.,1*3.,2*8.2);
44 run;
```

Line 40: This statement creates the tabular output to the OUTPUT window, not the LOG window.

(Partial SAS Log)

```
NOTE: 3 lines were written to file PRINT.
```

Snapshot from the SAS output window

```
NAME                 SEX AGE    HEIGHT  WEIGHT
Alfred                 M  14    69.00  112.50
Alice                  F  13    56.50   84.00
```

Acknowledgments: Portions of SAS' copyrighted SAS course content are reproduced here with permission of SAS Institute Inc., Cary, NC, USA. SAS Institute Inc. makes no warranties with respect to these materials and disclaims all liability therefor. Neither the GW nor the instructor shall be held liable or responsible to any person or entity with respect to any loss or incidental or consequential damages caused by anyone's use of the information or SAS codes contained herein. Circulating the handouts is strictly forbidden.

The PUT statement in a DATA step always writes to the SAS log, unless it is preceded by a FILE statement that specifies otherwise. [SAS® Documentation]

```
46 /* List values as a table and apply formats
47 to groups of variables. Route output to a file */
48 options nodate nonumber;
49 title;
50 data _null_;
51   set sashelp.class(obs=2);
52   file "E:\class_data.txt";
53   if _n_=1 then put @1 'NAME' @19 'SEX' @23 'AGE'
54                     @30 'HEIGHT' @38 'WEIGHT';
55   put (_all_)(1*$20.,1*$2.,1*3.,2*8.2);
56 run;
```

Line 52: This statement creates a tabular raw data file named CLASS_DATA.TXT in the path specified, all enclosed in quotation mark.

```
NOTE: The file "E:\class_data.txt" is:
      Filename=E:\class_data.txt,
      RECFM=V,LRECL=256,File Size (bytes)=0,
      Last Modified=20Aug2016:10:44:00,
      Create Time=20Aug2016:10:42:55

NOTE: 3 records were written to the file "E:\class_data.txt".
      The minimum record length was 41.
      The maximum record length was 43.
NOTE: There were 2 observations read from the data set SASHELP.CLASS.
```

Acknowledgments: Portions of SAS' copyrighted SAS course content are reproduced here with permission of SAS Institute Inc., Cary, NC, USA. SAS Institute Inc. makes no warranties with respect to these materials and disclaims all liability therefor. Neither the GW nor the instructor shall be held liable or responsible to any person or entity with respect to any loss or incidental or consequential damages caused by anyone's use of the information or SAS codes contained herein. Circulating the handouts is strictly forbidden.

Automatic variable _INFILE_ specifies a character variable that references the contents of the current input buffer for this INFILE statement.

You can use the variable in the same way as any other variable, even as the target of an assignment. The variable is automatically retained and initialized to blanks. As with automatic variables, the _INFILE_= variable is not written to the data set.

```
Data _Null_;
input name $ software $;
put _infile_;
datalines;
Mike R
John SAS
Rachel Python
;
run;
```

```
Mike R
John SAS
Rachel Python
```

_N_ and _ERROR_

These two variables are created by the SAS engine during compilation. These variables are used for processing, but are never written into the SAS data set created.

_N_ is initially set to 1. Each time the DATA step loops past the DATA statement, the variable _N_ increments by 1.

_N_ =1 means that the program left the DATA statement for the first time. If the data set has 5 observations, it will leave the DATA statement six times.

ERROR is 0 by default but is set to 1 whenever an error is encountered, such as an input data error, a conversion error, or a math error, as in division by 0 or a floating point overflow. You can use the value of this variable to help locate errors in data records and to print an error message to the SAS log.

Source: SAS Documentation

Acknowledgments: Portions of SAS' copyrighted SAS course content are reproduced here with permission of SAS Institute Inc., Cary, NC, USA. SAS Institute Inc. makes no warranties with respect to these materials and disclaims all liability therefor. Neither the GW nor the instructor shall be held liable or responsible to any person or entity with respect to any loss or incidental or consequential damages caused by anyone's use of the information or SAS codes contained herein. Circulating the handouts is strictly forbidden.

Creating a SAS List Report

Definition of a SAS List Report

A *list report* displays

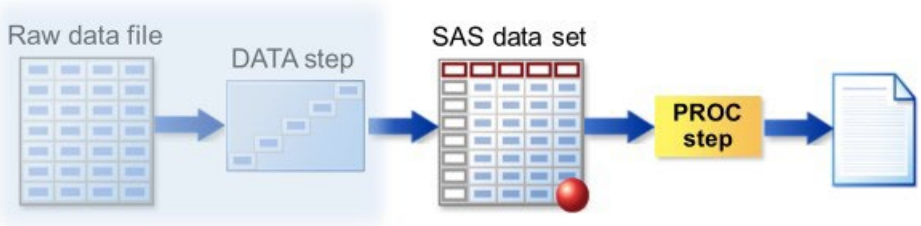
- the data in a SAS data set
- all observations (one per line) or only those specified
- all variables or only those specified.

5

Planning to Create a List Report

Now we can plan to create the list report.

Step 1	Name the SAS data set to be viewed.
Step 2	Determine the PROC step to use.
Step 3	Specify the variables to include in the report and their display order.



11

Acknowledgments: Portions of SAS' copyrighted SAS course content are reproduced here with permission of SAS Institute Inc., Cary, NC, USA. SAS Institute Inc. makes no warranties with respect to these materials and disclaims all liability therefor. Neither the GW nor the instructor shall be held liable or responsible to any person or entity with respect to any loss or incidental or consequential damages caused by anyone's use of the information or SAS codes contained herein. Circulating the handouts is strictly forbidden.

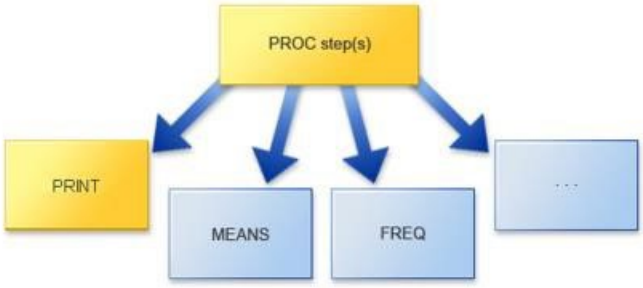
Planning to Create a List Report

Step 1 Name the SAS data set to be viewed.

12

Planning to Create a List Report

Step 2 Determine the PROC step to use.



Many procedures are available to create different types of reports. To create a simple list report, use the PRINT procedure.

13

Acknowledgments: Portions of SAS' copyrighted SAS course content are reproduced here with permission of SAS Institute Inc., Cary, NC, USA. SAS Institute Inc. makes no warranties with respect to these materials and disclaims all liability therefor. Neither the GW nor the instructor shall be held liable or responsible to any person or entity with respect to any loss or incidental or consequential damages caused by anyone's use of the information or SAS codes contained herein. Circulating the handouts is strictly forbidden.

Planning to Create a List Report

Step 3 Specify the variables to include in the report and their display order.

	Variable Names
1	EmployeeID
2	FirstName
3	LastName
4	JobCode
	Salary
	Category

The display order for the variables might not be the order in which they exist in the SAS data set.

14

PROC PRINT (a reporting procedure)

- displays data values (numeric or character)
- can provide summary totals for numeric variables (their values cannot be added, subtracted, multiplied or divided against each other)
- does not have OUT= data set option
- does not support the ODS OUTPUT statement

Acknowledgments: Portions of SAS' copyrighted SAS course content are reproduced here with permission of SAS Institute Inc., Cary, NC, USA. SAS Institute Inc. makes no warranties with respect to these materials and disclaims all liability therefor. Neither the GW nor the instructor shall be held liable or responsible to any person or entity with respect to any loss or incidental or consequential damages caused by anyone's use of the information or SAS codes contained herein. Circulating the handouts is strictly forbidden.

PROC PRINT Step

This code produces a default PROC PRINT report.

```
proc print data=pilotdata;
run;
```

PROC PRINT DATA=*SAS_data_set_name*;

Obs	Employee ID	FirstName	LastName	Job Code	Salary	Category
1	E01046	DAVID	CHAPMAN	PILOT1	72660	DOM
2	E01682	VICTOR	TAILOR	PILOT1	44980	DOM
3	E00746	MARTIN L.	DIXON	PILOT3	120330	INT
4	E02659	CLIFTON G.	WILDER	PILOT1	53630	DOM
5	E01642	NANCY A.	MCELROY	PILOT2	78260	DOM
. . .						
49	E02748	ALLAN	SUCHET	PILOT2	78260	INT
50	E02757	THOMAS E.	KRELLWITZ	PILOT3	122240	INT

By default, PROC PRINT displays ***all*** observations and variables in the data set.

in07d02

27

OBS is a column that is automatically displayed in a PROC PRINT report. It represents the number of an observation.

Controlling the Output Report

By default, PROC PRINT displays ***all*** of the variables in a SAS data set in the order in which they are stored.

Obs	Employee ID	FirstName	LastName	Job Code	Salary	Category
1	E01046	DAVID	CHAPMAN	PILOT1	72660	DOM
2	E01682	VICTOR	TAILOR	PILOT1	44980	DOM
3	E00746	MARTIN L.	DIXON	PILOT3	120330	INT
4	E02659	CLIFTON G.	WILDER	PILOT1	53630	DOM
5	E01642	NANCY A.	MCELROY	PILOT2	78260	DOM

How can you print only certain variables in the report?

How can you specify the order of variables in the report?

33

Acknowledgments: Portions of SAS' copyrighted SAS course content are reproduced here with permission of SAS Institute Inc., Cary, NC, USA. SAS Institute Inc. makes no warranties with respect to these materials and disclaims all liability therefor. Neither the GW nor the instructor shall be held liable or responsible to any person or entity with respect to any loss or incidental or consequential damages caused by anyone's use of the information or SAS codes contained herein. Circulating the handouts is strictly forbidden.

VAR Statement

The VAR statement names the variables to be displayed and specifies the desired order of the variables.

```
proc print data = pilotdata;
   var EmployeeID FirstName LastName JobCode;
run;
```

VAR *variables…;*

Obs	Employee ID	FirstName	LastName	Job Code
1	E01046	DAVID	CHAPMAN	PILOT1
2	E01682	VICTOR	TAILOR	PILOT1
3	E00746	MARTIN L.	DIXON	PILOT3
4	E02659	CLIFTON G.	WILDER	PILOT1
5	E01642	NANCY A.	MCELROY	PILOT2
6	E04732	CHRISTIAN	EDMINSTON	PILOT1

in07d03

34

Program to Add Labels

This code produces and specifies the labels for the report.

LABEL option

```
proc print data=pilotdata label;
   var EmployeeID FirstName
       LastName Salary;
   label EmployeeID='ID Number'
         FirstName='First Name'
         LastName='Last Name'
         Salary='Annual Salary';
run;
```

LABEL statement

Labels are ignored unless LABEL or SPLIT= is specified.

in08d03

25

Acknowledgments: Portions of SAS' copyrighted SAS course content are reproduced here with permission of SAS Institute Inc., Cary, NC, USA. SAS Institute Inc. makes no warranties with respect to these materials and disclaims all liability therefor. Neither the GW nor the instructor shall be held liable or responsible to any person or entity with respect to any loss or incidental or consequential damages caused by anyone's use of the information or SAS codes contained herein. Circulating the handouts is strictly forbidden.

SPLIT= Option

The labels are split at the asterisks. (The asterisk is not part of the label.)

```
proc print data=pilotdata split='*';
   var EmployeeID FirstName
       LastName Salary;
   label EmployeeID='ID*Number'
         FirstName='First*Name'
         LastName='Last*Name'
         Salary='Annual*Salary';
run;
```

PROC PRINT DATA=*SAS_data_set_name*
SPLIT='*split-character*';

It is ***not*** necessary to use ***both*** LABEL and SPLIT=.

in08d04

33

SPLIT= Option

The labels are split as specified.

Partial PROC PRINT Output

Obs	ID Number	First Name	Last Name	Annual Salary
1	E01046	DAVID	CHAPMAN	72660
2	E01682	VICTOR	TAILOR	44980
3	E00746	MARTIN L.	DIXON	120330
4	E02659	CLIFTON G.	WILDER	53630
5	E01642	NANCY A.	MCELROY	78260

34

Acknowledgments: Portions of SAS' copyrighted SAS course content are reproduced here with permission of SAS Institute Inc., Cary, NC, USA. SAS Institute Inc. makes no warranties with respect to these materials and disclaims all liability therefor. Neither the GW nor the instructor shall be held liable or responsible to any person or entity with respect to any loss or incidental or consequential damages caused by anyone's use of the information or SAS codes contained herein. Circulating the handouts is strictly forbidden.

FORMAT Statement

Adding the FORMAT statement displays the annual salary values with a dollar sign and comma.

```
proc print data=pilotdata split= '*';
   var EmployeeID FirstName
       LastName Salary;
   label EmployeeID='ID*Number'
         FirstName='First*Name'
         LastName='Last*Name'
         Salary='Annual*Salary';
   format Salary dollar8.0;
run;
```

Obs	ID Number	First Name	Last Name	Annual Salary
1	E01046	DAVID	CHAPMAN	$72,660
2	E01682	VICTOR	TAILOR	$44,980
3	E00746	MARTIN L.	DIXON	$120,330
4	E02659	CLIFTON G.	WILDER	$53,630
5	E01642	NANCY A.	MCELROY	$78,260

in08d06

43

FORMAT Statement

Formats associated with variables in a PROC step remain in effect only for that step.

```
proc print data=pilotdata;
   format Salary dollar8.;
run;

proc print data=pilotdata;
run;
```

Salary is formatted for this output.

Salary is ***not*** formatted for this output.

in08d07

44

Acknowledgments: Portions of SAS' copyrighted SAS course content are reproduced here with permission of SAS Institute Inc., Cary, NC, USA. SAS Institute Inc. makes no warranties with respect to these materials and disclaims all liability therefor. Neither the GW nor the instructor shall be held liable or responsible to any person or entity with respect to any loss or incidental or consequential damages caused by anyone's use of the information or SAS codes contained herein. Circulating the handouts is strictly forbidden.

Example of the PRINT Procedure

```
options nodate nonumber ps=30 ls=64;

proc print data=sashelp.shoes noobs split='*';
   var subsidiary product inventory sales;
   where product='Boot' or product='Sandal';
   sum inventory sales;
   by region;
   pageby region;
   label inventory='Total*Inventory'
         sales='Total*Sales';
   format inventory sales dollar14.2;
   title 'Boot and Sandal Report';
   footnote 'Created by Tony Smith';
   footnote2 'Chicago, IL';
run;
```

7

NOOBS Option

The *NOOBS option* suppresses the column in the output that identifies each observation by number.

```
proc print data=sashelp.shoes noobs split='*';
```

By default, the PRINT procedure gives an observation column.

11

Acknowledgments: Portions of SAS' copyrighted SAS course content are reproduced here with permission of SAS Institute Inc., Cary, NC, USA. SAS Institute Inc. makes no warranties with respect to these materials and disclaims all liability therefor. Neither the GW nor the instructor shall be held liable or responsible to any person or entity with respect to any loss or incidental or consequential damages caused by anyone's use of the information or SAS codes contained herein. Circulating the handouts is strictly forbidden.

NOOBS Option

Without NOOBS

Partial Output

```
                    Boot and Sandal Report

- Region=Asia -------------------------

                                   Total            Total
  Obs   Subsidiary   Product      Inventory         Sales

   57    Bangkok     Boot         $9,576.00       $1,996.00
   59    Bangkok     Sandal      $15,087.00       $3,230.00
   62    Seoul       Boot       $160,589.00      $60,712.00
   65    Seoul       Sandal      $21,483.00       $4,978.00
------
Region
```

With NOOBS

Partial Output

```
                    Boot and Sandal Report

- --- Region=Asia -------------------------

                                   Total            Total
  Subsidiary    Product          Inventory         Sales

   Bangkok      Boot              $9,576.00       $1,996.00
   Bangkok      Sandal           $15,087.00       $3,230.00
   Seoul        Boot            $160,589.00      $60,712.00
   Seoul        Sandal           $21,483.00       $4,978.00
  ----------                   --------------   --------------
   Region                      $206,735.00      $70,916.00
```

12

VAR Statement

The *VAR statement* selects variables that appear in the report and determines the order of the variables.

```
var subsidiary product inventory sales;
```

```
                    Boot and Sandal Report

------------------------- Region=Asia ---------------------------

                                   Total            Total
  Subsidiary    Product          Inventory         Sales

   Bangkok      Boot              $9,576.00       $1,996.00
   Bangkok      Sandal           $15,087.00       $3,230.00
   Seoul        Boot            $160,589.00      $60,712.00
   Seoul        Sandal           $21,483.00       $4,978.00
  ----------                   --------------   --------------
   Region                      $206,735.00      $70,916.00
```

By default, the PRINT procedure displays all variables in the order that the variables are stored in the data set.

13

Acknowledgments: Portions of SAS' copyrighted SAS course content are reproduced here with permission of SAS Institute Inc., Cary, NC, USA. SAS Institute Inc. makes no warranties with respect to these materials and disclaims all liability therefor. Neither the GW nor the instructor shall be held liable or responsible to any person or entity with respect to any loss or incidental or consequential damages caused by anyone's use of the information or SAS codes contained herein. Circulating the handouts is strictly forbidden.

WHERE Statement

The *WHERE statement* subsets the input data set by specifying certain conditions that each observation must meet before it is available for the report.

```
where product='Boot' or product='Sandal';
```

- The WHERE statement does not alter the original data set.
- Use only one WHERE statement in a step unless you use a WHERE SAME AND or WHERE ALSO statement with a WHERE statement.
- Character values are case sensitive.

16

WHERE Statement

Examples:

```
where sales > 100000;
where sales eq .;
where name = 'Smith';
where name = ' ';
where sales ge 100000 and name = 'Smith';
where sales ge 100000 or name = 'Smith';
where revenue >= 150 and revenue <= 999;
where revenue between 150 and 999;
where revenue not between 150 and 999;
where month contains 'uary';
where birthdate > '11JUL1968'd;
```

19

Acknowledgments: Portions of SAS' copyrighted SAS course content are reproduced here with permission of SAS Institute Inc., Cary, NC, USA. SAS Institute Inc. makes no warranties with respect to these materials and disclaims all liability therefor. Neither the GW nor the instructor shall be held liable or responsible to any person or entity with respect to any loss or incidental or consequential damages caused by anyone's use of the information or SAS codes contained herein. Circulating the handouts is strictly forbidden.

SUM Statement

The *SUM statement* totals values of numeric variables.

```
sum inventory sales;
```

Partial Output

```
                    Boot and Sandal Report

------------------------- Region=Asia -------------------------

                                   Total            Total
  Subsidiary     Product         Inventory            Sales

   Bangkok       Boot            $9,576.00        $1,996.00
   Bangkok       Sandal         $15,087.00        $3,230.00
   Seoul         Boot          $160,589.00       $60,712.00
   Seoul         Sandal         $21,483.00        $4,978.00
  ----------                 --------------    --------------
   Region                      $206,735.00       $70,916.00
```

The SUM statement always gives grand totals and gives subtotals if used with a BY statement.

22

BY Statement

The *BY statement* produces a separate section of the report for each BY group.

```
by region;
```

Partial Output

```
                    Boot and Sandal Report

------------------------- Region=Asia -------------------------

                                   Total            Total
  Subsidiary     Product         Inventory            Sales

   Bangkok       Boot            $9,576.00        $1,996.00
   Bangkok       Sandal         $15,087.00        $3,230.00
   Seoul         Boot          $160,589.00       $60,712.00
   Seoul         Sandal         $21,483.00        $4,978.00
  ----------                 --------------    --------------
   Region                      $206,735.00       $70,916.00
```

Data must be indexed or sorted to use a BY statement.

23

Acknowledgments: Portions of SAS' copyrighted SAS course content are reproduced here with permission of SAS Institute Inc., Cary, NC, USA. SAS Institute Inc. makes no warranties with respect to these materials and disclaims all liability therefor. Neither the GW nor the instructor shall be held liable or responsible to any person or entity with respect to any loss or incidental or consequential damages caused by anyone's use of the information or SAS codes contained herein. Circulating the handouts is strictly forbidden.

SUM and BY Statements

Partial Output

---------- Region=Western Europe ---------- **BY Group**

Subsidiary	Product	Total Inventory	Total Sales
Copenhagen	Boot	$4,657.00	$1,663.00
Geneva	Boot	$171,030.00	$41,341.00
Geneva	Sandal	$3,529.00	$736.00
Heidelberg	Boot	$301,779.00	$65,610.00
Heidelberg	Sandal	$4,618.00	$977.00
Lisbon	Boot	$341,911.00	$76,349.00
Lisbon	Sandal	$24,253.00	$1,650.00
London	Boot	$289,527.00	$54,449.00
London	Sandal	$11,111.00	$5,217.00
Madrid	Boot	$1,027.00	$1,179.00
Paris	Boot	$41,506.00	$19,196.00
Paris	Sandal	$23,816.00	$1,520.00
Rome	Boot	$209,271.00	$36,244.00
Rome	Sandal	$4,611.00	$1,249.00
Region		$1,432,646.00	$307,380.00 **Subtotal**
		$12,956,946.00	$3,218,979.00 **Grand Total**

24

Acknowledgments: Portions of SAS' copyrighted SAS course content are reproduced here with permission of SAS Institute Inc., Cary, NC, USA. SAS Institute Inc. makes no warranties with respect to these materials and disclaims all liability therefor. Neither the GW nor the instructor shall be held liable or responsible to any person or entity with respect to any loss or incidental or consequential damages caused by anyone's use of the information or SAS codes contained herein. Circulating the handouts is strictly forbidden.

PAGEBY Statement

The *PAGEBY statement* puts each separate section of a BY group on separate pages.

```
pageby region;
```

The PAGEBY statement must name a variable that appears in the BY statement.

25

PAGEBY Statement

Partial Output

```
                    Boot and Sandal Report

---------------------- Region=Asia --------------------------
```

```
                    Boot and Sandal Report

---------------- Region=Eastern Europe ---------------------
```

```
                    Boot and Sandal Report

------------------ Region=Middle East ----------------------

                                   Total            Total
Subsidiary     Product           Inventory          Sales

Al-Khobar      Boot            $44,658.00       $15,062.00
Al-Khobar      Sandal          $13,343.00        $1,380.00
Dubai          Boot           $403,259.00       $90,972.00
Dubai          Sandal          $59,985.00       $17,492.00
Tel Aviv       Boot           $222,165.00       $65,248.00
Tel Aviv       Sandal          $71,094.00       $16,314.00
----------                   --------------   --------------
Region                        $814,504.00      $206,468.00
```

26

Acknowledgments: Portions of SAS' copyrighted SAS course content are reproduced here with permission of SAS Institute Inc., Cary, NC, USA. SAS Institute Inc. makes no warranties with respect to these materials and disclaims all liability therefor. Neither the GW nor the instructor shall be held liable or responsible to any person or entity with respect to any loss or incidental or consequential damages caused by anyone's use of the information or SAS codes contained herein. Circulating the handouts is strictly forbidden.

ID Statement

The *ID statement* specifies the variable(s) to print at the beginning of each row instead of an observation number.

```
id region;
by region;
```

When used with a BY statement, the ID statement eliminates the BY line and suppresses repetitious printing of the BY variable(s).

Partial Output

Region	Subsidiary	Product	Total Inventory	Total Sales
Western Europe	Copenhagen	Boot	$4,657.00	$1,663.00
	Geneva	Boot	$171,030.00	$41,341.00
	Geneva	Sandal	$3,529.00	$736.00
	Heidelberg	Boot	$301,779.00	$65,610.00
	Heidelberg	Sandal	$4,618.00	$977.00

29

LABEL Statement

The *LABEL statement* assigns descriptive labels to variable names.

```
label inventory='Total*Inventory'
      sales='Total*Sales';
```

Partial Output

Boot and Sandal Report

---------- Region=Asia ----------

Obs	Subsidiary	Product	Total Inventory	Total Sales
57	Bangkok	Boot	$9,576.00	$1,996.00
59	Bangkok	Sandal	$15,087.00	$3,230.00
62	Seoul	Boot	$160,589.00	$60,712.00
65	Seoul	Sandal	$21,483.00	$4,978.00
Region			$206,735.00	$70,916.00

30 A label can be up to 256 characters.

Acknowledgments: Portions of SAS' copyrighted SAS course content are reproduced here with permission of SAS Institute Inc., Cary, NC, USA. SAS Institute Inc. makes no warranties with respect to these materials and disclaims all liability therefor. Neither the GW nor the instructor shall be held liable or responsible to any person or entity with respect to any loss or incidental or consequential damages caused by anyone's use of the information or SAS codes contained herein. Circulating the handouts is strictly forbidden.

LABEL Statement

The LABEL option enforces variables' labels as column headings.

```
proc print data=sashelp.shoes noobs label;
```

The SPLIT= option specifies the split character, which controls line breaks in column headers and implies the use of labels.

```
proc print data=sashelp.shoes noobs split='*';
```

33

FORMAT Statement

The *FORMAT statement* associates formats to variable values.

```
format inventory sales dollar14.2;
```

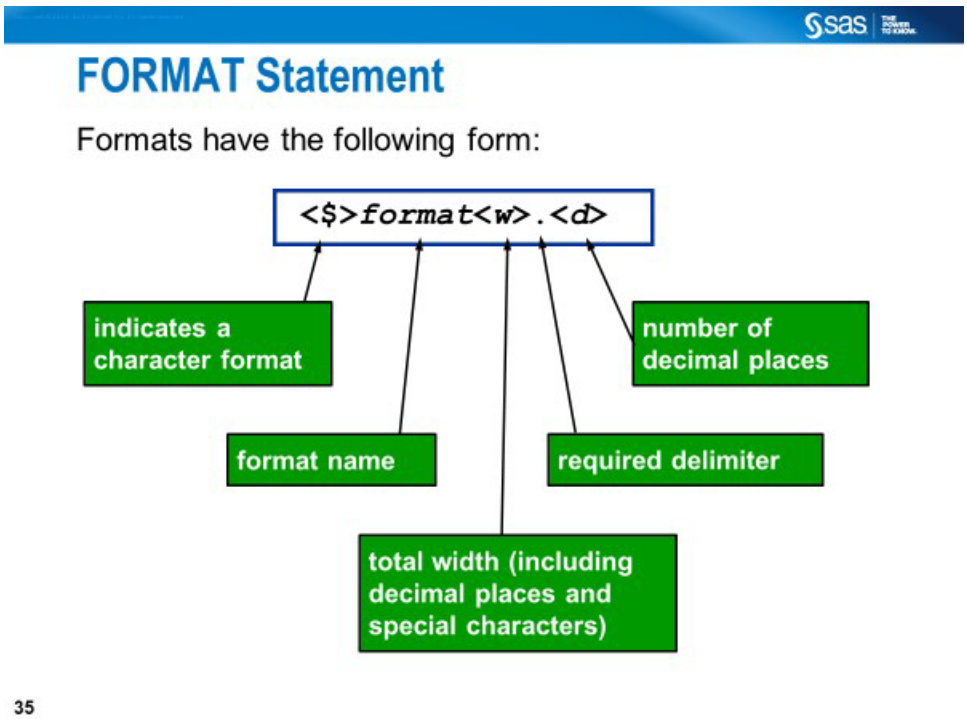
Partial Output

```
------------------------- Region=Asia -------------------------

                                        Total            Total
  Obs   Subsidiary   Product        Inventory            Sales

   57    Bangkok     Boot            $9,576.00        $1,996.00
   59    Bangkok     Sandal         $15,087.00        $3,230.00
   62    Seoul       Boot          $160,589.00       $60,712.00
   65    Seoul       Sandal         $21,483.00        $4,978.00
------                             -------------    -------------
Region                             $206,735.00       $70,916.00
```

A *format* is an instruction that SAS uses to write data values.

34

Acknowledgments: Portions of SAS' copyrighted SAS course content are reproduced here with permission of SAS Institute Inc., Cary, NC, USA. SAS Institute Inc. makes no warranties with respect to these materials and disclaims all liability therefor. Neither the GW nor the instructor shall be held liable or responsible to any person or entity with respect to any loss or incidental or consequential damages caused by anyone's use of the information or SAS codes contained herein. Circulating the handouts is strictly forbidden.



Acknowledgments: Portions of SAS' copyrighted SAS course content are reproduced here with permission of SAS Institute Inc., Cary, NC, USA. SAS Institute Inc. makes no warranties with respect to these materials and disclaims all liability therefor. Neither the GW nor the instructor shall be held liable or responsible to any person or entity with respect to any loss or incidental or consequential damages caused by anyone's use of the information or SAS codes contained herein. Circulating the handouts is strictly forbidden.

FORMAT Statement

Stored Value	Format	Displayed Value
Washington	$4.	Wash
1234.4567	8.0	1234
1234.4567	8.2	1234.46
1234.4567	comma8.2	1,234.46
1234.4567	dollar9.2	$1,234.46

36

FORMAT Statement

A

Partial Output

What minimum widths are needed to complete the FORMAT statement for this desired output?

```
-------------------- Region=Western Europe --------------------

                                    Total              Total
Subsidiary     Product           Inventory              Sales

Copenhagen     Boot               $4,657.00          $1,663.00
Geneva         Boot             $171,030.00         $41,341.00
               dal                $3,529.00            $736.00
               t                $301,779.00         $65,610.00
               dal                $4,618.00            $977.00
               t                $341,911.00         $76,349.00
               dal               $24,253.00          $1,650.00
               t                $289,527.00         $54,449.00
               dal               $11,111.00          $5,217.00
Madrid         Boot               $1,027.00          $1,179.00
Paris          Boot              $41,506.00         $19,196.00
Paris          Sandal            $23,816.00          $1,520.00
Rome           Boot             $209,271.00         $36,244.00
Rome           Sandal             $4,611.00          $1,249.00
----------                    --------------     --------------
Region                         $1,432,646.00        $307,380.00
                               ==============     ==============
                              $12,956,946.00      $3,218,979.00
```

```
format inventory dollar14.2 sales dollar13.2;
```

Acknowledgments: Portions of SAS' copyrighted SAS course content are reproduced here with permission of SAS Institute Inc., Cary, NC, USA. SAS Institute Inc. makes no warranties with respect to these materials and disclaims all liability therefor. Neither the GW nor the instructor shall be held liable or responsible to any person or entity with respect to any loss or incidental or consequential damages caused by anyone's use of the information or SAS codes contained herein. Circulating the handouts is strictly forbidden.

FORMAT Statement

Stored Value	Format	Displayed Value
17332	mmddyy6.	061507
17332	mmddyy8.	06/15/07
17332	mmddyy10.	06/15/2007
17332	date7.	15JUN07
17332	date9.	15JUN2007
17332	ddmmyy8.	15/06/07
17332	worddate.	June 15, 2007
17332	weekdate.	Friday, June 15, 2007
17332	monyy7.	JUN2007

41

Acknowledgments: Portions of SAS' copyrighted SAS course content are reproduced here with permission of SAS Institute Inc., Cary, NC, USA. SAS Institute Inc. makes no warranties with respect to these materials and disclaims all liability therefor. Neither the GW nor the instructor shall be held liable or responsible to any person or entity with respect to any loss or incidental or consequential damages caused by anyone's use of the information or SAS codes contained herein. Circulating the handouts is strictly forbidden.

LABEL and FORMAT Statements

LABEL and FORMAT statements assigned in a PROC step are considered ***temporary*** attributes (apply only for the duration of the step).

LABEL and FORMAT statements assigned in a DATA step are considered ***permanent*** attributes (stored in the descriptor portion).

```
          Alphabetic List of Variables and Attributes

#  Variable    Type  Len  Format     Informat   Label

6  Inventory   Num     8  DOLLAR12.  DOLLAR12.  Total Inventory
2  Product     Char   14
1  Region      Char   25
7  Returns     Num     8  DOLLAR12.  DOLLAR12.  Total Returns
5  Sales       Num     8  DOLLAR12.  DOLLAR12.  Total Sales
4  Stores      Num     8                        Number of Stores
3  Subsidiary  Char   12
```

44

TITLE Statement

The *TITLE statement* specifies up to 10 lines of text at the top of output.

```
title 'Boot and Sandal Report';
```

Partial Output

```
                    Boot and Sandal Report

------------------------- Region=Asia -------------------------

                                   Total            Total
   Subsidiary    Product         Inventory            Sales

    Bangkok      Boot            $9,576.00        $1,996.00
    Bangkok      Sandal         $15,087.00        $3,230.00
    Seoul        Boot          $160,589.00       $60,712.00
    Seoul        Sandal         $21,483.00        $4,978.00
   ----------                 -------------    -------------
    Region                     $206,735.00       $70,916.00
```

TITLE is the same as TITLE1.

45

Acknowledgments: Portions of SAS' copyrighted SAS course content are reproduced here with permission of SAS Institute Inc., Cary, NC, USA. SAS Institute Inc. makes no warranties with respect to these materials and disclaims all liability therefor. Neither the GW nor the instructor shall be held liable or responsible to any person or entity with respect to any loss or incidental or consequential damages caused by anyone's use of the information or SAS codes contained herein. Circulating the handouts is strictly forbidden.