SAS Workshop

Introduction to SAS Programming

Session I

Iowa State University

Structure of a SAS Program

- SAS programs consist of SAS statements
- General Form: **SAS_keyword** ***operand*****;**
  e.g. **PROC** ***ANOVA DATA=CORN*****;**
- SAS statements are interpreted and executed in their order of appearance

However, blocks of statements called *steps*, that define one of two basic activities, are executed as groups

DATA step:
statements leading to the creation of a SAS data set

PROC step:
statements needed to tell a SAS procedure to perform a statistical analysis

A SAS program consists of several logically related DATA and PROC steps

```
data oranges;
   input Variety $ Flavor Texture Looks;
   Total=Flavor+Texture+Looks;
datalines;
navel 9 8 6
temple 7 7 7
valencia 8 9 9
mandarin 5 7 8
;
proc sort data=oranges;
   by descending Total;
run;
proc print data=oranges;
   title 'Taste Test Results for Oranges';
run;
```

SAS Data Set

A *SAS data set* (or table) is a rectangular table of rows and columns.

Customer_Name	Customer_Age	Order_Date	Total_Retail_Price
James Klisurich	38	11JAN2003	$16.50
Sandrina Stephano	28	15JAN2003	$247.50
Dianne Patchin	28	20JAN2003	$28.30
Wendell Summersby	43	28JAN2003	$32.00
Duncan Robertshawe	63	27FEB2003	$63.60
Najma Hicks	21	02MAR2003	$234.60

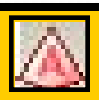
Rows (observations)

Columns (variables)

SAS Data Set

All variables must have a *name*, *type*, and *length*. A variable's *type* is either **character** (string) or **numeric**. The type plays a role in determining the *length*.

Customer_Name	Customer_Age	Order_Date	Total_Retail_Price
James Klisurich	38	15716	16.50
Sandrina Stephano	28	15720	247.50
Dianne Patchin	28	15725	28.30
Wendell Summersby	43	15733	32.00
Duncan Robertshawe	63	15763	63.60
Najma Hicks	21	15766	234.60



Character values are 1 to 32,767 characters (bytes) long.

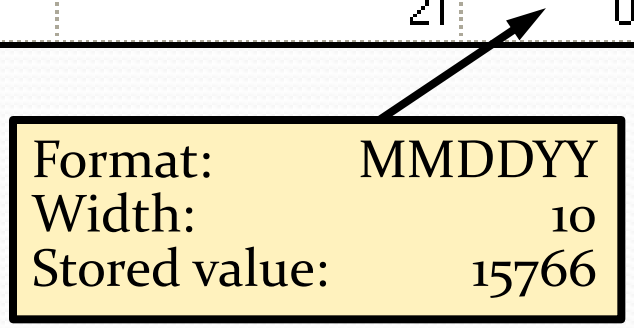
Numeric values are 8 bytes of floating point storage:

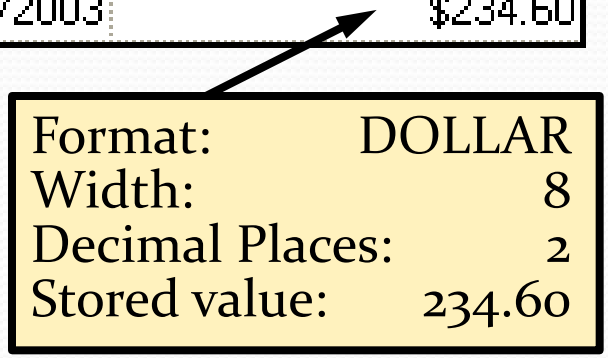
- **Numeric**
- **Currency**
- **Date (days from 01JAN1960)**
- **Time (seconds from midnight)**

SAS Data Set

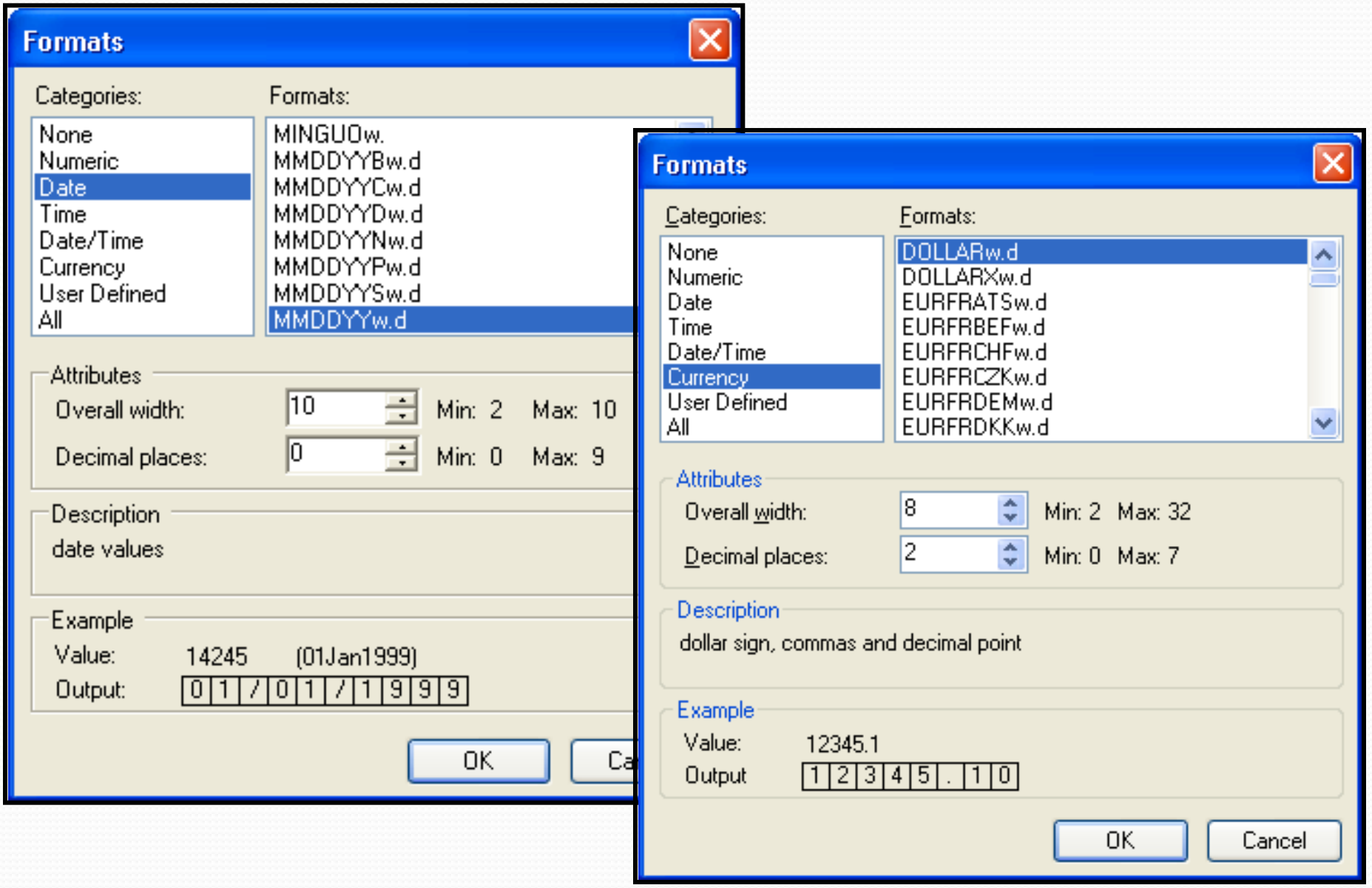
A *format* is used to control how values are displayed. Formats do not affect how values are stored.

Customer_Name	Customer_Age	Order_Date	Total_Retail_Price
James Klisurich	38	01/11/2003	$16.50
Sandrina Stephano	28	01/15/2003	$247.50
Dianne Patchin	28	01/20/2003	$28.30
Wendell Summersby	43	01/28/2003	$32.00
Duncan Robertshawe	63	02/27/2003	$63.60
Najma Hicks	21	03/02/2003	$234.60





Formats


Formats
Categories:
Formats:
None
Numeric
Date
Time
Date/Time
Currency
User Defined
All
MINGUOw.
MMDDYYBw.d
MMDDYYCw.d
MMDDYYDw.d
MMDDYYNw.d
MMDDYYPw.d
MMDDYYSw.d
MMDDYYw.d
Attributes
Overall width:
10
Min: 2
Max: 10
Decimal places:
0
Min: 0
Max: 9
Description
date values
Example
Value:
14245
(01Jan1999)
Output:
0 1 / 0 1 / 1 9 9 9
OK
Formats
Categories:
Formats:
None
Numeric
Date
Time
Date/Time
Currency
User Defined
All
DOLLARw.d
DOLLARXw.d
EURFRATSw.d
EURFRBEFw.d
EURFRCHFw.d
EURFRCZKw.d
EURFRDEMw.d
EURFRDKKw.d
Attributes
Overall width:
8
Min: 2
Max: 32
Decimal places:
2
Min: 0
Max: 7
Description
dollar sign, commas and decimal point
Example
Value:
12345.1
Output
1 2 3 4 5 . 1 0
OK
Cancel

Missing Values

If a data value is not present for a variable in a particular observation, it is considered *missing*.

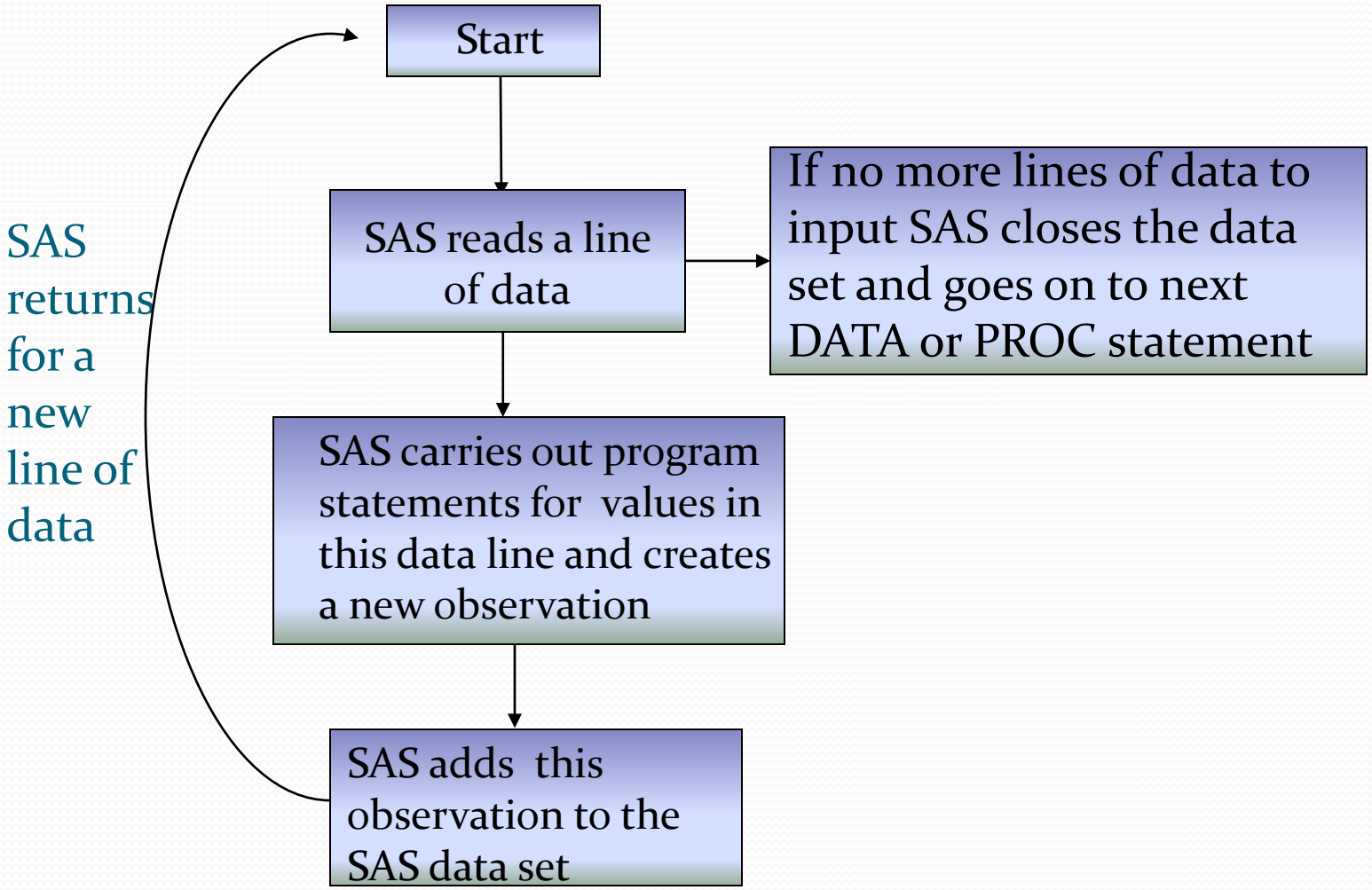
- A missing character value is displayed as a blank.
- A missing numeric value is displayed as a period.

Customer_Name	Customer_Age	Order_Date	Total_Retail_Price
James Klisurich	38	11JAN2003	$16.50
Sandrina Stephano	.	15JAN2003	$247.50
Dianne Patchin	28	20JAN2003	$28.30
Wendell Summersby	43	28JAN2003	$32.00
	63	27FEB2003	$63.60
Najma Hicks	21	02MAR2003	$234.60

SAS Data Step

- Begins with the statement
  - DATA *name* ;
- Followed by **one** of these statements:
  - INPUT ;
  - SET ;
- A SAS data step is (usually) used to create a new **SAS** *data set* from
  - external data (using an INPUT statement)
  - another SAS data set (using a SET statement)
- **SAS** *program statements* are used in a SAS data step to modify input data, if necessary

SAS Data Step: Flow of Operations



SAS Data Step: Flow of Operations

```
data oranges;
  input Variety $ Flavor Texture Looks;
  Total=Flavor+Texture+Looks;
datalines;
navel 9 8 6
temple 7 7 7
valencia 8 9 9
mandarin 5 7 8
;
```

Variety	Flavor	Texture	Looks	Total
navel	9	8	6	23
temple	7	7	7	21

SAS Program Statements

```
Y1 = X1+X2**2;
Y2 = ABS(X3) ;
Y3 = SQRT(X4 + 4.0*X5**2) –X6;
X7=3.14156*log(X7);
```

```
IF INCOME = .   THEN DELETE  ;
IF STATE = 'CA'  | STATE ='OR' THEN
                          REGION = 'PACIFIC COAST' ;
IF  SCORE < 0  THEN SCORE = 0;
```

```
IF SCORE < 80 THEN WEIGHT=.67;
ELSE WEIGHT=.75;
```

```
WEIGHT = (SCORE < 80 ) * .67 + (SCORE >=80) * .75;
```

SAS Program Statements

```
IF SCORE < 80 THEN  DO;
        WEIGHT =0.67;
        RATE=5.70;
    END;
    ELSE DO;
        WEIGHT =0.75;
        RATE=6.50;
    END;
```

```
DATA ;
INPUT X 1 – X 5 ;
X6 = (X 4+X 5) / 2 ;
DROP  X 4  X 5 ;
DATALINES ;
          :
```

SAS Functions

A *SAS function* is a routine that returns a value that is determined from specified arguments.

General form of a SAS function:

function-name(*argument1*,*argument2*, . . .)

- Example:

```
sum(Salary,Bonus)
```

Using SAS Functions

- SAS functions can do the following:
  - perform arithmetic operations
  - compute sample statistics (for example: sum, mean, and standard deviation)
  - manipulate SAS dates
  - process character values
  - perform many other tasks

Sample statistics functions ignore missing values.

```
data oranges;
   input Variety $ Flavor Texture Looks;
   Average=mean(Flavor,Texture,Looks);
datalines;
navel 9 8 6
temple 7 7 7
valencia 8 9 9
mandarin 5 7 8
;
proc sort data=oranges;
   by descending Average;
run;
proc print data=oranges;
   title 'Taste Test Results for Oranges';
run;
```

Simple INPUT Statements

- List Input

```
INPUT   ID   SEX   $   AGE   WEIGHT ;
1342      F     27    121.2

INPUT   SCORE1-SCORE4  ;
63.1      94   87.5   72
```

- Formatted Input

```
INPUT  ID  4.  STATE  $2.  FERT  5.2  PERCENT  3.2 ;
0001IA _ _504089

INPUT  @10  ITEM  $4.  +5  PRICE  6.2;
xxxxxxxxxR2D2xxxxxb91350

INPUT (ID  SEX  AGE  WT  HT)  (3.  $1.  2.  2*5.1);
123M21_1650_ _721
```

The general form of the *Informats* we used above:

w. $w. w.d

Examples:

4. $2. 5.2

- Column Input

  INPUT ID 1-4 STATE $ 5-6 FERT 7-12 PERCENT 13-15 .2;

  0001IAbb5.04b89

Order of Evaluating Expressions

- Rule 1: Expressions within *parenthesis* are evaluated first

- Rule 2: Higher *priority* operators are performed first

  Group I **, + (prefix), – (prefix), ^(NOT), ><, <>
  Group II *, /
  Group III + (infix), –(infix)
  Group IV | |
  Group V <, <=, =, ^=, >=, >, ^>, ^<
  Group VI & (AND)
  Group VII | (OR)

- Rule 3: For operators with the same priority, the operations take place from left to right of the expression (except for Group I operators, which are executed right to left.)

Bonus Data Input Examples Example A2

```
data reaction;
length Concentration $4;
do Amount =.9 to .6 by -.1;
  do Concentration = '1%' , '1.5%' , '2%' , '2.5%' , '3%' ;
  input Time @@;
  output;
  end;
end;
datalines;
10.9 11.5 9.8 12.7 10.6
 9.2 10.3 9.0 10.6  9.4
 8.7  9.7 8.2  9.4  8.5
 7.2  8.6 7.5  9.7  7.7
;
proc print;
title 'Reaction times for biological substrate';
run;
```

Example A3

```
data ledger;
retain Store Region Month;
input Type $1. @;
if Type='S' then input @3 Store 4. Region $10. Month : $8. ;
else do;
      input @4 Date ddmmyy8. Sales 7.2;
      output;
      end;
drop Type;
datalines;
S 0021 Southeast  March
  10/05/04 134510
  12/05/04  23675
  21/05/04  96860
  28/05/04 265036
S 0173 Northwest  January
  15/05/04  67200
  18/05/04 158325
  29/05/04 127950
  30/05/04  45845
  02/06/04 304730
;
proc print data=ledger;
id Store;
format Store z4. Date ddmmyy8. Sales dollar10.2 ;
title 'Sales analysis for Martin & Co.';
run;
```