

# STA 311 Statistical Computing & Data Management

Instructor: Cheng Peng  
Department of Mathematics  
West Chester University  
West Chester, PA 19383

Office: 25 University Avenue, RM 111  
Phone: 610.436.2369  
Email: [cpeng@wcupa.edu](mailto:cpeng@wcupa.edu)

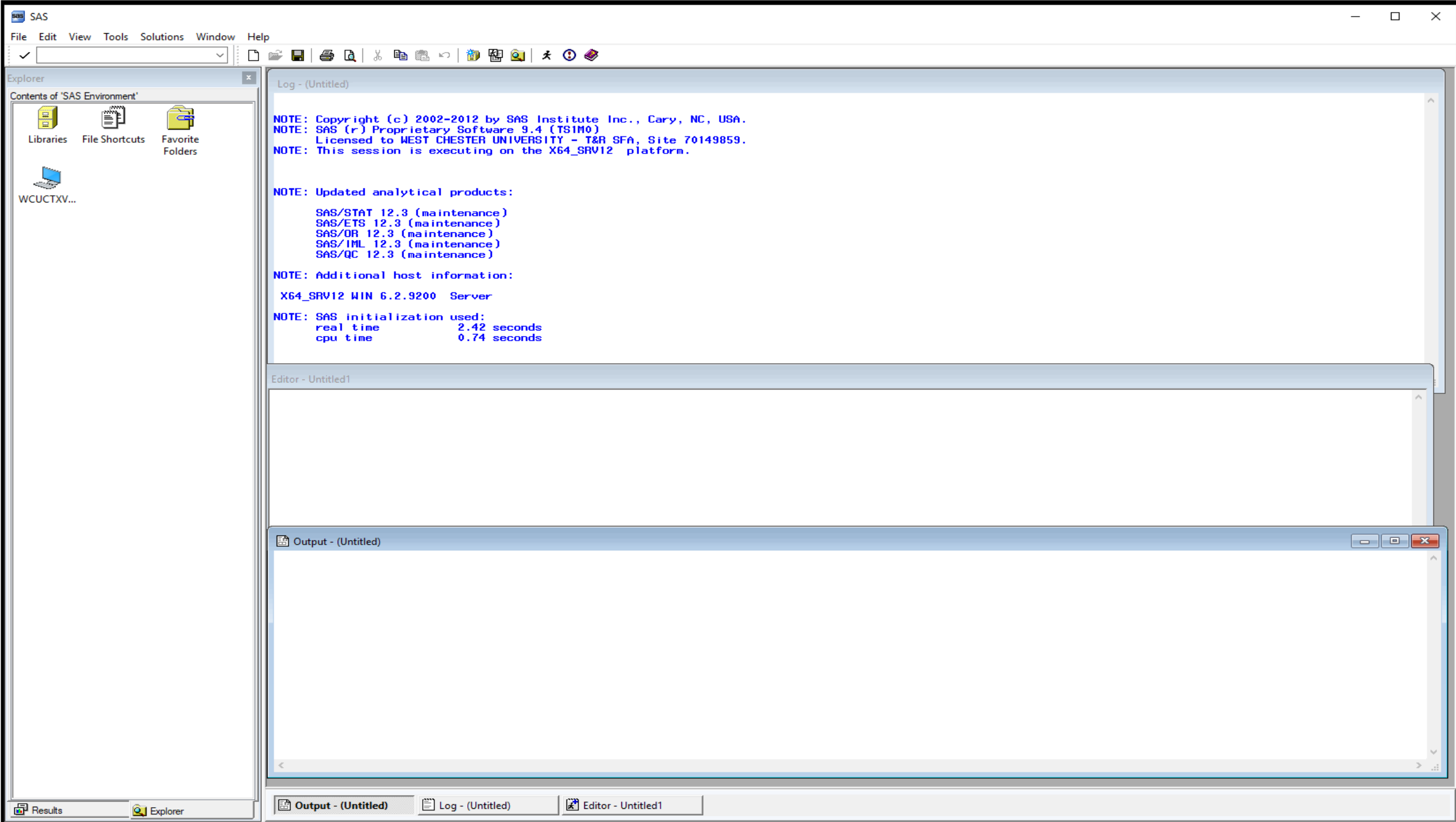
# Course Objectives

❑ To familiarize you with programming in the SAS language. After completing this course, you should be able to:

1. Create SAS data sets from multiple sources, including direct input, external text files, and dataset generated from other applications.
2. Write SAS data sets out in appropriate format and stored it in certain directory.
3. Use appropriate techniques to combine data sets, subset data sets, extract certain observations from datasets
4. Create basic SAS graphs and generate basic reports using appropriate procedures.
5. Perform basic statistical analyses of data.
6. Prepare for the Base SAS Programming Certification Exam

## What is SAS?

- SAS® is a statistical analysis package. A collection of computer “procedures” and a very powerful data management tool that facilitates data and statistical analysis.
- SAS® is not a menu-driven or command driven application. Rather it relies on user-written scripts or “programs” that are processed when requested to know what to do.
- Because it is a script-based application, the key to being successful using SAS is learning the rule and tricks of writing scripts.



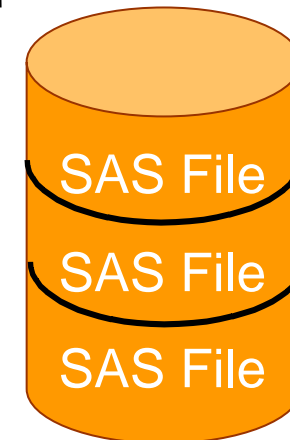
# SAS Data Libraries

A SAS data library is a collection of SAS files that are recognized as a unit by SAS.

**SAS Data Library**

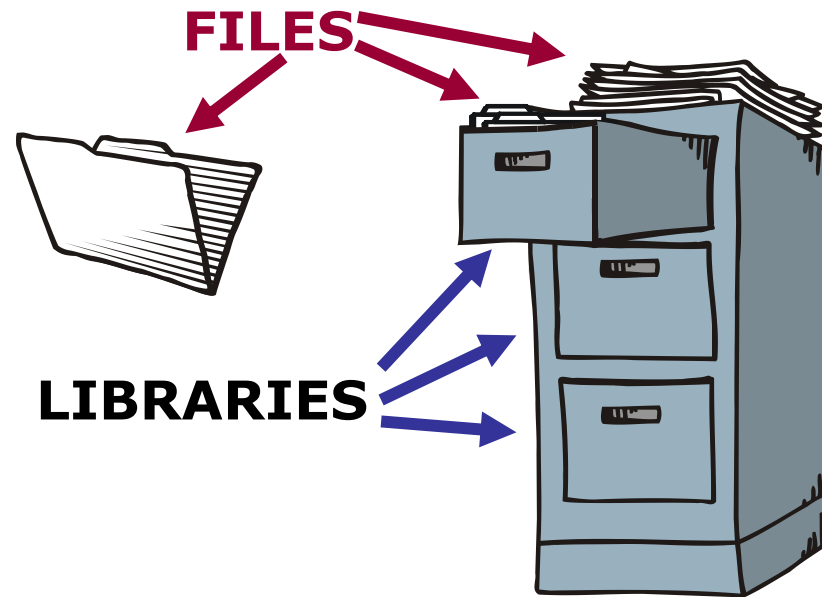
c:\mysasfiles

A SAS data set is a type of SAS file.



# SAS Data Libraries

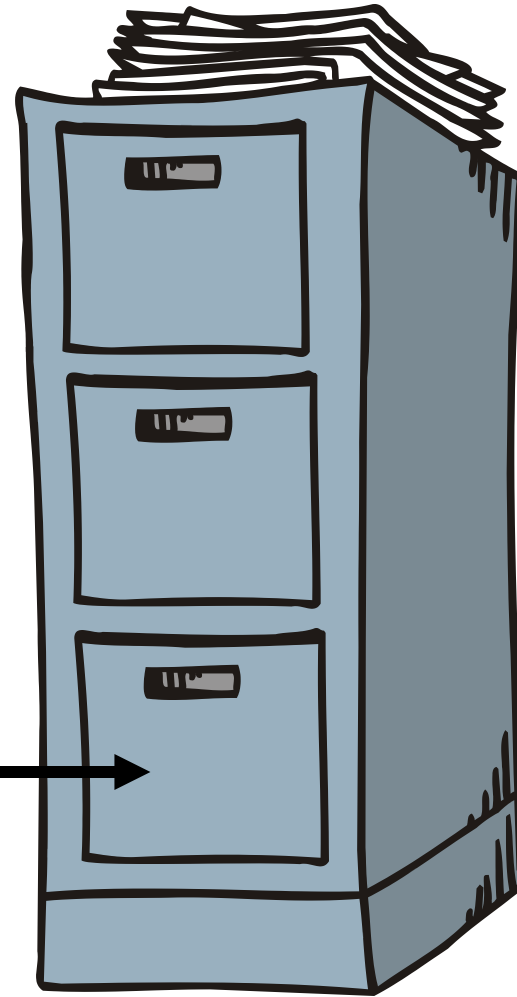
You can think of a SAS data library as a drawer in a filing cabinet and a SAS data set as one of the file folders in the drawer.



## Assigning a LIBREF

Regardless of which host operating system you use, you identify SAS data libraries by assigning each a library reference name (libref).

libref



# SAS Data Libraries

When we invoke SAS, we automatically have access to a temporary and a permanent SAS data library.

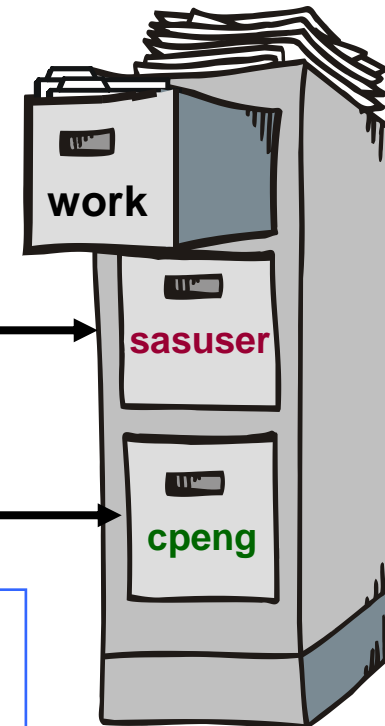
**work** - temporary library



**sasuser** - permanent library



**cpeng** - permanent library



**We can create and access our own permanent libraries.**



## Assigning a LIBREF

WE can use the LIBNAME statement to assign a libref to a SAS data library. General form of the LIBNAME statement:

```
LIBNAME libref 'SAS-data-library' <options>;
```

### Rules for naming a libref

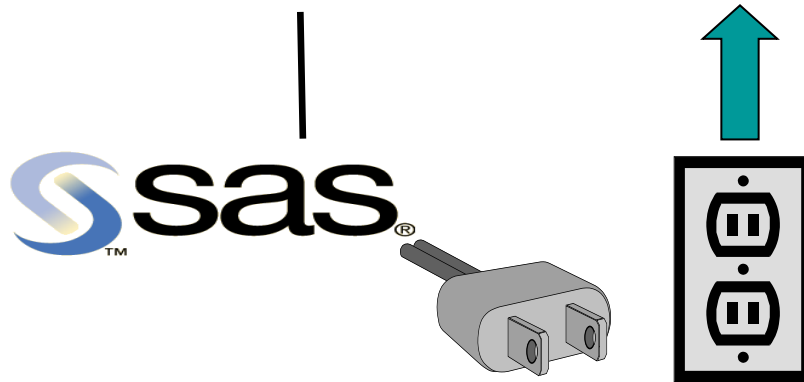
- must be 8 characters or less
- must begin with a letter or underscore
- remaining characters are letters, numbers, or underscores.

```
libname cpeng 'I:\cpeng\teaching\sta311\datasets';  
* I: is a network drive on the cloud (OneDrive);
```

## Making the Connection

When you submit the LIBNAME statement, a connection is made between a libref in SAS and the physical location of files on your operating system.

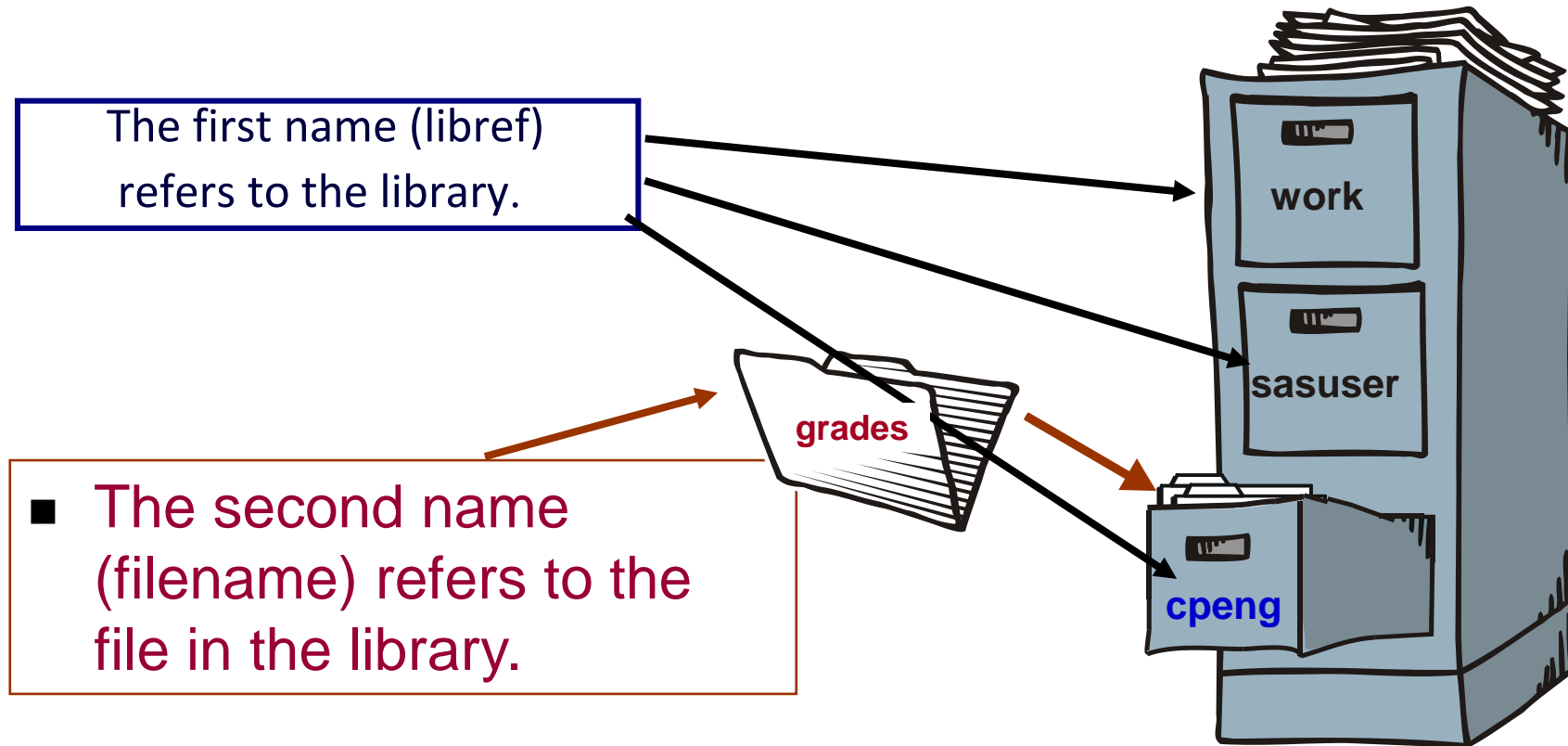
```
libname cpeng 'C:\STA311\Fall2020'; /* local drive */
```



## Assigning a LIBREF

Every SAS file has a two-level name: *libref.filename*

The data set **cpeng** . **grades** is a SAS file in the **cpeng** library.



# Temporary & Permanent SAS Filename

A *temporary* SAS data set is one that exists only for the current SAS session or job.

- The **Work** library is a temporary data library.
- Data sets held in the **Work** library are deleted at the end of the SAS session.

A *permanent* SAS data set is one that resides on the external storage medium of your computer and is not deleted when the SAS session terminates.

- Any data library referenced with a LIBNAME statement is considered a permanent data library by default.

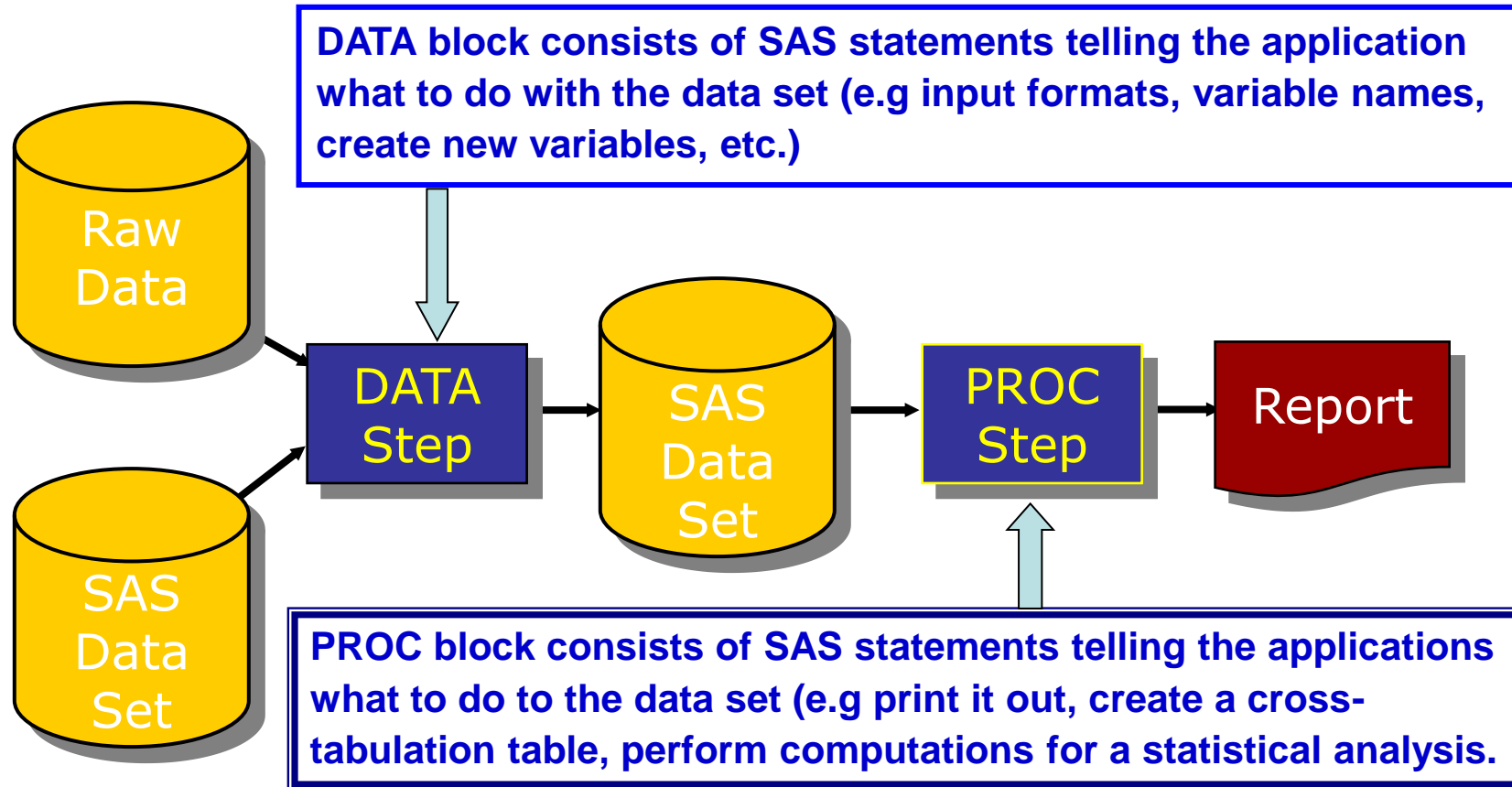
The libref **work** can be omitted when you refer to a file in the **work** library.  
The default libref is **work** if the libref is omitted.

**work.employee**



**employee**

# General Form of a SAS Program



## Some Naming Rules of SAS

1. Every SAS statement ends in a semicolon (;)!!!!!!!
2. All variable and data set names must start with a letter or an underscore (\_).
3. Names can contain only letters, numerals, and the underscore. No %\$!\*&#@.
4. SAS is case **insensitive** (e.g Body, body, boDY and BoDy are all the same name to SAS).
5. There are no restrictions on where in the line statements start or stop and commands may be wrapped onto multiple lines.
6. All variable and data set names must be thirty-two (32) or fewer characters in length (8 characters in SAS V6.12 or lower).

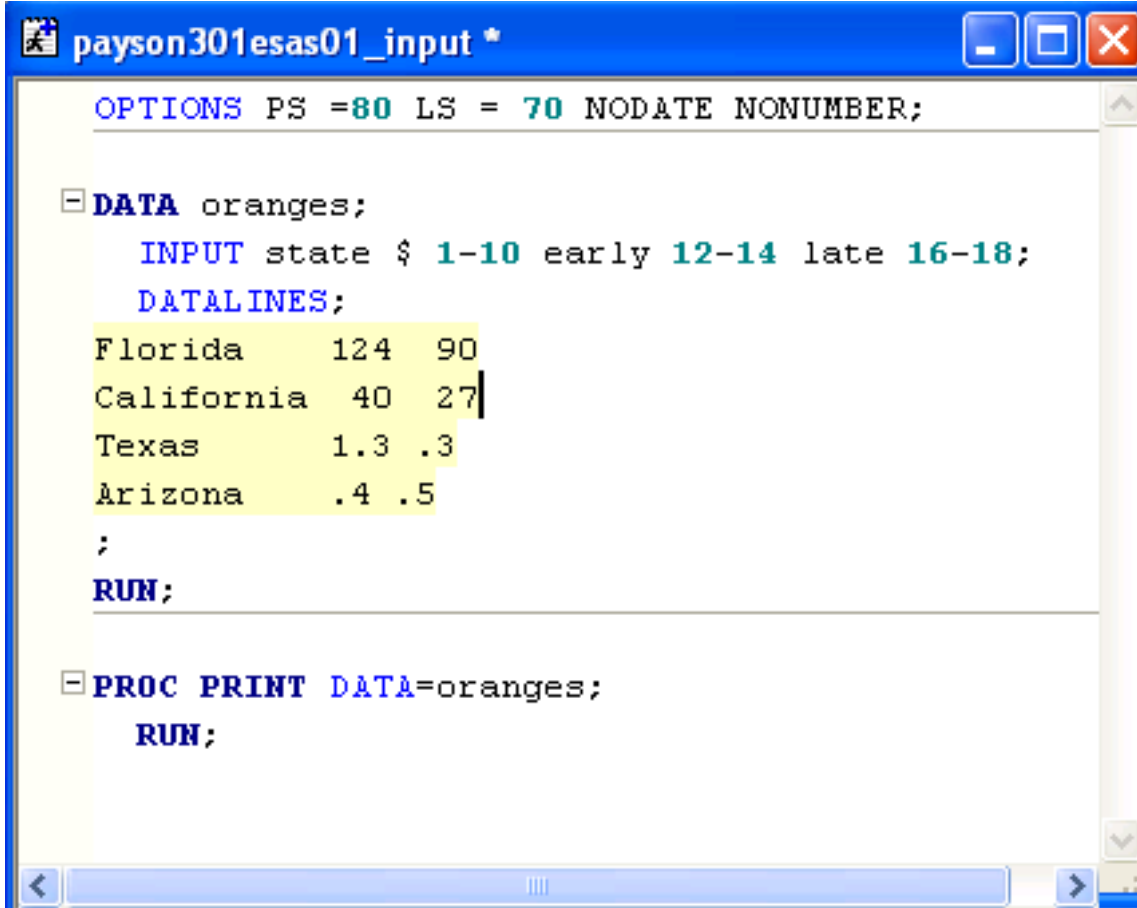
*For this class in order to make SAS programs easier to read, we will (require) attempt to have each new SAS statement start on a new line.*

# General Form of a SAS Program

**SAS programs (scripts) are composed of DATA block and PROCedure Blocks.**

- 1. DATA block consists of SAS statements telling the application what to do with the data set (e.g, input formats, variable names, create new variables, etc.)**
- 2. PROC block consists of SAS statements telling the applications what to do to the data set (e.g, print it out, create a cross-tabulation table, perform computations for a statistical analysis.**

# A Simple SAS Program



The screenshot shows a SAS editor window with the following code:

```
OPTIONS PS =80 LS = 70 NODATE NONUMBER;

DATA oranges;
  INPUT state $ 1-10 early 12-14 late 16-18;
  DATALINES;
Florida      124  90
California   40  27
Texas        1.3  .3
Arizona      .4  .5
;
RUN;

PROC PRINT DATA=oranges;
RUN;
```

Annotations on the left side of the image:

- DATA STEP**: A bracket groups the `DATA oranges;` through `RUN;` lines.
- PROC STEP**: A bracket groups the `PROC PRINT DATA=oranges;` through `RUN;` lines.



Editor - Untitled1 \*

```
DM 'CLEAR LOG';
DM 'CLEAR OUTPUT';

/* column input */
DATA Orange;
INPUT state $ 1-10 early 12-14 late 16-18;
DATALINES;
Florida 130 90
California 37 26
Texas 1.3 .15
Arizona .65 .85
;
RUN;

PROC PRINT DATA = Orange;
TITLE "Output of theTesting Data";
RUN;
```

DATA Step

PROC Step

Output - (Untitled)

Output of theTesting Data

09:35 Monday, September 10, 2012

Obs	state	early	late
1	Florida	130.00	90.00
2	California	37.00	26.00
3	Texas	1.30	0.15
4	Arizona	0.65	0.85

Output

Log - (Untitled)

```
20
21 DM 'CLEAR LOG';
22 DM 'CLEAR OUTPUT';
23
24 /* column input */
25 DATA Orange;
26 INPUT state $ 1-10 early 12-14 late 16-18;
27 DATALINES;
```

NOTE: The data set WORK.ORANGE has 4 observations and 3 variables.  
NOTE: DATA statement used (Total process time):  
real time 0.01 seconds  
cpu time 0.01 seconds

```
32 ;
33 RUN;
34
35 PROC PRINT DATA = Orange;
36 TITLE "Output of theTesting Data";
37 RUN;
```

NOTE: There were 4 observations read from the data set WORK.ORANGE.  
NOTE: PROCEDURE PRINT used (Total process time):  
real time 0.50 seconds  
cpu time 0.09 seconds

Log File

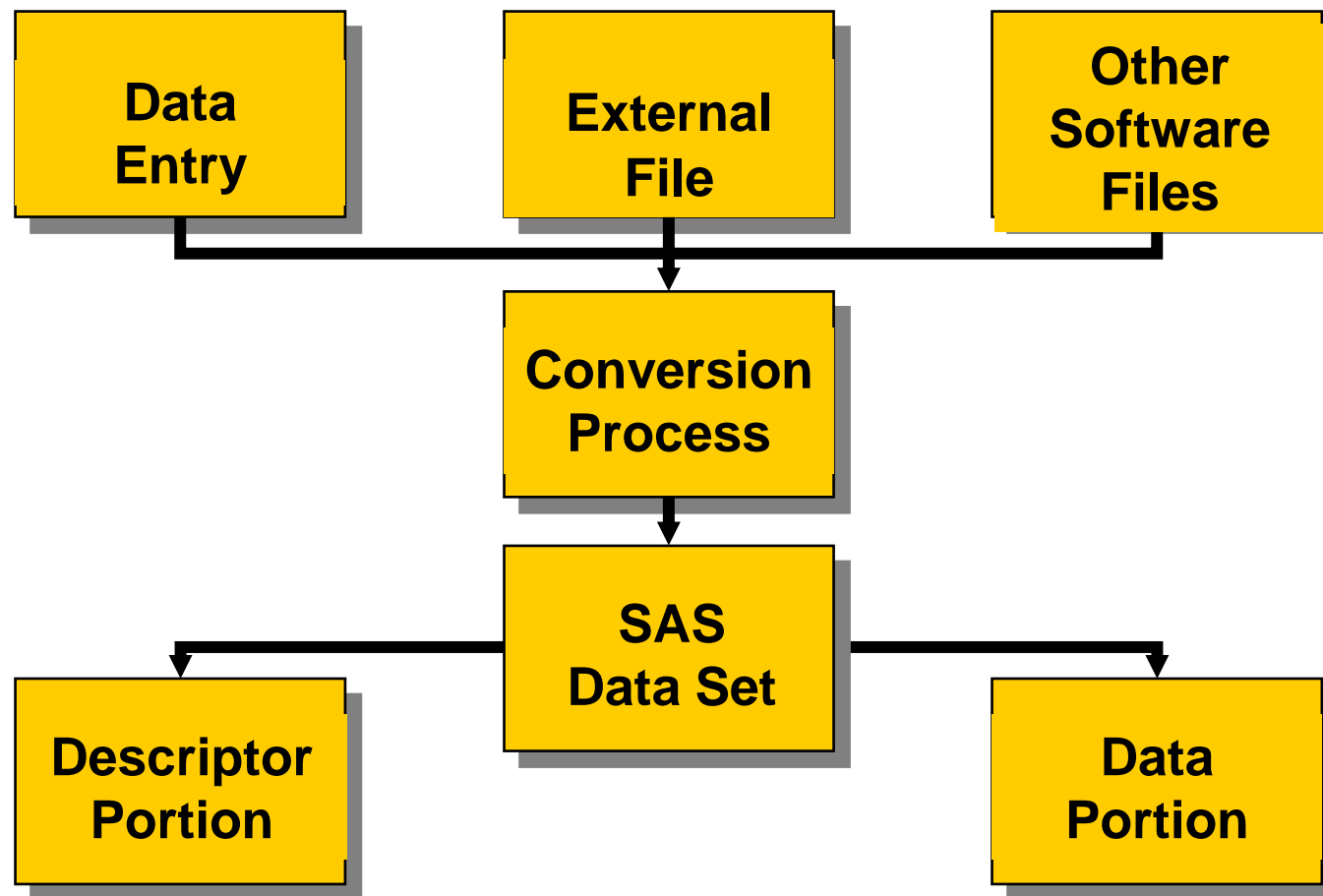
# SAS Log

The SAS log is a record of your submitted SAS program.

```
125 libname project 'C:\workshop\winsas\lwcrb\data';  
NOTE: Libref PROJECT was successfully assigned as follows:  
      Engine:          V9  
      Physical Name: C:\workshop\winsas\lwcrb\data  
  
126 proc sort data=work.enroll  
127           out=project.enroll;  
128     by last;  
129 run;  
  
NOTE: There were 4 observations read from the data set WORK.ENROLL.  
NOTE: The data set PROJECT.ENROLL has 4 observations and 5 variables.
```

- Original program statements are identified by line numbers.
- SAS messages can include the words NOTE, INFO, WARNING, or ERROR.

# Creating SAS Datasets: Process and Structure



# SAS Datasets

A SAS data set has these characteristics:

- is a SAS file stored in a SAS library that SAS creates and processes
- contains data values that are organized as a table of observations (rows) and variables (columns)
- contains descriptor information such as the data types and lengths of the variables

# SAS Datasets

SAS data sets have a descriptor portion and a data portion.

General data set information

\* data set name            \* data set label  
\* date/time created   \* storage information  
\* number of observations

**Descriptor  
Portion**

Information for each variable

\* Name    \* Type            \* Length   \* Position  
\* Format \* Informat   \* Label

**Data  
Portion**

# SAS Statements

A SAS statement is a series of items that might include keywords, SAS names, special characters, and operators.

The two types of SAS statements are as follows:

- those that are used in DATA and PROC steps
- those that are global in scope and can be used anywhere in a SAS program

All SAS statements end with a semicolon.

Global statements

- are used anywhere in a SAS program
- stay in effect until changed or canceled, or until you end your SAS session.

## Browsing the Descriptor Portion



The descriptor portion of a SAS data set contains

- general information about the SAS data set (such as data set name and number of observations)
- variable attributes (name, type, length, position, informat, format, label).

The CONTENTS procedure displays the descriptor portion of a SAS data set.

# Comments

There are two ways to add comments in a SAS program.

```
 *The SORT procedure is creating a data set;  
proc sort data=work.enroll  
          out=project.enroll;  
  by last;  
run;  
  
 /* The PRINT procedure is creating a report. */  
proc print data=project.enroll;  
  var last state2 _age_ enrolldate;  
  format enrolldate mmddyy10.;  
run;
```

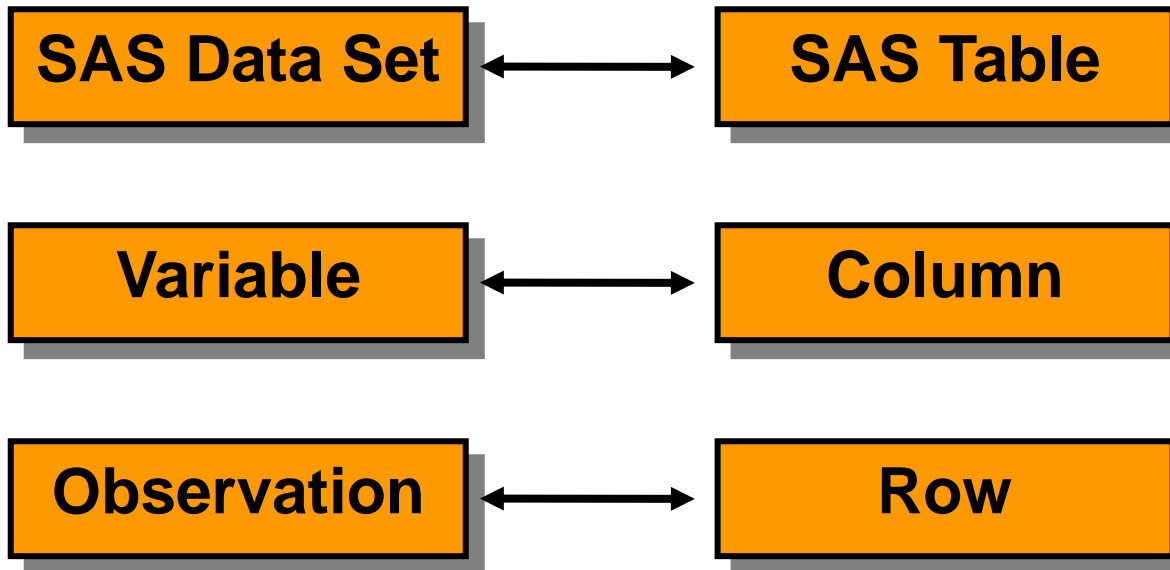
During processing, SAS ignores text in comments.

**Comments make your code more readable!**



# SAS Dataset Terminology

SAS documentation and text in the SAS windowing environment use the following terms interchangeably:



# SAS Variables & Variable Naming Rules

Data values are organized into columns called *variables*.

Variables have attributes, such as name and type, that enable you to identify them and that define how they can be used.

By default, a SAS variable name

- can be up to 32 characters long
- must begin with a letter (A-Z) or an underscore (\_)
- can contain only letters, digits (0-9), or underscores.

The names `_N_`, `_ERROR_`, `_FILE_`, `_INFILE_`, `_MSG_`, `_IORC_`, and `_CMD_` are reserved for the variables that are generated automatically for a DATA step. Note that SAS products use variable names that start and end with an underscore; it is recommended that you do not use names that start and end with an underscore in your own applications

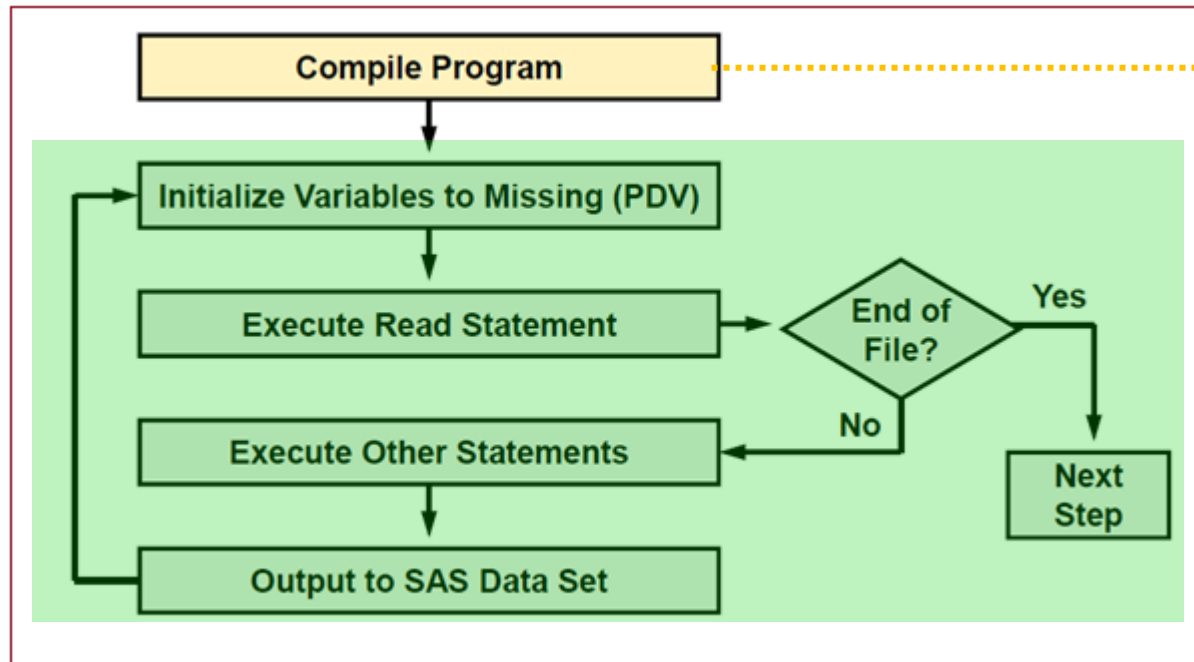
**An effective programmer always uses meaningful names!**

# Formats and Informats of SAS Variables

- **Informats** is used to tell SAS how to **read a variable** whereas **Formats** is used to tell SAS how to **display or write values** of a variable.
- **Informats** is basically used when you read or import data from either an external file (Text/Excel/CSV) or read in sample data which was created using CARDS/DATALINES statement. It is also used when you create a new variable in a dataset.
- **Formats** can be used in both Data Steps and PROC Steps *whereas Informat can be used only in Data Steps.*

**We will learn how to define formats/informats later.**

# Data Step Execution Process



## Compile Phase Creates

- An input buffer
- A program data vector(PDV)
- Descriptor information

## Execution Phase

- Data step reads data records interactively. Automatic counter `_N_` updates its value
- SAS sets the newly created program variable to missing in the program data vector(PDV).
- Variables that you read with a SET, MERGE, MODIFY, or UPDATE statement are **not** reset to missing in PDV.
- SAS reads a data record from a raw data file into the input buffer, or it reads an observation from a SAS data set directly into the PDV
- The DATA step terminates when SAS encounters the end-of-file in a SAS data set or a raw data file

We will do an example data step using INPUT statement to illustrate the detailed process of executing data step and how SAS data set.

## Some Best Coding Practices for This Class

The following are some best practices you should always have in mind when you make a program

1. Write a clear program header;
2. Write comments and documentation;
3. Use consistent indentation;
4. Don't repeat yourself (aka DRY principle);
5. Avoid deep nesting;
6. Limit line length;
7. Set up a simple and clear file and folder structure;
8. Follow naming convention;
9. **Keep the code simple!**
10. Save the code a version number – version control;
11. **Never modify the source data!!!**