

Chapter 1: Introduction



1.1 Course Logistics

1.2 An Overview of Foundation SAS

Chapter 1: Introduction



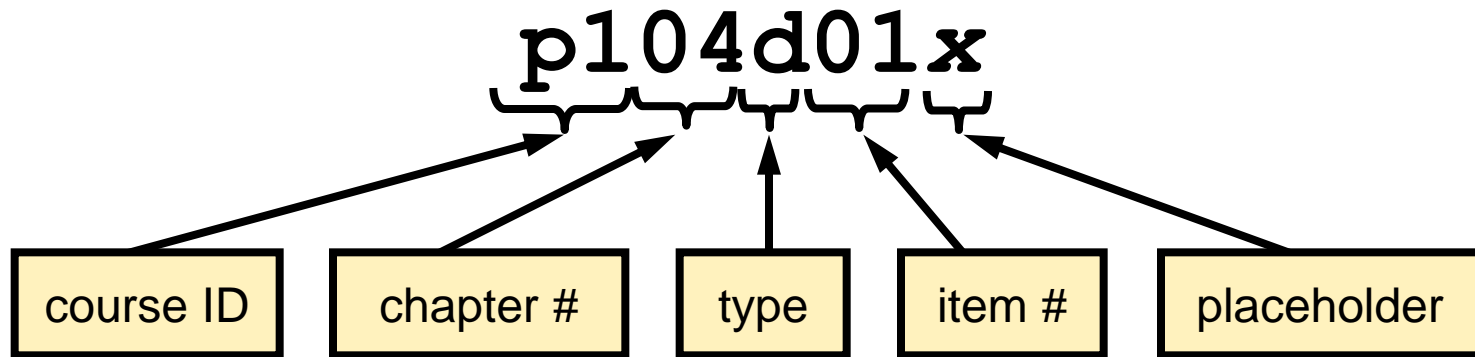
1.1 Course Logistics

1.2 An Overview of Foundation SAS

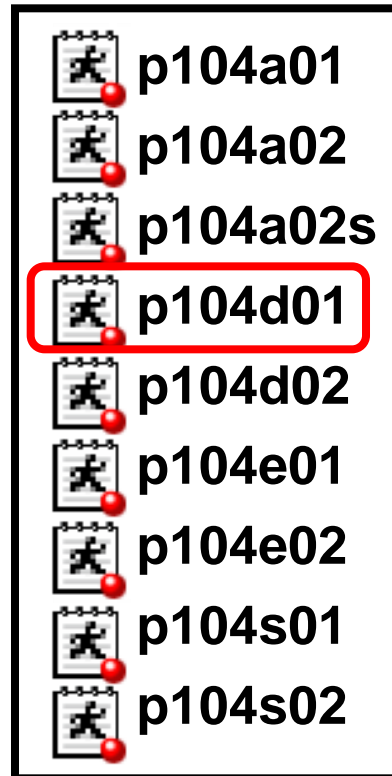
Objectives

- Explain the naming convention that is used for the course files.
- Compare the three levels of exercises that are used in the course.
- Describe at a high level how data is used and stored at Orion Star Sports & Outdoors.
- Navigate to the Help facility.

Filename Conventions



Code	Type
a	Activity
d	Demo
e	Exercise
s	Solution



Example:
The Programming 1
course ID is p1, so
p104d01 =
Programming 1
Chapter 4, Demo 1.

SAS Data & Sample Programs

Available for download:

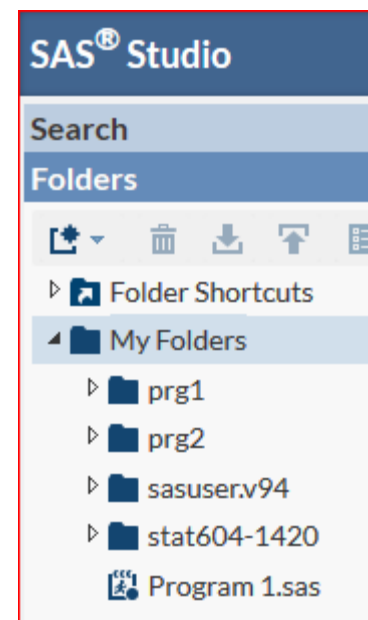
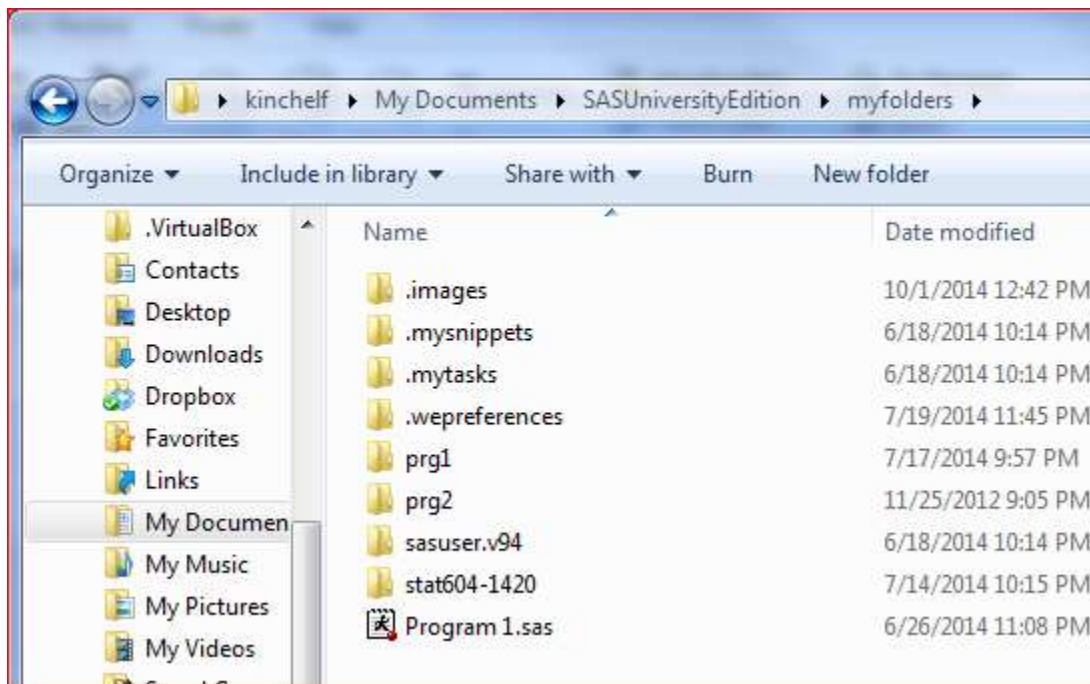
The screenshot displays a course management system interface. At the top, a breadcrumb trail shows 'Course Material > Prog 1 SAS Notes'. The sidebar on the left contains a list of navigation links: 'Home Page', 'Announcements', 'General Information', 'Syllabus & Policies', 'Lecture Notes', 'Class Videos', 'Online Sessions', 'Discussions', and 'Course Material'. A red arrow points to the 'Course Material' link. The main content area is titled 'Prog 1 SAS Notes' with a red arrow pointing to a dropdown arrow next to the title. Below the title, there are four tabs: 'Build Content', 'Assessments', 'Tools', and 'Partner Content'. The 'Assessments' tab is selected. Under this tab, there are two items listed: 'SAS Programming 1 Course Notes' and 'Prog 1 Sample Data and Programs (zip archive)', both with document icons.

SAS Data & Sample Programs

Making available to the SAS Studio Editor

Computer Folder

SAS Studio



eCampus – Course Material – Lecture and Reading Schedule

29-Sep	SAS01	Prog1 Chapters 1-3
6-Oct	SAS02	Prog1 Sections 4.1-4.3 and Section 11.5
11-Oct	SAS03	Prog1 Sections 5.1-5.3
13-Oct	SAS04	Prog1 Sections 5.4 and 9.1
18-Oct	SAS05	Prog1 Sections 9.2-9.3 and Prog2 Sections 2.1-2.2
20-Oct	SAS06	Prog2 Section 2.3 and Sections 5.1-5.2

- Lecture slides will be very similar to those shown in course notes but may include “enhancements” not in the course notes.

Note: Unless otherwise noted, Self-Study material will be covered in STAT 604.

Three Levels of Exercises

Level 1

The exercise mimics an example presented in the section.

Level 2

Less information and guidance are provided in the exercise instructions.

Level 3

Only the task you are to perform or the results to be obtained are provided. Typically, you will need to use the Help facility.



It is recommended that you complete as many of these exercises as your personal schedule will allow.

Orion Star Sports & Outdoors



Orion Star Sports & Outdoors is a fictitious global sports and outdoors retailer with traditional stores, an online store, and a large catalog business.

The corporate headquarters is located in the United States with offices and stores in many countries throughout the world.

Orion Star has about 1,000 employees and 90,000 customers, processes approximately 150,000 orders annually, and purchases products from 64 suppliers.

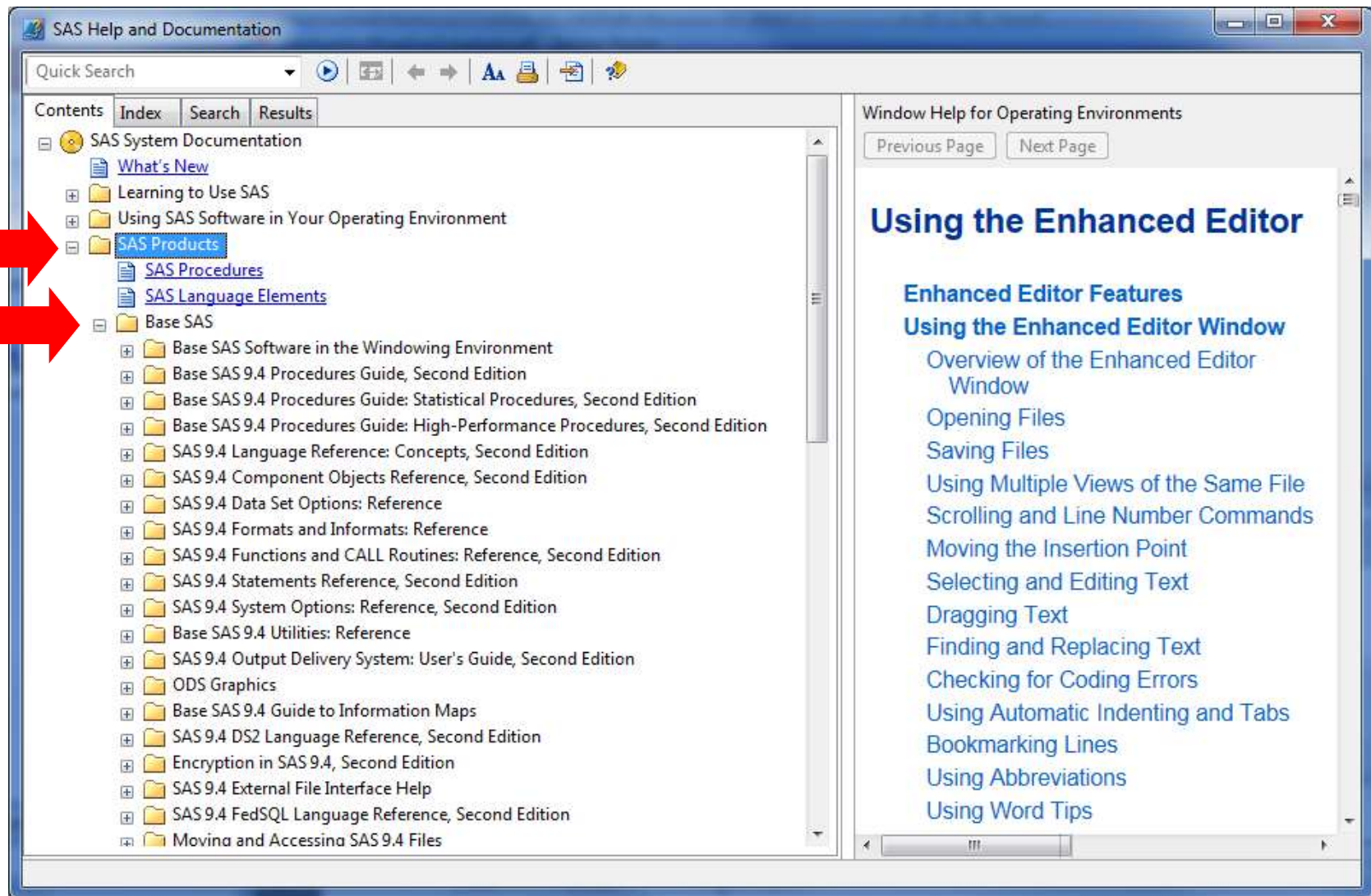
Orion Star Data

As is the case with most organizations, Orion Star has a large amount of data about its customers, suppliers, products, and employees. Much of this information is stored in transactional systems in various formats.

Using applications and processes such as SAS Data Integration Studio, this transactional information was extracted, transformed, and loaded into a data warehouse.

Data marts were created to meet the needs of specific departments such as Marketing.

The SAS Help Facility



Chapter 1: Introduction



1.1 Course Logistics



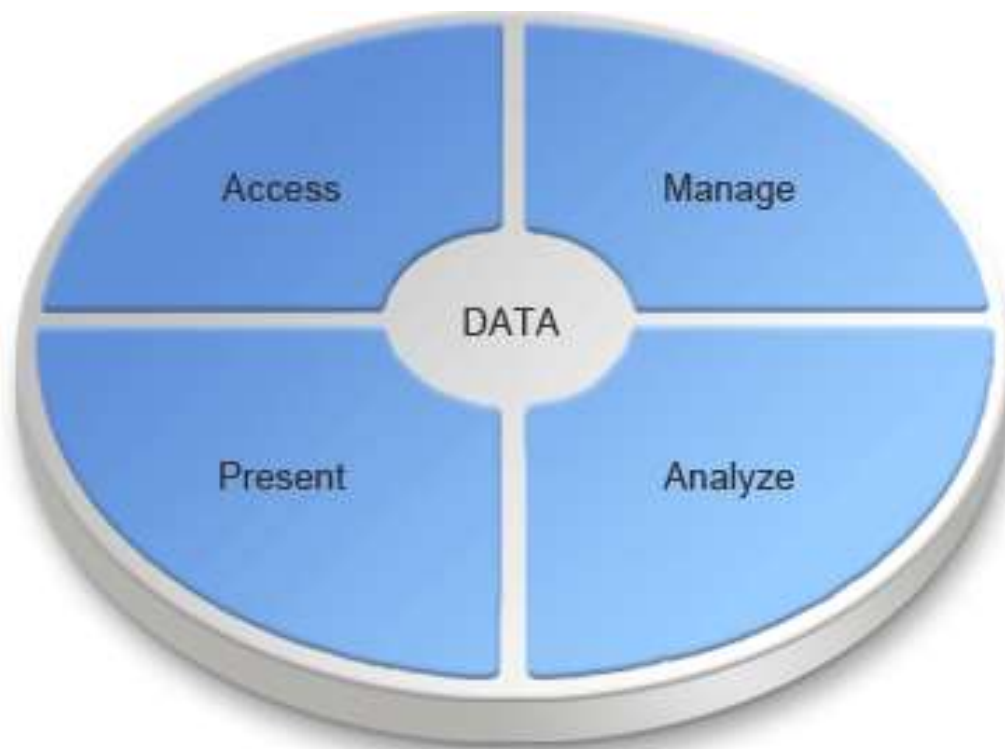
1.2 An Overview of Foundation SAS

Objectives

- Describe the structure and design of Foundation SAS.
- Describe the functionality of Foundation SAS.

What Is Foundation SAS?

Foundation SAS is a highly flexible and integrated software environment that can be used in virtually any setting to access, manipulate, manage, store, analyze, and report on data.



What Is Foundation SAS?

Foundation SAS provides the following:

- a graphical user interface for administering SAS tasks
- a highly flexible and extensible programming language
- a rich library of prewritten, ready-to-use SAS procedures
- the flexibility to run on all major operating environments such as Windows, UNIX, and z/OS (OS/390)
- the access to virtually any data source such as DB2, Oracle, SYBASE, Teradata, SAP, and Microsoft Excel
- the support for most widely used character encodings for globalization

What Is Foundation SAS?

At the core of Foundation SAS is Base SAS software.

Components of Foundation SAS

Reporting and Graphics	Data Access and Management	User Interfaces
Analytics	Base SAS	Application Development
Visualization and Discovery	Business Solutions	Web Enablement

Base SAS capabilities can be extended with additional components.

Chapter 2: Getting Started with SAS



2.1 Introduction to SAS Programs

2.2 Submitting a SAS Program

Chapter 2: Getting Started with SAS

2.1 Introduction to SAS Programs

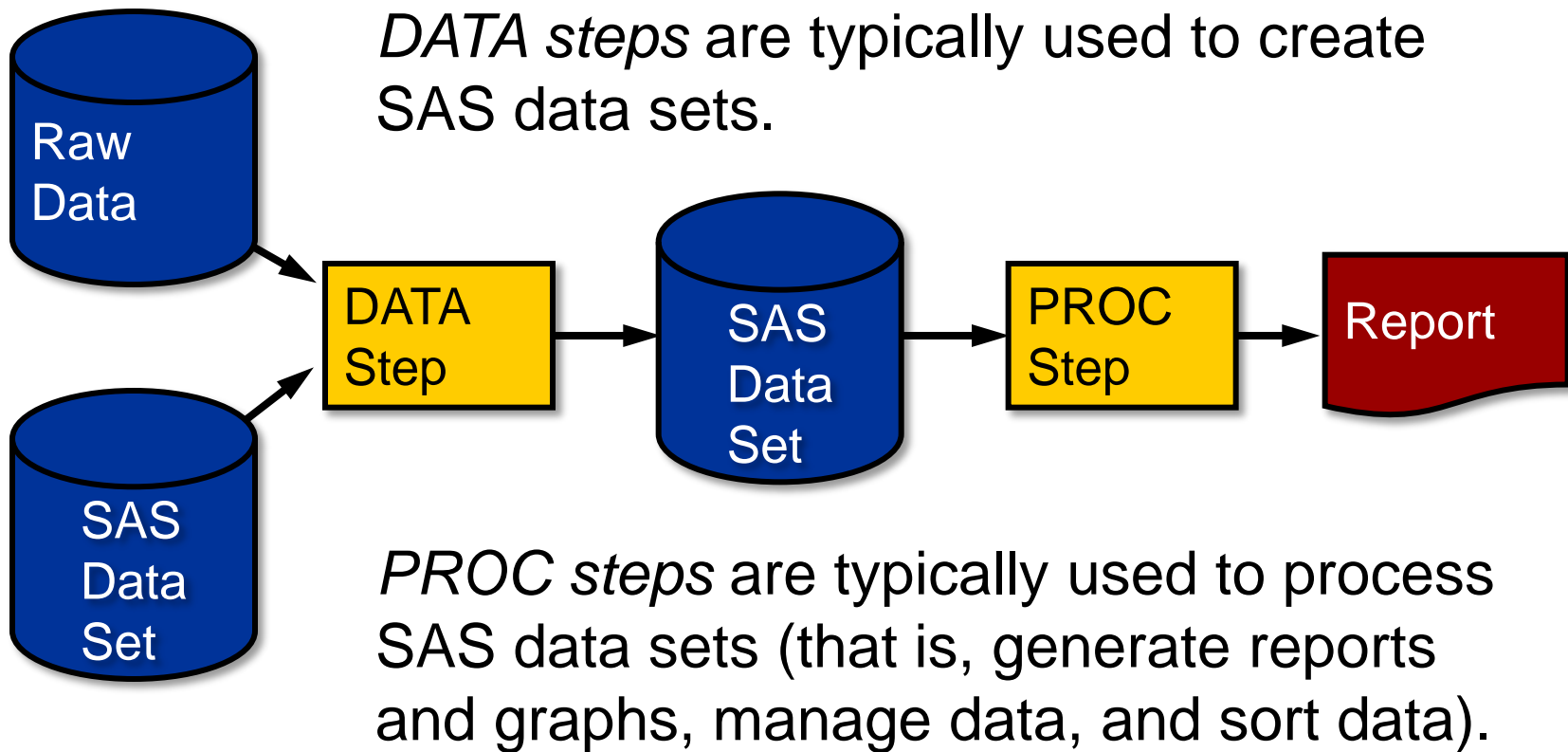
2.2 Submitting a SAS Program

Objectives

- List the components of a SAS program.
- State the modes in which you can run a SAS program.

SAS Programs

A *SAS program* is a sequence of steps that the user submits for execution.



Poll

Quiz



2.01 Quiz

How many steps are in this program?

```
data work.NewSalesEmps;  
    length First Name $ 12  
           Last Name $ 18 Job Title $ 25;  
    infile 'newemps.csv' dlm=' ', '  
    input First Name $ Last Name $  
           Job Title $ Salary;  
run;  
  
proc print data=work.NewSalesEmps;  
run;  
  
proc means data=work.NewSalesEmps;  
    class Job Title;  
    var Salary;  
run;
```

2.01 Quiz – Correct Answer

How many steps are in this program?

```
data work.NewSalesEmps;  
  length First Name $ 12  
         Last_Name $ 18 Job_Title $ 25;  
  infile 'newemps.csv' dlm=' ', '  
  input First_Name $ Last_Name $  
        Job_Title $ Salary;  
run;  
  
proc print data=work.NewSalesEmps;  
run;  
  
proc means data=work.NewSalesEmps;  
  class Job_Title;  
  var Salary;  
run;
```

DATA
Step

PROC
Step

PROC
Step

3 steps

SAS Program Example

This DATA step creates a temporary SAS data set named **Work.NewSalesEmps** by reading four fields from a raw data file.

```
data work.NewSalesEmps;  
    length First Name $ 12  
           Last Name $ 18 Job Title $ 25;  
    infile 'newemps.csv' dlm=',';  
    input First Name $ Last Name $  
          Job Title $ Salary;  
run;  
  
proc print data=work.NewSalesEmps;  
run;  
  
proc means data=work.NewSalesEmps;  
    class Job Title;  
    var Salary;  
run;
```


SAS Program Example

This PROC PRINT step creates a listing report of the **Work.NewSalesEmps** data set.

```
data work.NewSalesEmps;  
    length First Name $ 12  
           Last Name $ 18 Job Title $ 25;  
    infile 'newemps.csv' dlm=',';  
    input First Name $ Last Name $  
          Job Title $ Salary;  
run;  
  
proc print data=work.NewSalesEmps;  
run;  
  
proc means data=work.NewSalesEmps;  
    class Job Title;  
    var Salary;  
run;
```

SAS Program Example

This PROC MEANS step creates a summary report of the **Work.NewSalesEmps** data set with statistics for the variable **Salary** for each value of **Job_Title**.

```
data work.NewSalesEmps;  
    length First Name $ 12  
           Last Name $ 18 Job Title $ 25;  
    infile 'newemps.csv' dlm=',';  
    input First Name $ Last Name $  
          Job Title $ Salary;  
run;  
  
proc print data=work.NewSalesEmps;  
run;  
  
proc means data=work.NewSalesEmps;  
    class Job Title;  
    var Salary;  
run;
```

Step Boundaries

SAS steps begin with either of the following:

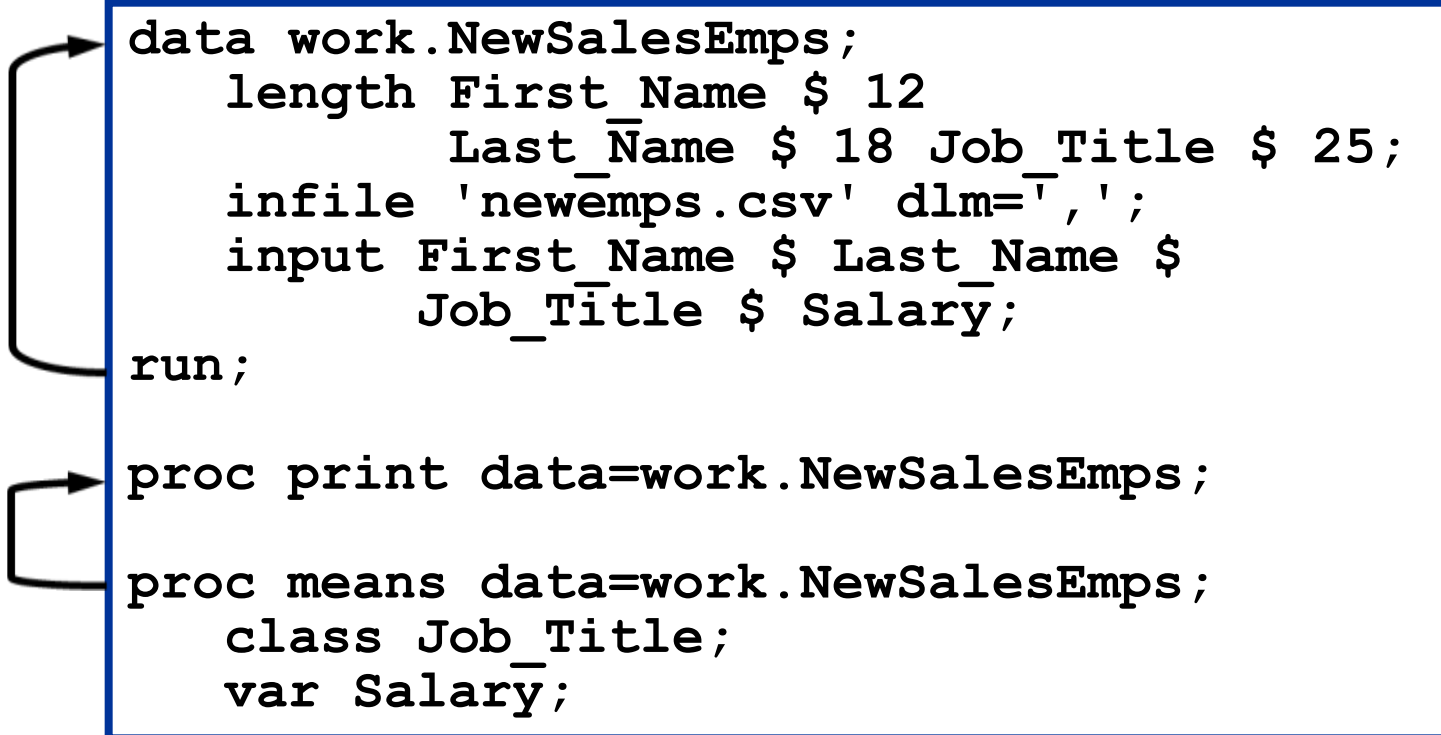
- a DATA statement
- a PROC statement

SAS detects the end of a step when it encounters one of the following:

- a RUN statement (for most steps)
- a QUIT statement (for some procedures)
- the beginning of another step (DATA statement or PROC statement)

Step Boundaries

SAS detects the end of the DATA step when it encounters the RUN statement.



```
data work.NewSalesEmps;  
    length First_Name $ 12  
           Last_Name $ 18 Job_Title $ 25;  
    infile 'newemps.csv' dlm='|';  
    input First_Name $ Last_Name $  
          Job_Title $ Salary;  
run;  
  
proc print data=work.NewSalesEmps;  
  
proc means data=work.NewSalesEmps;  
    class Job_Title;  
    var Salary;
```

SAS detects the end of the PROC PRINT step when it encounters the beginning of the PROC MEANS step.


Poll

Quiz



2.02 Quiz

How does SAS detect the end of the PROC MEANS step?




```
data work.NewSalesEmps;  
    length First Name $ 12  
           Last_Name $ 18 Job_Title $ 25;  
    infile 'newemps.csv' dlm='|';  
    input First Name $ Last_Name $  
          Job_Title $ Salary;  
run;  
  
proc print data=work.NewSalesEmps;  
  
proc means data=work.NewSalesEmps;  
    class Job_Title;  
    var Salary;
```

The diagram illustrates the execution flow of the SAS code. A large bracket on the left groups the first five lines of code (the DATA step), with an arrow pointing to the first line. A second bracket on the left groups the next three lines of code (the PROC PRINT and PROC MEANS steps), with an arrow pointing to the first line of the PROC PRINT step.

2.02 Quiz – Correct Answer

How does SAS detect the end of the PROC MEANS step?

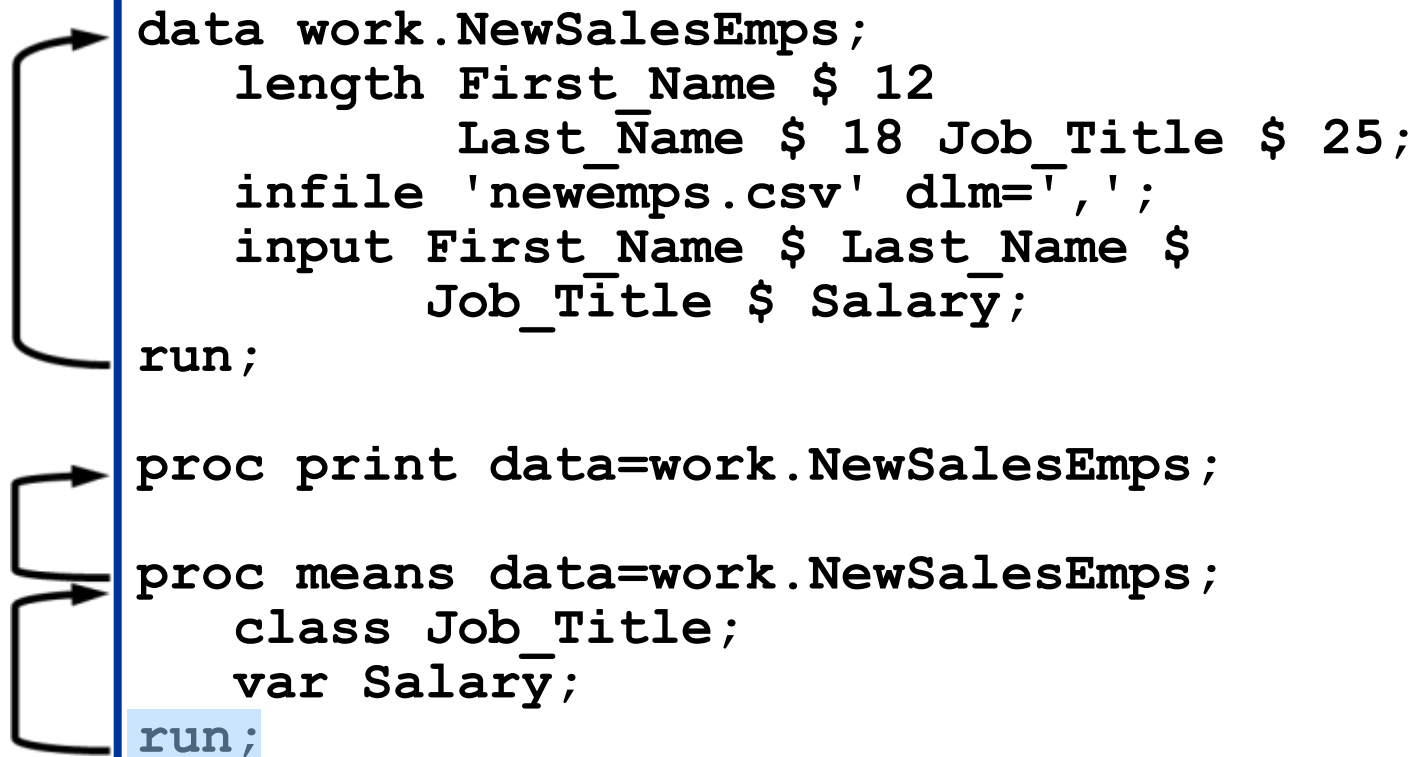


```
data work.NewSalesEmps;  
    length First Name $ 12  
           Last Name $ 18 Job Title $ 25;  
    infile 'newemps.csv' dlm='|';  
    input First Name $ Last Name $  
          Job Title $ Salary;  
run;  
  
proc print data=work.NewSalesEmps;  
proc means data=work.NewSalesEmps;  
    class Job Title;  
    var Salary;
```

**SAS does not detect the end of the PROC MEANS step.
SAS needs a RUN statement to detect the end.**

Step Boundaries

SAS detects the end of the PROC MEANS step when it encounters the RUN statement.



```
data work.NewSalesEmps;  
    length First_Name $ 12  
           Last_Name $ 18 Job_Title $ 25;  
    infile 'newemps.csv' dlm='|';  
    input First_Name $ Last_Name $  
          Job_Title $ Salary;  
run;  
  
proc print data=work.NewSalesEmps;  
  
proc means data=work.NewSalesEmps;  
    class Job_Title;  
    var Salary;  
run;
```

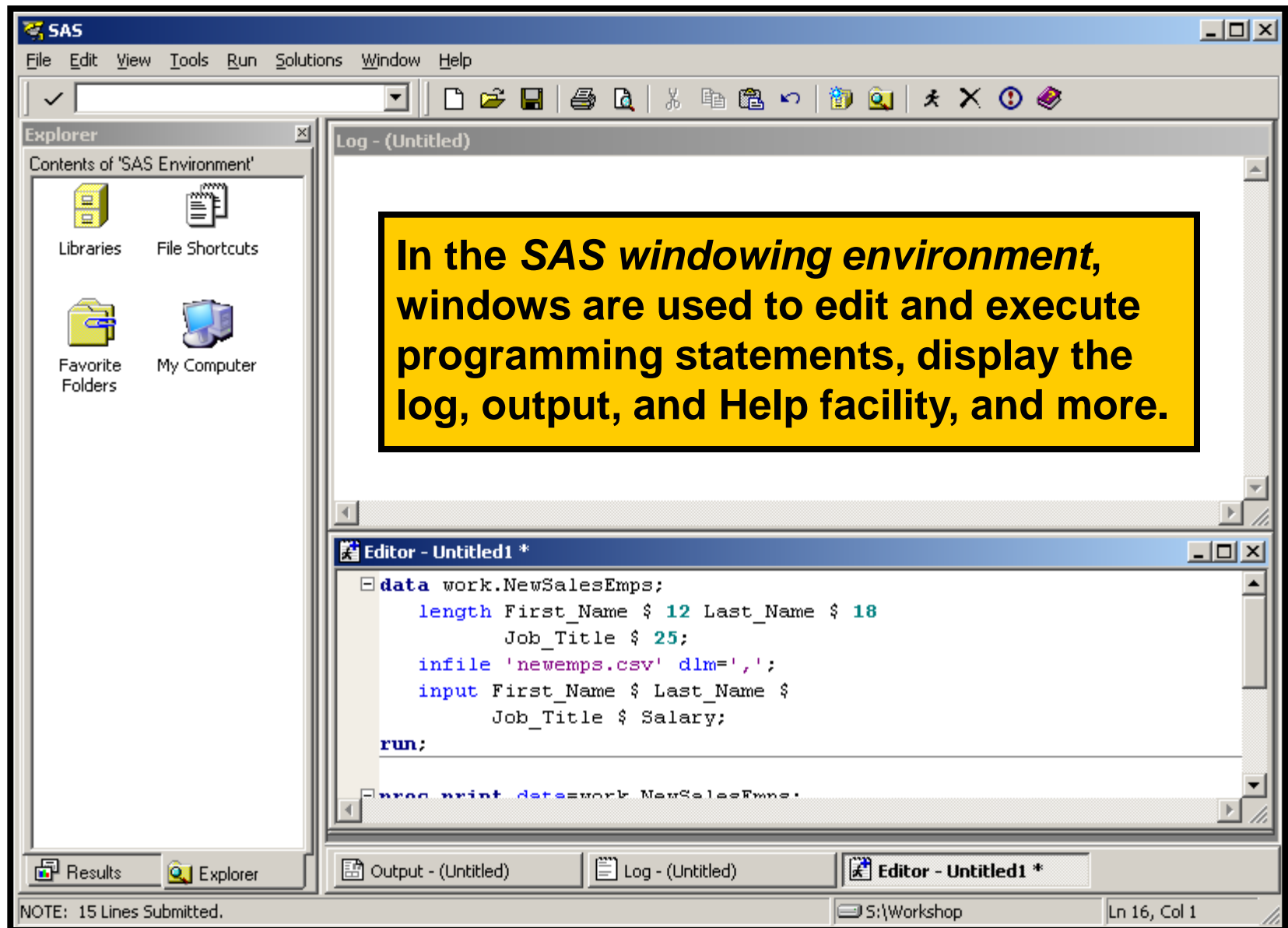
The diagram illustrates step boundaries in SAS code. It shows three distinct steps: a DATA step, a PROC PRINT step, and a PROC MEANS step. Each step is terminated by a 'run;' statement. Arrows on the left side of the code block point to each 'run;' statement, indicating the end of a step. The final 'run;' statement is highlighted with a light blue background.

Running a SAS Program

You can invoke SAS in the following ways:

- interactive mode (for example, SAS windowing environment and SAS Enterprise Guide)
- batch mode
- noninteractive mode

SAS Windowing Environment



Batch Mode

Batch mode is a method of running SAS programs in which you prepare a file that contains SAS statements plus any necessary operating system control statements and submit the file to the operating system.

Partial z/OS (OS/390) Example:

```
//jobname JOB accounting info,name ...  
// EXEC SAS  
//SYSIN DD *
```

**Appropriate JCL
is placed before
SAS statements.**

```
data work.NewSalesEmps;  
  length First_Name $ 12  
         Last_Name $ 18 Job_Title $ 25;  
  infile '.workshop.rawdata(newemps)' dlm=',';  
  input First_Name $ Last_Name $  
        Job_Title $ Salary;  
run;
```

Noninteractive Mode

In *noninteractive mode*, SAS program statements are stored in an external file and are executed immediately after you issue a SAS command referencing the file.

Directory-based Example:

SAS *filename*

z/OS (OS/390) Example:

SAS INPUT(*filename*)

Chapter 2: Getting Started with SAS



2.1 Introduction to SAS Programs

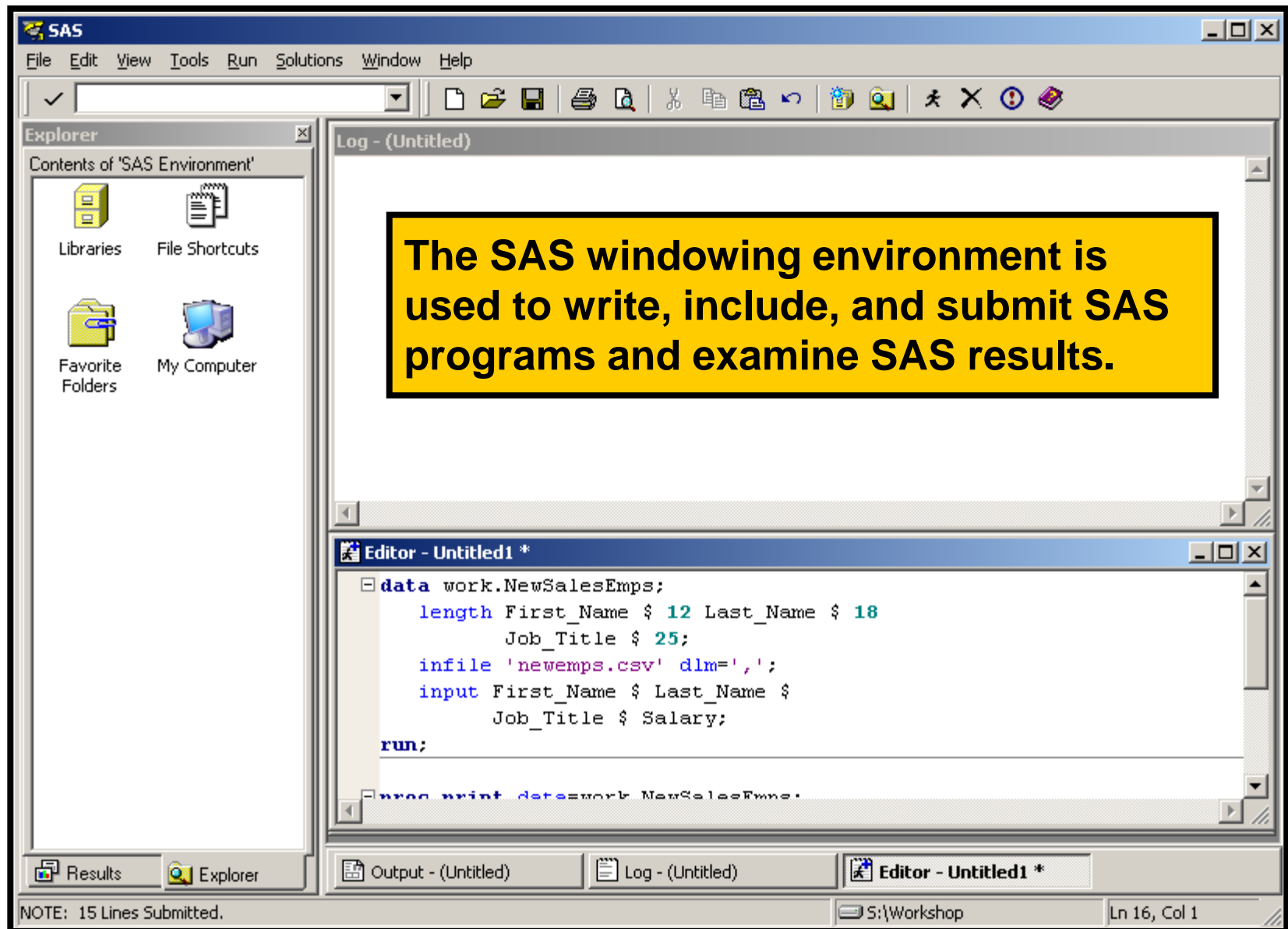


2.2 Submitting a SAS Program

Objectives





- Include a SAS program in your session.
- Submit a program and browse the results.
- Navigate the SAS windowing environment.

SAS Windowing Environment





Three Primary Windows

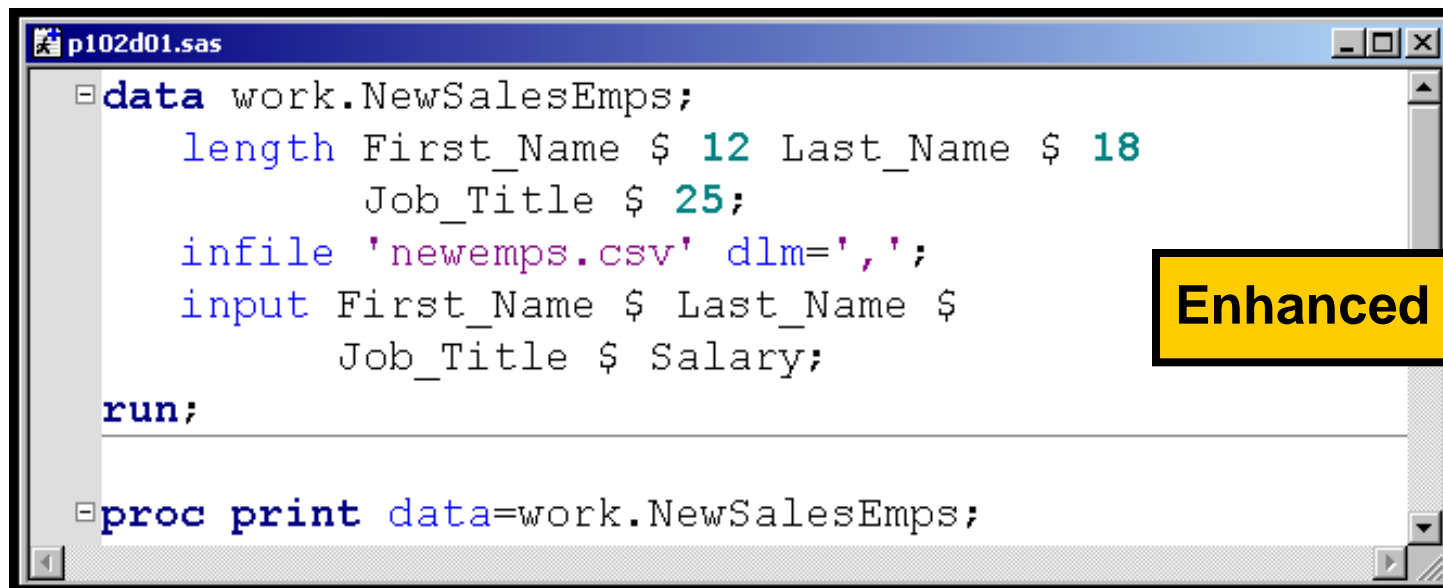
In the SAS windowing environment, you submit and view the results of a SAS program using three primary windows.

 Editor - Untitled1  Program Editor - (Untitled)	contains the SAS program to submit.
 Log - (Untitled)	contains information about the processing of the SAS program, including any warning and error messages.
 Output - (Untitled)	contains reports generated by the SAS program.

Editor Windows

Enhanced Editor	Program Editor
 Editor - Untitled1	 Program Editor - (Untitled)
Only available in the Windows operating environment	Available in all operating environments
Default editor for Windows operating environment	Default editor for all operating environments except Windows
Multiple instances of the editor can be open at one time	Only one instance of the editor can be open at one time
Code does not disappear after it is submitted	Code disappears after it is submitted
Incorporates color-coding as you type	Incorporates color-coding after you press ENTER

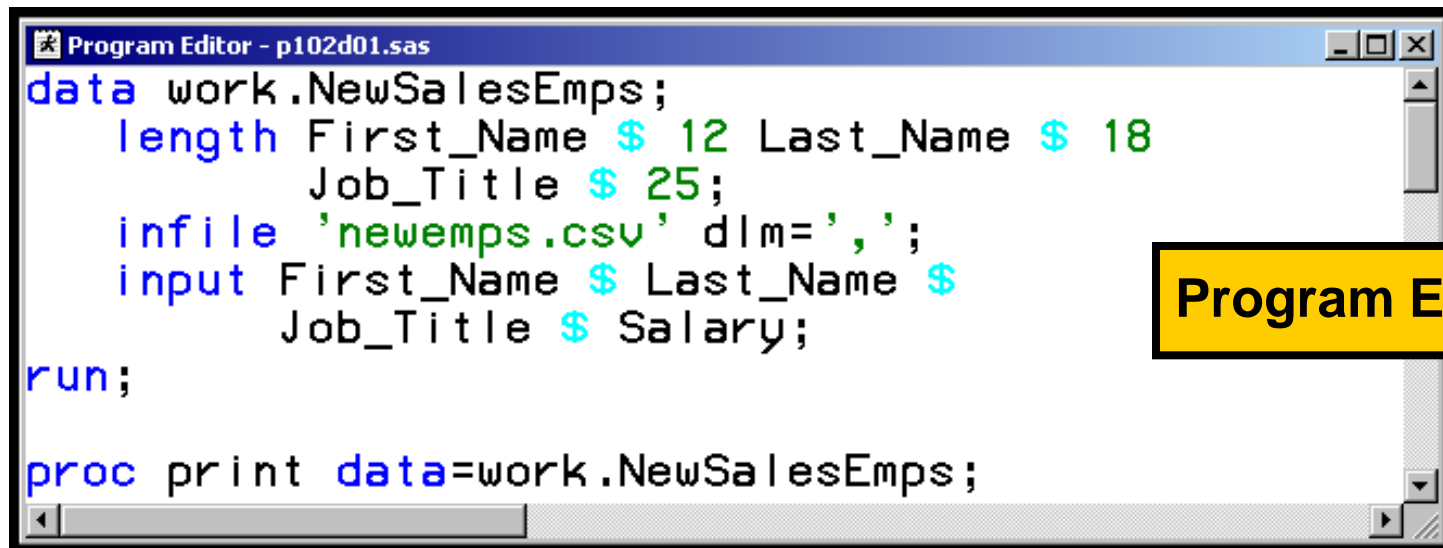
Editor Windows



The Enhanced Editor window displays SAS code with syntax highlighting. The code defines a dataset 'work.NewSalesEmps' with variables 'First_Name', 'Last_Name', 'Job_Title', and 'Salary'. It reads data from 'newemps.csv' and prints the results.

```
p102d01.sas  
data work.NewSalesEmps;  
    length First_Name $ 12 Last_Name $ 18  
           Job_Title $ 25;  
    infile 'newemps.csv' dlm=',';  
    input First_Name $ Last_Name $  
          Job_Title $ Salary;  
  
run;  
  
proc print data=work.NewSalesEmps;
```

Enhanced Editor



The Program Editor window displays the same SAS code as the Enhanced Editor, but without syntax highlighting. The code defines a dataset 'work.NewSalesEmps' with variables 'First_Name', 'Last_Name', 'Job_Title', and 'Salary'. It reads data from 'newemps.csv' and prints the results.

```
Program Editor - p102d01.sas  
data work.NewSalesEmps;  
    length First_Name $ 12 Last_Name $ 18  
           Job_Title $ 25;  
    infile 'newemps.csv' dlm=',';  
    input First_Name $ Last_Name $  
          Job_Title $ Salary;  
  
run;  
  
proc print data=work.NewSalesEmps;
```

Program Editor

Log Window

Partial SAS Log

```
33  data work.NewSalesEmps;  
34      length First_Name $ 12 Last_Name $ 18  
35          Job_Title $ 25;  
36      infile 'newemps.csv' dlm=',';  
37      input First_Name $ Last_Name $  
38          Job_Title $ Salary;  
39  run;
```

NOTE: The infile 'newemps.csv' is:
File Name=S:\Workshop\newemps.csv,
RECFM=V,LRECL=256

NOTE: 71 records were read from the infile 'newemps.csv'.
The minimum record length was 28.
The maximum record length was 47.

NOTE: The data set WORK.NEWSALESEMPs has 71 observations and 4 variables.

```
40  
41  proc print data=work.NewSalesEmps;  
42  run;
```

NOTE: There were 71 observations read from the data set WORK.NEWSALESEMPs.

Output Window

Partial PROC PRINT Output

Obs	First_Name	Last_Name	Job_Title	Salary
1	Satyakam	Denny	Sales Rep. II	26780
2	Monica	Kletschkus	Sales Rep. IV	30890
3	Kevin	Lyon	Sales Rep. I	26955
4	Petrea	Soltau	Sales Rep. II	27440
5	Marina	Iyengar	Sales Rep. III	29715
6	Shani	Duckett	Sales Rep. I	25795
7	Fang	Wilson	Sales Rep. II	26810
8	Michael	Minas	Sales Rep. I	26970
9	Amanda	Liebman	Sales Rep. II	27465
10	Vincent	Eastley	Sales Rep. III	29695
11	Viney	Barbis	Sales Rep. III	30265
12	Skev	Rusli	Sales Rep. II	26580
13	Narelle	James	Sales Rep. III	29990
14	Gerry	Snellings	Sales Rep. I	26445
15	Leonid	Karavdic	Sales Rep. II	27860

Output Window

PROC MEANS Output

The MEANS Procedure						
Analysis Variable : Salary						
Job_Title	N Obs	N	Mean	Std Dev	Minimum	Maximum
Sales Rep. I	21	21	26418.81	713.1898498	25275.00	27475.00
Sales Rep. II	9	9	26902.22	592.9487283	26080.00	27860.00
Sales Rep. III	11	11	29345.91	989.4311956	28025.00	30785.00
Sales Rep. IV	6	6	31215.00	545.4997709	30305.00	31865.00
Temp. Sales Rep.	24	24	26265.83	732.6480659	25020.00	27480.00

Chapter 3: Working with SAS Syntax



3.1 Mastering Fundamental Concepts

3.2 Diagnosing and Correcting Syntax Errors

Chapter 3: Working with SAS Syntax

3.1 Mastering Fundamental Concepts

3.2 Diagnosing and Correcting Syntax Errors

Objectives

- Identify the characteristics of SAS statements.
- Explain SAS syntax rules.
- Insert SAS comments using two methods.

SAS Programs

A *SAS program* is a sequence of steps.

```
data work.NewSalesEmps;  
    length First Name $ 12  
           Last Name $ 18 Job Title $ 25;  
    infile 'newemps.csv' dlm=' ', '  
    input First Name $ Last Name $  
          Job Title $ Salary;  
run;  
  
proc print data=work.NewSalesEmps;  
run;  
  
proc means data=work.NewSalesEmps;  
    class Job Title;  
    var Salary;  
run;
```

DATA
Step

PROC
Step

PROC
Step

A *step* is a sequence of SAS statements.

Statements

SAS statements have these characteristics:

- usually begin with an **identifying keyword**
- always end with a **semicolon**

```
data work.NewSalesEmps;  
    length First_Name $ 12  
           Last_Name $ 18 Job_Title $ 25;  
    infile 'newemps.csv' dlm=';',  
    input First_Name $ Last_Name $  
          Job_Title $ Salary;  
run;  
  
proc print data=work.NewSalesEmps;  
run;  
  
proc means data=work.NewSalesEmps;  
    class Job_Title;  
    var Salary;  
run;
```

Poll

Quiz



3.01 Quiz

How many statements are in the DATA step?

- a. 1
- b. 3
- c. 5
- d. 7

```
data work.NewSalesEmps;  
    length First Name $ 12  
           Last Name $ 18 Job Title $ 25;  
    infile 'newemps.csv' dlm='|';  
    input First Name $ Last Name $  
          Job Title $ Salary;  
run;
```

3.01 Quiz – Correct Answer

How many statements are in the DATA step?

- a. 1
- b. 3
- c. 5**
- d. 7

```
data work.NewSalesEmps;  
    length First Name $ 12  
           Last_Name $ 18 Job Title $ 25;  
    infile 'newemps.csv' dlm=' ', '  
    input First Name $ Last Name $  
          Job Title $ Salary;  
run;
```

SAS Syntax Rules

Structured, consistent spacing makes a SAS program easier to read.

Conventional Formatting

```
data work.NewSalesEmps;  
    length First Name $ 12  
           Last Name $ 18 Job Title $ 25;  
    infile 'newemps.csv' dlm=' ', '  
    input First Name $ Last Name $  
           Job Title $ Salary;  
run;  
  
proc print data=work.NewSalesEmps;  
run;  
  
proc means data=work.NewSalesEmps;  
    class Job Title;  
    var Salary;  
run;
```

SAS Syntax Rules

- SAS statements are free-format.
- One or more blanks or special characters can be used to separate words.
- They can begin and end in any column.
- A single statement can span multiple lines.
- Several statements can be on the same line.

```
data work.NewSalesEmps;  
length First Name $ 12  
Last Name $ 18 Job Title $ 25;  
infile 'newemps.csv' dlm=',';  
input First Name $ Last Name $  
Job Title $ Salary;  
run;  
proc print data=work.NewSalesEmps; run;  
    proc means data =work.NewSalesEmps;  
class Job Title; var Salary;run;
```

Unconventional Formatting

SAS Syntax Rules

- SAS statements are free-format.
- ➔ ■ One or more blanks or special characters can be used to separate words.
- They can begin and end in any column.
- A single statement can span multiple lines.
- Several statements can be on the same line.

```
data work.NewSalesEmps;  
length First Name $ 12  
Last Name $ 18 Job Title $ 25;  
infile 'newemps.csv' dlm=',';  
input First Name $ Last Name $  
Job Title $ Salary;  
run;  
proc print data=work.NewSalesEmps; run;  
proc means data=work.NewSalesEmps;  
class Job Title; var Salary; run;
```

Unconventional Formatting

SAS Syntax Rules

- SAS statements are free-format.
- One or more blanks or special characters can be used to separate words.
- ➔ ■ They can begin and end in any column.
- A single statement can span multiple lines.
- Several statements can be on the same line.

```
data work.NewSalesEmps;  
length First Name $ 12  
Last Name $ 18 Job Title $ 25;  
infile 'newemps.csv' dlm=',';  
input First Name $ Last Name $  
Job Title $ Salary;  
run;  
proc print data=work.NewSalesEmps; run;  
proc means data =work.NewSalesEmps;  
class Job Title; var Salary;run;
```

Unconventional Formatting

SAS Syntax Rules

- SAS statements are free-format.
- One or more blanks or special characters can be used to separate words.
- They can begin and end in any column.
- A single statement can span multiple lines.
- Several statements can be on the same line.

```
data work.NewSalesEmps;  
length First Name $ 12  
Last Name $ 18 Job Title $ 25;  
infile 'newemps.csv' dlm=',';  
input First Name $ Last Name $  
Job Title $ Salary;  
run;  
proc print data=work.NewSalesEmps; run;  
    proc means data =work.NewSalesEmps;  
class Job Title; var Salary;run;
```

Unconventional Formatting

SAS Syntax Rules

- SAS statements are free-format.
- One or more blanks or special characters can be used to separate words.
- They can begin and end in any column.
- A single statement can span multiple lines.
- ➡ ■ Several statements can be on the same line.

```
data work.NewSalesEmps;  
length First Name $ 12  
Last Name $ 18 Job Title $ 25;  
infile 'newemps.csv' dlm=',';  
input First Name $ Last Name $  
Job Title $ Salary;  
run;  
proc print data=work.NewSalesEmps; run;  
proc means data =work.NewSalesEmps;  
class Job Title; var Salary;run;
```

Unconventional Formatting

SAS Comments

SAS comments are text that SAS ignores during processing. You can use comments anywhere in a SAS program to document the purpose of the program, explain segments of the program, or mark SAS code as non-executing text.

Two methods of commenting:

```
/* comment */
```

```
* comment ;
```

SAS Comments

This program contains four comments.

```
*-----*
|   This program creates and uses the   |
|   data set called work.NewSalesEmps.   |
*-----*;

data work.NewSalesEmps;
    length First_Name $ 12 Last_Name $ 18
           Job_Title $ 25;
    infile 'newemps.csv' dlm=',';
    input First_Name $ Last_Name $
           Job_Title $ Salary /*numeric*/;

run;
/*
proc print data=work.NewSalesEmps;
run;
*/
proc means data=work.NewSalesEmps;
    *class Job_Title;
    var Salary;
run;
```

Header Demo

Poll 

Quiz

Setup for the Poll

- Retrieve program **p103a01**.
- Read the comment concerning DATALINES.
- Submit the program and view the log to confirm that the PROC CONTENTS step did not execute.

3.02 Multiple Choice Poll

Which statement is true concerning the DATALINES statement based on reading the comment?

- a. The DATALINES statement is used when reading data located in a raw data file.
- b. The DATALINES statement is used when reading data located directly in the program.

3.02 Multiple Choice Poll – Correct Answer

Which statement is true concerning the DATALINES statement based on reading the comment?

- a. The DATALINES statement is used when reading data located in a raw data file.
- ☒ b. The DATALINES statement is used when reading data located directly in the program.



Question & Answer

Chapter 3: Working with SAS Syntax



3.1 Mastering Fundamental Concepts

3.2 Diagnosing and Correcting Syntax Errors

Syntax Errors

Syntax errors occur when program statements do not conform to the rules of the SAS language.

Examples of syntax errors:

- misspelled keywords
- unmatched quotation marks
- missing semicolons
- invalid options

When SAS encounters a syntax error, SAS prints a warning or an error message to the log.

```
ERROR 22-322: Syntax error, expecting one of the following:  
a name, a quoted string, (, /, ;, _DATA_, _LAST_,  
_NULL_.
```

Poll

Quiz



3.03 Quiz

This program has three syntax errors.

What are the errors?

```
daat work.NewSalesEmps;  
  length First Name $ 12  
          Last Name $ 18 Job Title $ 25;  
  infile 'newemps.csv' dlm=' ', '  
  input First Name $ Last Name $  
          Job Title $ Salary;  
run;  
  
proc print data=work.NewSalesEmps  
run;  
  
proc means data=work.NewSalesEmps average max;  
  class Job Title;  
  var Salary;  
run;
```

3.03 Quiz – Correct Answer

This program has three syntax errors.

What are the errors?

```
daat work.NewSalesEmps;  
  length First Name $ 12  
         Last Name $ 18 Job Title $ 25;  
  infile 'newemps.csv' dlm=' ', '  
  input First Name $ Last Name $  
         Job Title $ Salary;  
run;  
  
proc print data=work.NewSalesEmps  
run;  
  
proc means data=work.NewSalesEmps average max;  
  class Job Title;  
  var Salary;  
run;
```