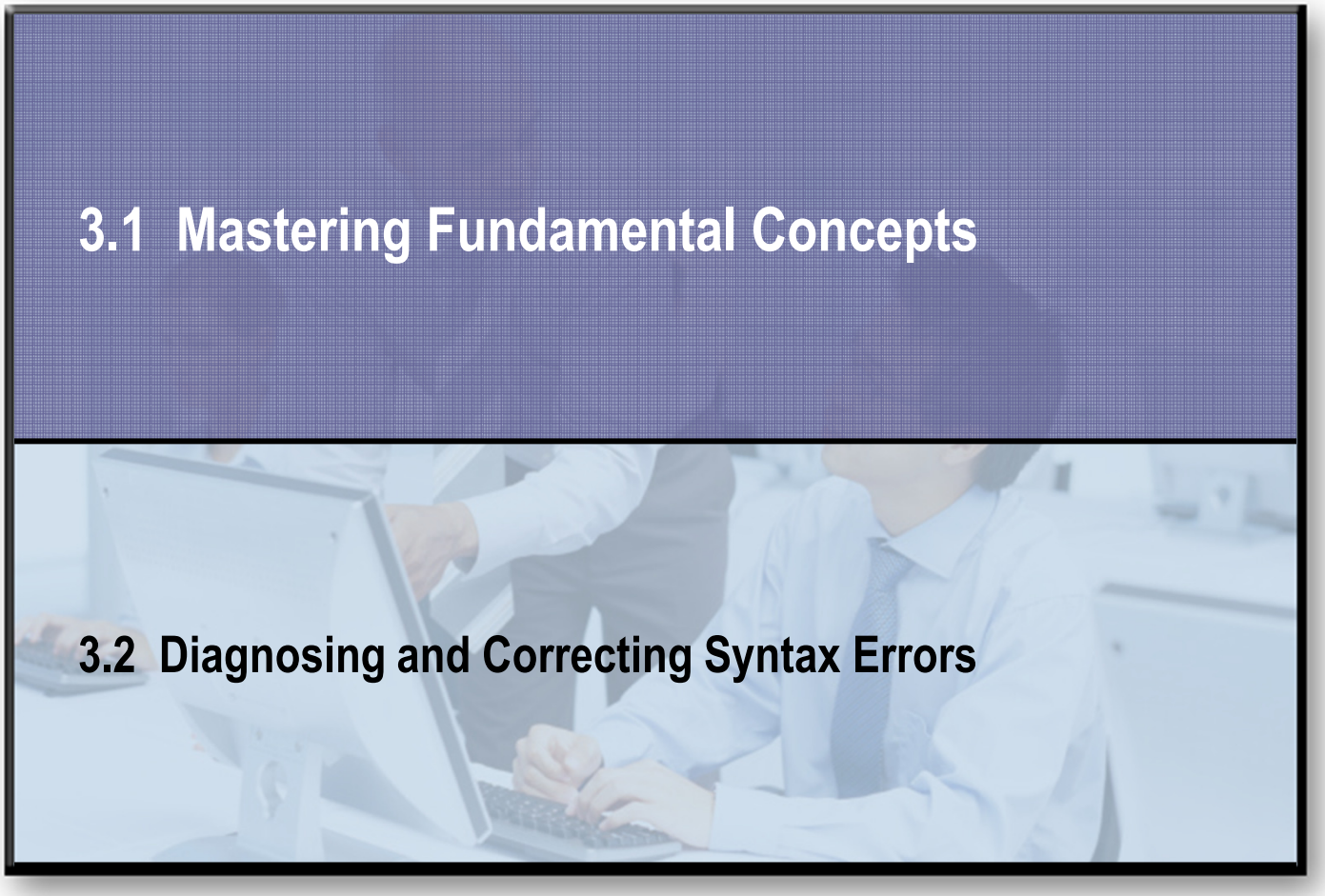
Chapter 3: Working with SAS Syntax



Objectives

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

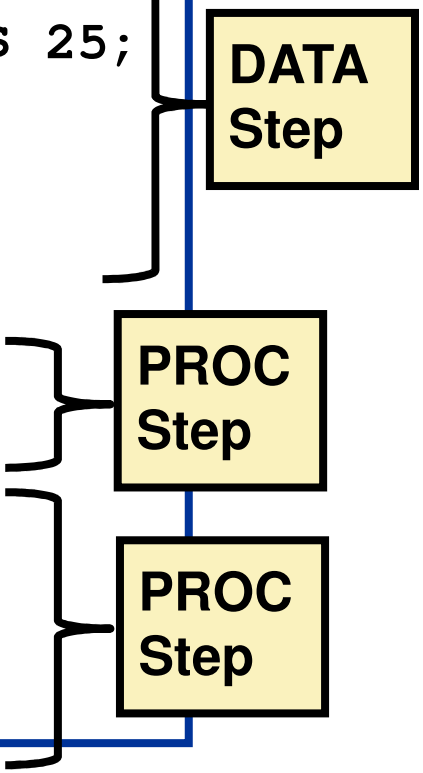
SAS Programs

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;
```



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;
```

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;
```

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.

Unconventional 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.

Unconventional 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.

Unconventional 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.

Unconventional 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.

Unconventional 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 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;
```

p103d02

Setup for the Poll

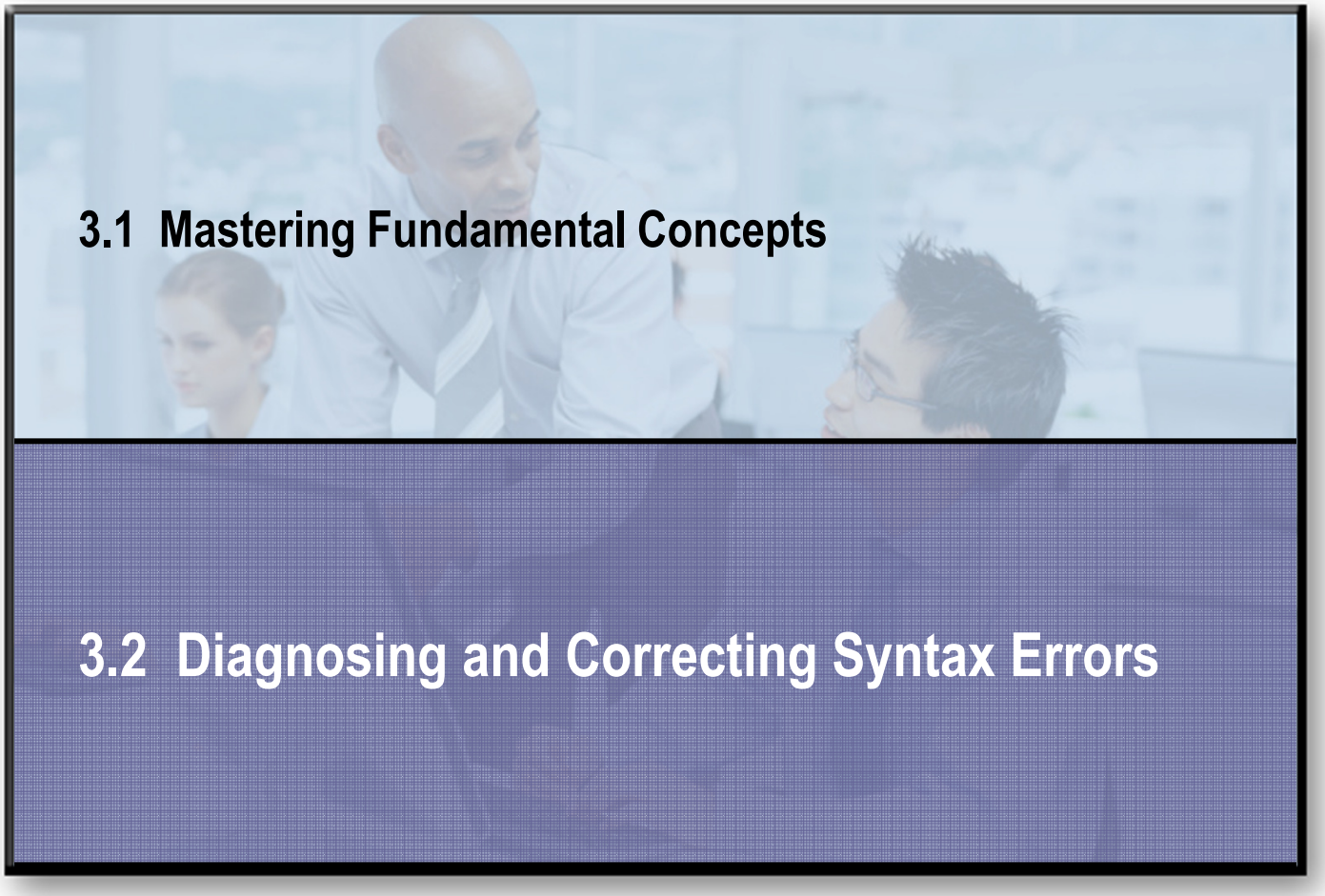
- Retrieve program **p103a01**.
- Read the comment concerning DATALINES.
- Submit the program and view the log to confirm that the PROC CONTENTS step did not execute.

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.

Chapter 3: Working with SAS Syntax



Objectives

- Identify SAS syntax errors.
- Diagnose and correct a program with errors.
- Save the corrected program.

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_.
```

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;
```

p103d03

Chapter Review

1. With what do SAS statements usually begin?

2. With what do SAS statements always end?

3. What are two methods of commenting?

4. Name four types of syntax errors.

5. How do you save a program?