Chapter 3
Understanding How the DATA Step Works

Arthur Li

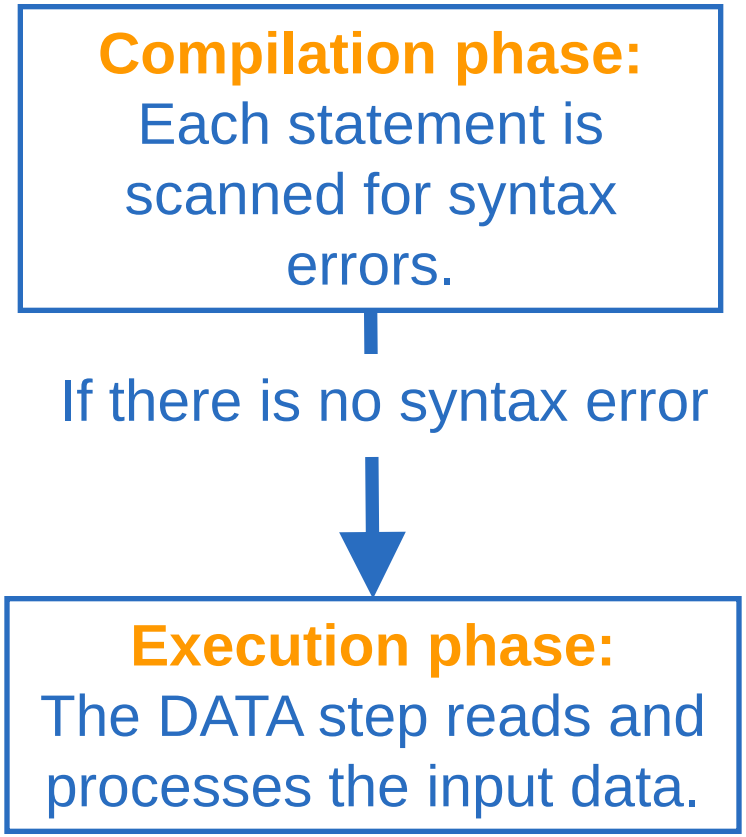
DATA Step Processing Overview

- A common befuddlement for beginning programmer:
  - The newly-created SAS dataset is not what we intended .
  - There are more or less observations.
  - The value of the variable was not retained correctly.

- Reason:
  - Learning only SAS language syntax.
  - Not understanding the fundamental SAS programming concepts.

DATA Step Processing Overview

A DATA step is processed in two-phase sequences:



DATA Step Processing Overview

- SAS statements in the DATA step:
  - executable
  - declarative

- The declarative statements:
  - provide information to SAS and only take effect during the compilation phase
  - can be placed in any order within the DATA step
  - LENGTH, FORMAT, LABEL, DROP, KEEP

- Executable statements: The order of the statement matters

DATA Step Processing Overview

Example3_1.txt

```
12345678901234567890
Barbara 61 12D
John    62 175
```

Variable names	Columns
Name	1-7
Height	9-10
Weight	12-14

- Read the external file and create BMI:
  - INFILE: identify the location of the external file
  - INPUT: instruct SAS how to read observation
  - Create BMI

DATA Step Processing Overview

Example3_1.txt

```
12345678901234567890
Barbara 61 12D
John    62 175
```

Data Entry Error

Variable names	Columns
Name	1-7
Height	9-10
Weight	12-14

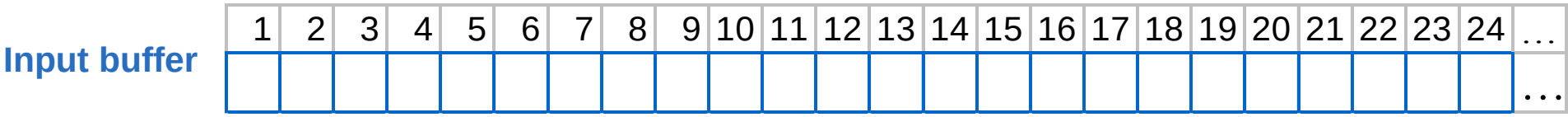
- Read the external file and create BMI:
  - INFILE: identify the location of the external file
  - INPUT: instruct SAS how to read observation
  - Create BMI

Program 3.1:

```
data ex3_1;
     infile 'W:\SAS Book\dat\example3_1.txt';
     input name $ 1-7 height 9-10 weight 12-14;
     BMI = 700*weight/(height*height);
     output;
run;
```

DATA Step Compilation Phase

```
data ex3_1;
    infile 'W:\SAS Book\dat\example3_1.txt';
    input name $ 1-7 height 9-10 weight 12-14;
    BMI = 700*weight/(height*height);
    output;
run;
```



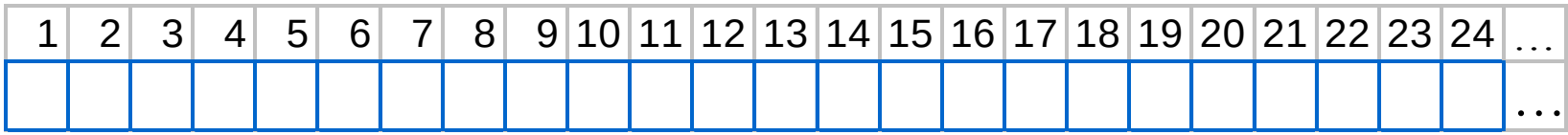
- Used to hold raw data
- Will not be created when reading a SAS dataset

DATA Step Compilation Phase

```
data ex3_1;
    infile 'W:\SAS Book\dat\example3_1.txt';
    input name $ 1-7 height 9-10 weight 12-14;
    BMI = 700*weight/(height*height);
    output;
run;
```

PDV is created

Input buffer



PDV

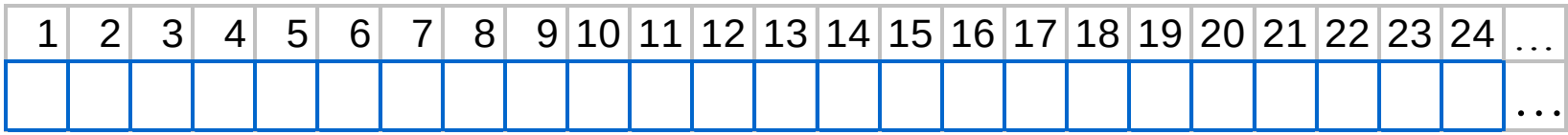
Memory area where SAS builds its new data set, 1 observation at a time.

DATA Step Compilation Phase

```
data ex3_1;
    infile 'W:\SAS Book\dat\example3_1.txt';
    input name $ 1-7 height 9-10 weight 12-14;
    BMI = 700*weight/(height*height);
    output;
run;
```

PDV is created

Input buffer



PDV

Automatic variables:

_N_ = 1: 1st observation is being processed

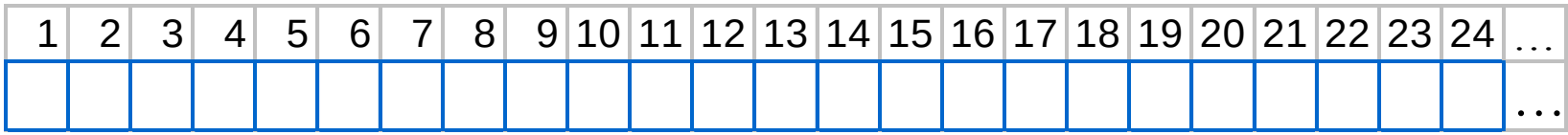
_N_ = 2: 2nd observation is being processed

DATA Step Compilation Phase

```
data ex3_1;
    infile 'W:\SAS Book\dat\example3_1.txt';
    input name $ 1-7 height 9-10 weight 12-14;
    BMI = 700*weight/(height*height);
    output;
run;
```

PDV is created

Input buffer



PDV

Automatic variables:
ERROR = 1: signals the data error of the currently-processed observation

DATA Step Compilation Phase

```
data ex3_1;
    infile 'W:\SAS Book\dat\example3_1.txt';
    input name $ 1-7 height 9-10 weight 12-14;
    BMI = 700*weight/(height*height);
    output;
run;
```

PDV is created

Input buffer

1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	…
																								…

PDV

_N_	D	_ERROR_	D	NAME	K	HEIGHT	K	WEIGHT	K	

A space is added to the PDV for each variable

DATA Step Compilation Phase

```
data ex3_1;
    infile 'W:\SAS Book\dat\example3_1.txt';
    input name $ 1-7 height 9-10 weight 12-14;
    BMI = 700*weight/(height*height);
    output;
run;
```

PDV is created

Input buffer

1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	…
																								…

PDV

_N_	D	_ERROR_	D	NAME	K	HEIGHT	K	WEIGHT	K	BMI	K

BMI is added to the PDV

DATA Step Compilation Phase

```
data ex3_1;
    infile 'W:\SAS Book\dat\example3_1.txt';
    input name $ 1-7 height 9-10 weight 12-14;
    BMI = 700*weight/(height*height);
    output;
run;
```

PDV is created

Input buffer

1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	…
																								…

PDV

_N_	D	_ERROR_	D	NAME	K	HEIGHT	K	WEIGHT	K	BMI	K

D = dropped

K = kept

DATA Step Compilation Phase

```
data ex3_1;
    infile 'W:\SAS Book\dat\example3_1.txt';
    input name $ 1-7 height 9-10 weight 12-14;
    BMI = 700*weight/(height*height);
    output;
run;
```

PDV is created

Input buffer

1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	…
																								…

PDV

_N_	D	_ERROR_	D	NAME	K	HEIGHT	K	WEIGHT	K	BMI	K

- Checks for syntax errors
  - invalid variable names
  - invalid options
  - incorrect punctuations
  - misspelled keywords

Execution Phase of Program 3.1

```
data ex3_1;
     infile 'W:\SAS Book\dat\example3_1.txt';
     input name $ 1-7 height 9-10 weight 12-14;
     BMI = 700*weight/(height*height);
     output;
run;
```

Input buffer

1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	…
																								…

PDV

_N_	D	_ERROR_	D	NAME	K	HEIGHT	K	WEIGHT	K	BMI	K
1		0				.		.		.	

1st Iteration:

- At the beginning
  - _N_ ←1, _ERROR_ ← 0

Execution Phase of Program 3.1

```
data ex3_1;
     infile 'W:\SAS Book\dat\example3_1.txt';
     input name $ 1-7 height 9-10 weight 12-14;
     BMI = 700*weight/(height*height);
     output;
run;
```

Input buffer

1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	…
																								…

PDV

_N_	D	_ERROR_	D	NAME	K	HEIGHT	K	WEIGHT	K	BMI	K
1		0				.		.		.	

1st Iteration:

- At the beginning
  - _N_ ←1, _ERROR_ ← 0
  - The remaining variables are set to *missing*

Execution Phase of Program 3.1

```
data ex3_1;
   infile 'W:\SAS Book\dat\example3_1.txt';
   input name $ 1-7 height 9-10 weight 12-14;
   BMI = 700*weight/(height*height);
   output;
run;
```

Example3_1.txt

```
12345678901234567890
Barbara 61 12D
John    62 175
```

Input buffer

1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	…
																								…

PDV

_N_	D	_ERROR_	D	NAME	K	HEIGHT	K	WEIGHT	K	BMI	K
1		0				.		.		.	

1st Iteration:

- The INFILE statement identifies the location of Exampl1.txt

Execution Phase of Program 3.1

```
data ex3_1;
    infile 'W:\SAS Book\dat\example3_1.txt';
    input name $ 1-7 height 9-10 weight 12-14;
    BMI = 700*weight/(height*height);
    output;
run;
```

Example3_1.txt

```
12345678901234567890
Barbara 61 12D
John    62 175
```

Input buffer

1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	…
B	a	r	b	a	r	a		6	1		1	2	D											…

PDV

_N_	D	_ERROR_	D	NAME	K	HEIGHT	K	WEIGHT	K	BMI	K
1		0				.		.		.	

1st Iteration:

- The INPUT statement copies the 1st data line into the input buffer

Execution Phase of Program 3.1

```
data ex3_1;
    infile 'W:\SAS Book\dat\example3_1.txt';
    input name $ 1-7 height 9-10 weight 12-14;
    BMI = 700*weight/(height*height);
    output;
run;
```

Example3_1.txt

```
12345678901234567890
Barbara 61 12D
John    62 175
```

Input buffer

1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	…
B	a	r	b	a	r	a		6	1		1	2	D											…

PDV

_N_	D	_ERROR_	D	NAME	K	HEIGHT	K	WEIGHT	K	BMI	K
1		0				.		.		.	

1st Iteration:

- The input pointer positions at the beginning of the input buffer

Execution Phase of Program 3.1

```
data ex3_1;
    infile 'W:\SAS Book\dat\example3_1.txt';
    input name $ 1-7 height 9-10 weight 12-14;
    BMI = 700*weight/(height*height);
    output;
run;
```

Example3_1.txt

```
12345678901234567890
Barbara 61 12D
John    62 175
```

Input buffer

1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	…
B	a	r	b	a	r	a		6	1		1	2	D											…

PDV

_N_	D	_ERROR_	D	NAME	K	HEIGHT	K	WEIGHT	K	BMI	K
1		0				.		.		.	

1st Iteration:

- The INPUT statement reads data values: input buffer → PDV

Execution Phase of Program 3.1

```
data ex3_1;
    infile 'W:\SAS Book\dat\example3_1.txt';
    input name $ 1-7 height 9-10 weight 12-14;
    BMI = 700*weight/(height*height);
    output;
run;
```

Example3_1.txt

```
12345678901234567890
Barbara 61 12D
John    62 175
```

Input buffer

1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	…
B	a	r	b	a	r	a		6	1		1	2	D											…

PDV

_N_	D	_ERROR_	D	NAME	K	HEIGHT	K	WEIGHT	K	BMI	K
1		0		Barbara		.		.		.	

1st Iteration:

- input buffer (columns 1-7) → “Name” in the PDV

Execution Phase of Program 3.1

```
data ex3_1;
    infile 'W:\SAS Book\dat\example3_1.txt';
    input name $ 1-7 height 9-10 weight 12-14;
    BMI = 700*weight/(height*height);
    output;
run;
```

Example3_1.txt

```
12345678901234567890
Barbara 61 12D
John    62 175
```

Input buffer

1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	…
B	a	r	b	a	r	a		6	1		1	2	D											…

PDV

_N_	D	_ERROR_	D	NAME	K	HEIGHT	K	WEIGHT	K	BMI	K
1		0		Barbara		.		.		.	

1st Iteration:

- The input pointer @ column 8

Execution Phase of Program 3.1

```
data ex3_1;
    infile 'W:\SAS Book\dat\example3_1.txt';
    input name $ 1-7 height 9-10 weight 12-14;
    BMI = 700*weight/(height*height);
    output;
run;
```

Example3_1.txt

```
12345678901234567890
Barbara 61 12D
John    62 175
```

Input buffer

1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	…
B	a	r	b	a	r	a		6	1		1	2	D											…

PDV

_N_	D	_ERROR_	D	NAME	K	HEIGHT	K	WEIGHT	K	BMI	K
1		0		Barbara		61		.		.	

1st Iteration:

- input buffer (columns 9-10) → “Height” in the PDV

Execution Phase of Program 3.1

```
data ex3_1;
    infile 'W:\SAS Book\dat\example3_1.txt';
    input name $ 1-7 height 9-10 weight 12-14;
    BMI = 700*weight/(height*height);
    output;
run;
```

Example3_1.txt

```
12345678901234567890
Barbara 61 12D
John    62 175
```

Input buffer

1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	…
B	a	r	b	a	r	a		6	1		1	2	D											…

PDV

_N_	D	_ERROR_	D	NAME	K	HEIGHT	K	WEIGHT	K	BMI	K
1		0		Barbara		61		.		.	

1st Iteration:

- The input pointer @ column 11

Execution Phase of Program 3.1

```
data ex3_1;
    infile 'W:\SAS Book\dat\example3_1.txt';
    input name $ 1-7 height 9-10 weight 12-14;
    BMI = 700*weight/(height*height);
    output;
run;
```

Example3_1.txt

```
12345678901234567890
Barbara 61 12D
John    62 175
```

Input buffer

1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	…
B	a	r	b	a	r	a		6	1		1	2	D											…

PDV

_N_	D	_ERROR_	D	NAME	K	HEIGHT	K	WEIGHT	K	BMI	K
1		0		Barbara		61		.		.	

1st Iteration:

- Tries to read Weight – invalid value

Execution Phase of Program 3.1

```
data ex3_1;
    infile 'W:\SAS Book\dat\example3_1.txt';
    input name $ 1-7 height 9-10 weight 12-14;
    BMI = 700*weight/(height*height);
    output;
run;
```

Example3_1.txt

```
12345678901234567890
Barbara 61 12D
John    62 175
```

Input buffer

1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	…
B	a	r	b	a	r	a		6	1		1	2	D											…

PDV

_N_	D	_ERROR_	D	NAME	K	HEIGHT	K	WEIGHT	K	BMI	K
1		1		Barbara		61		.		.	

1st Iteration:

- Tries to read Weight – invalid value
- _ERROR_ ← 1

Execution Phase of Program 3.1

```
data ex3_1;
    infile 'W:\SAS Book\dat\example3_1.txt';
    input name $ 1-7 height 9-10 weight 12-14;
    BMI = 700*weight/(height*height);
    output;
run;
```

Example3_1.txt

```
12345678901234567890
Barbara 61 12D
John    62 175
```

Input buffer

1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	…
B	a	r	b	a	r	a		6	1		1	2	D											…

PDV

_N_	D	_ERROR_	D	NAME	K	HEIGHT	K	WEIGHT	K	BMI	K
1		1		Barbara		61		.		.	

1st Iteration:

- The input pointer @ column 15

Execution Phase of Program 3.1

```
data ex3_1;
    infile 'W:\SAS Book\dat\example3_1.txt';
    input name $ 1-7 height 9-10 weight 12-14;
    BMI = 700*weight/(height*height);
    output;
run;
```

Example3_1.txt

```
12345678901234567890
Barbara 61 12D
John    62 175
```

Input buffer

1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	…
B	a	r	b	a	r	a		6	1		1	2	D											…

PDV

_N_	D	_ERROR_	D	NAME	K	HEIGHT	K	WEIGHT	K	BMI	K
1		1		Barbara		61		.		.	

1st Iteration:

- BMI will remain missing:
  operations on a missing value → a missing value

Execution Phase of Program 3.1

```
data ex3_1;
     infile 'W:\SAS Book\dat\example3_1.txt';
     input name $ 1-7 height 9-10 weight 12-14;
     BMI = 700*weight/(height*height);
     output;
run;
```

Example3_1.txt

```
12345678901234567890
Barbara 61 12D
John    62 175
```

Input buffer

1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	…
B	a	r	b	a	r	a		6	1		1	2	D											…

PDV

_N_	D	_ERROR_	D	NAME	K	HEIGHT	K	WEIGHT	K	BMI	K
1		1		Barbara		61		.		.	

1st Iteration:

- OUTPUT statement is executed
- Only values marked with (K) are copied as a single observation to the SAS dataset ex1

Ex3_1:

	Name	Height	Weight	BMI
1	Barbara	61	.	.

Execution Phase of Program 3.1

```
data ex3_1;
     infile 'W:\SAS Book\dat\example3_1.txt';
     input name $ 1-7 height 9-10 weight 12-14;
     BMI = 700*weight/(height*height);
     output;
run;
```

Example3_1.txt

```
12345678901234567890
Barbara 61 12D
John    62 175
```

Input buffer

1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	…
B	a	r	b	a	r	a		6	1		1	2	D											…

PDV

_N_	D	_ERROR_	D	NAME	K	HEIGHT	K	WEIGHT	K	BMI	K
1		1		Barbara		61		.		.	

1st Iteration:

- At the end of the DATA step, two things occur automatically:

Ex3_1:

	Name	Height	Weight	BMI
1	Barbara	61	.	.

Execution Phase of Program 3.1

```
data ex3_1;
     infile 'W:\SAS Book\dat\example3_1.txt';
     input name $ 1-7 height 9-10 weight 12-14;
     BMI = 700*weight/(height*height);
     output;
run;
```

Example3_1.txt

```
12345678901234567890
Barbara 61 12D
John    62 175
```

Input buffer

1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	…
B	a	r	b	a	r	a		6	1		1	2	D											…

PDV

_N_	D	_ERROR_	D	NAME	K	HEIGHT	K	WEIGHT	K	BMI	K
1		1		Barbara		61		.		.	

1. The SAS system returns to the beginning of the DATA step

Ex3_1:

	Name	Height	Weight	BMI
1	Barbara	61	.	.

Execution Phase of Program 3.1

```
data ex3_1;
     infile 'W:\SAS Book\dat\example3_1.txt';
     input name $ 1-7 height 9-10 weight 12-14;
     BMI = 700*weight/(height*height);
     output;
run;
```

Example3_1.txt

```
12345678901234567890
Barbara 61 12D
John    62 175
```

Input buffer

1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	…
B	a	r	b	a	r	a		6	1		1	2	D											…

PDV

_N_	D	_ERROR_	D	NAME	K	HEIGHT	K	WEIGHT	K	BMI	K
2		0				.		.		.	

2. The values of the variables in the PDV are reset to *missing*
   _N_ ↑ 2
   ERROR ← 0

Ex3_1:

	Name	Height	Weight	BMI
1	Barbara	61	.	.

Execution Phase of Program 3.1

```
data ex3_1;
    infile 'W:\SAS Book\dat\example3_1.txt';
    input name $ 1-7 height 9-10 weight 12-14;
    BMI = 700*weight/(height*height);
    output;
run;
```

Example3_1.txt

```
12345678901234567890
Barbara 61 12D
John    62 175
```

Input buffer

1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	…
J	o	h	n					6	2		1	7	5											…

PDV

_N_	D	_ERROR_	D	NAME	K	HEIGHT	K	WEIGHT	K	BMI	K
2		0				.		.		.	

2nd Iteration:

- 2nd data line → input buffer
- The input pointer @ beginning of the input buffer

Ex3_1:

	Name	Height	Weight	BMI
1	Barbara	61	.	.

Execution Phase of Program 3.1

```
data ex3_1;
    infile 'W:\SAS Book\dat\example3_1.txt';
    input name $ 1-7 height 9-10 weight 12-14;
    BMI = 700*weight/(height*height);
    output;
run;
```

Example3_1.txt

```
12345678901234567890
Barbara 61 12D
John    62 175
```

Input buffer

1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	…
J	o	h	n					6	2		1	7	5											…

PDV

_N_	D	_ERROR_	D	NAME	K	HEIGHT	K	WEIGHT	K	BMI	K
2		0		John		62		175		.	

2nd Iteration:

- The INPUT statement: input buffer → PDV

Ex3_1:

	Name	Height	Weight	BMI
1	Barbara	61	.	.

Execution Phase of Program 3.1

```
data ex3_1;
    infile 'W:\SAS Book\dat\example3_1.txt';
    input name $ 1-7 height 9-10 weight 12-14;
    BMI = 700*weight/(height*height);
    output;
run;
```

Example3_1.txt

```
12345678901234567890
Barbara 61 12D
John    62 175
```

Input buffer

1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	…
J	o	h	n					6	2		1	7	5											…

PDV

_N_	D	_ERROR_	D	NAME	K	HEIGHT	K	WEIGHT	K	BMI	K
2		0		John		62		175		31.868	

2nd Iteration:

- BMI is calculated

Ex3_1:

	Name	Height	Weight	BMI
1	Barbara	61	.	.

Execution Phase of Program 3.1

```
data ex3_1;
    infile 'W:\SAS Book\dat\example3_1.txt';
    input name $ 1-7 height 9-10 weight 12-14;
    BMI = 700*weight/(height*height);
    output;
run;
```

Example3_1.txt

```
12345678901234567890
Barbara 61 12D
John    62 175
```

Input buffer

1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	…
J	o	h	n					6	2		1	7	5											…

PDV

_N_	D	_ERROR_	D	NAME	K	HEIGHT	K	WEIGHT	K	BMI	K
2		0		John		62		175		31.868	

2nd Iteration:

- The OUTPUT statement is executed

Ex3_1:

	Name	Height	Weight	BMI
1	Barbara	61	.	.
2	John	62	175	31.868

Execution Phase of Program 3.1

```
data ex3_1;
     infile 'W:\SAS Book\dat\example3_1.txt';
     input name $ 1-7 height 9-10 weight 12-14;
     BMI = 700*weight/(height*height);
     output;
run;
```

Example3_1.txt

```
12345678901234567890
Barbara 61 12D
John    62 175
```

Input buffer

1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	…
J	o	h	n					6	2		1	7	5											…

PDV

_N_	D	_ERROR_	D	NAME	K	HEIGHT	K	WEIGHT	K	BMI	K
2		0		John		62		175		31.868	

2nd Iteration:

- At the end of the DATA step, two things occur automatically:

Ex3_1:

	Name	Height	Weight	BMI
1	Barbara	61	.	.
2	John	62	175	31.868

Execution Phase of Program 3.1

```
data ex3_1;
     infile 'W:\SAS Book\dat\example3_1.txt';
     input name $ 1-7 height 9-10 weight 12-14;
     BMI = 700*weight/(height*height);
     output;
run;
```

Example3_1.txt

```
12345678901234567890
Barbara 61 12D
John    62 175
```

Input buffer

1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	…
J	o	h	n					6	2		1	7	5											…

PDV

_N_	D	_ERROR_	D	NAME	K	HEIGHT	K	WEIGHT	K	BMI	K
2		0		John		62		175		31.868	

1. The SAS system returns to the beginning of the DATA step

Ex3_1:

	Name	Height	Weight	BMI
1	Barbara	61	.	.
2	John	62	175	31.868

Execution Phase of Program 3.1

```
data ex3_1;
     infile 'W:\SAS Book\dat\example3_1.txt';
     input name $ 1-7 height 9-10 weight 12-14;
     BMI = 700*weight/(height*height);
     output;
run;
```

Example3_1.txt

```
12345678901234567890
Barbara 61 12D
John    62 175
```

Input buffer

1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	…
J	o	h	n					6	2		1	7	5											…

PDV

_N_	D	_ERROR_	D	NAME	K	HEIGHT	K	WEIGHT	K	BMI	K
3		0				.		.		.	

2. The values of the variables in the PDV are reset to *missing*
   _N_ ↑3

Ex3_1:

	Name	Height	Weight	BMI
1	Barbara	61	.	.
2	John	62	175	31.868

Execution Phase of Program 3.1

```
data ex3_1;
    infile 'W:\SAS Book\dat\example3_1.txt';
    input name $ 1-7 height 9-10 weight 12-14;
    BMI = 700*weight/(height*height);
    output;
run;

proc print data=ex3_1;
run;
```

The end-of-file marker

Example3_1.txt

```
12345678901234567890
Barbara 61 12D
John    62 175
```

- SAS attempts to read an observation from the input data set, but it reaches the end-of-file-marker, which means that there are no more observations to read
- The SAS system → next DATA/PROC step

The Importance of the OUTPUT Statement

- The explicit OUTPUT statement:

```
data ex3_1;
    infile 'W:\SAS Book\dat\example3_1.txt';
    input name $ 1-7 height 9-10 weight 12-14;
    BMI = 700*weight/(height*height);
    output;
run;
```

  - writes the current observation from the PDV to a SAS dataset immediately
  - not at the end of the DATA step

The Importance of the OUTPUT Statement

```
data ex3_1;
    infile 'W:\SAS Book\dat\example3_1.txt';
    input name $ 1-7 height 9-10 weight 12-14;
    BMI = 700*weight/(height*height);

run;
```

- The implicit OUTPUT:
  - It tells SAS to write observations to the dataset at the end of the DATA step
  - Without explicit OUTPUT statements, every DATA step contains an implicit OUTPUT statement at the end of the DATA step

The Importance of the OUTPUT Statement

- Placing an explicit OUTPUT
  - Override the implicit OUTPUT
  - SAS adds an observation to a dataset only when an explicit OUTPUT is executed
  - You can use more than one OUTPUT statement in the DATA step

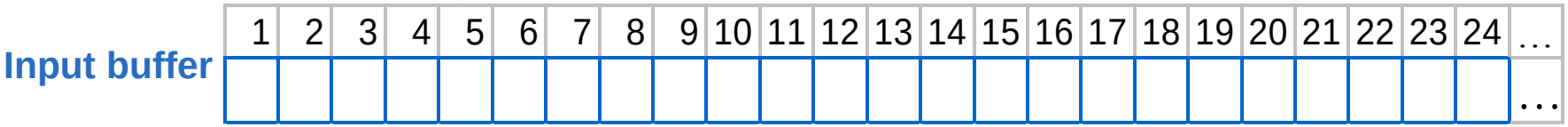
The Difference between Reading a Raw Data Set and a SAS Data Set

- When Reading a raw dataset …

```
data ex3_1;
    infile 'W:\SAS Book\dat\example3_1.txt';
    input name $ 1-7 height 9-10 weight 12-14;
    BMI = 700*weight/(height*height);
    output;
run;
```

Raw data

```
Barbara 61 12D
John    62 175
```



SAS dataset

	Name	Height	Weight	BMI
1	Barbara	61	.	.
2	John	62	175	31.868

The Difference between Reading a Raw Data Set and a SAS Data Set

- When Reading a SAS dataset …

```
data ex3_1;
     set example1;
     BMI = 700*weight/(height*height);
     output;
run;
```

SAS dataset

	Name	Height	Weight
1	Barbara	61	170
2	John	62	175

PDV

_N_ D	_ERROR_ D	NAME K	HEIGHT K	WEIGHT K	BMI K

SAS dataset

	Name	Height	Weight	BMI
1	Barbara	61	170	31.981
2	John	62	175	31.868

The Difference between Reading a Raw Data Set and a SAS Data Set

- When reading a raw dataset, SAS sets each variable value in the PDV to *missing* at the beginning of each iteration of execution, except for …
  - the automatic variables
  - variables that are named in the RETAIN or SUM statements
  - data elements in a _TEMPORARY_ array
  - variables created in the options of the FILE/INFILE statement

The Difference between Reading a Raw Data Set and a SAS Data Set

❖When Reading a SAS dataset …

```
data ex3_1;
     set example1;
     BMI = 700*weight/(height*height);
     output;
run;
```

Example1:

	Name	Height	Weight
1	Barbara	61	170
2	John	62	175

- ❖ SAS sets each variable to *missing* in the PDV only before the 1st iteration of the execution.
- ❖ Variables will retain their values in the PDV until they are replaced by the new values.

PDV

_N_	D	_ERROR_	D	NAME	K	HEIGHT	K	WEIGHT	K	BMI	K

Variables exist in the **input** dataset

The Difference between Reading a Raw Data Set and a SAS Data Set

- When Reading a SAS dataset …

```
data ex3_1;
     set example1;
     BMI = 700*weight/(height*height);
     output;
run;
```

- SAS sets each variable to *missing* in the PDV at the beginning of **every** iteration of the execution.

Example1:

	Name	Height	Weight
1	Barbara	61	170
2	John	62	175

PDV

_N_	D	_ERROR_	D	NAME	K	HEIGHT	K	WEIGHT	K	BMI	K

Variables being created in the DATA step

The Difference between Reading a Raw Data Set and a SAS Data Set

❖When Reading a SAS dataset …

```
data ex3_1;
     set example1;
     BMI = 700*weight/(height*height);
     output;
run;
```

Example1:

	Name	Height	Weight
1	Barbara	61	170
2	John	62	175

1st Iteration:

❖At the beginning of the execution phase, SAS sets each variable to *missing* in the PDV.

PDV

_N_ D	_ERROR_ D	NAME K	HEIGHT K	WEIGHT K	BMI K
1	0		.	.	.

The Difference between Reading a Raw Data Set and a SAS Data Set

❖When Reading a SAS dataset …

```
data ex3_1;
   set example1;
   BMI = 700*weight/(height*height);
   output;
run;
```

1st Iteration:

❖The SET statement is executed.

Example1:

	Name	Height	Weight
1	Barbara	61	170
2	John	62	175

PDV

_N_ D	_ERROR_ D	NAME K	HEIGHT K	WEIGHT K	BMI K
1	0	Barbara	61	170	.

The Difference between Reading a Raw Data Set and a SAS Data Set

❖When Reading a SAS dataset …

```
data ex3_1;
    set example1;
    BMI = 700*weight/(height*height);
    output;
run;
```

1st Iteration:

❖BMI is calculated.

Example1:

	Name	Height	Weight
1	Barbara	61	170
2	John	62	175

PDV

_N_ D	_ERROR_ D	NAME K	HEIGHT K	WEIGHT K	BMI K
1	0	Barbara	61	170	31.981

The Difference between Reading a Raw Data Set and a SAS Data Set

❖When Reading a SAS dataset …

```
data ex3_1;
     set example1;
     BMI = 700*weight/(height*height);
     output;
run;
```

1st Iteration:

❖The OUTPUT statement executes.

Example1:

	Name	Height	Weight
1	Barbara	61	170
2	John	62	175

PDV

_N_ D	_ERROR_ D	NAME K	HEIGHT K	WEIGHT K	BMI K
1	0	Barbara	61	170	31.981

Ex3_1:

	Name	Height	Weight	BMI
1	Barbara	61	170	31.981

The Difference between Reading a Raw Data Set and a SAS Data Set

❖When Reading a SAS dataset …

```
data ex3_1;
    set example1;
    BMI = 700*weight/(height*height);
    output;
run;
```

1st Iteration:

❖SAS reaches the end of DATA step.

Example1:

	Name	Height	Weight
1	Barbara	61	170
2	John	62	175

PDV

_N_ D	_ERROR_ D	NAME K	HEIGHT K	WEIGHT K	BMI K
1	0	Barbara	61	170	31.981

Ex3_1:

	Name	Height	Weight	BMI
1	Barbara	61	170	31.981

The Difference between Reading a Raw Data Set and a SAS Data Set

❖When Reading a SAS dataset …

```
data ex3_1;
    set example1;
    BMI = 700*weight/(height*height);
    output;
run;
```

Example1:

	Name	Height	Weight
1	Barbara	61	170
2	John	62	175

2nd Iteration:

❖_N_ ← 2

❖Variables will retain their values in the PDV until they are replaced by the new values from the input dataset, except for BMI.

PDV

_N_ D	_ERROR_ D	NAME K	HEIGHT K	WEIGHT K	BMI K
2	0	Barbara	61	170	.

Ex3_1:

	Name	Height	Weight	BMI
1	Barbara	61	170	31.981

The Difference between Reading a Raw Data Set and a SAS Data Set

❖When Reading a SAS dataset …

```
data ex3_1;
   set example1;
   BMI = 700*weight/(height*height);
   output;
run;
```

2nd Iteration:

❖The SET statement copies the 2nd observation to the PDV.

Example1:

	Name	Height	Weight
1	Barbara	61	170
2	John	62	175

PDV

_N_	D	_ERROR_	D	NAME	K	HEIGHT	K	WEIGHT	K	BMI	K
2		0		John		62		175		.	

Ex3_1:

	Name	Height	Weight	BMI
1	Barbara	61	170	31.981

The Difference between Reading a Raw Data Set and a SAS Data Set

❖When Reading a SAS dataset …

```
data ex3_1;
    set example1;
    BMI = 700*weight/(height*height);
    output;
run;
```

2nd Iteration:

❖BMI is calculated

Example1:

	Name	Height	Weight
1	Barbara	61	170
2	John	62	175

PDV

_N_ D	_ERROR_ D	NAME K	HEIGHT K	WEIGHT K	BMI K
2	0	John	62	175	31.868

Ex3_1:

	Name	Height	Weight	BMI
1	Barbara	61	170	31.981

Retaining the Value of Newly-Created Variables

Consider the following dataset:

	ID	SCORE
1	A01	3
2	A02	.
3	A03	4

Retaining the Value of Newly-Created Variables

Consider the following dataset:

	ID	SCORE	TOTAL
1	A01	3	3
2	A02	.	3
3	A03	4	7

Task: Create a new variable that accumulates the values of SCORE

Retaining the Value of Newly-Created Variables

Consider the following dataset:

	ID	SCORE	TOTAL
1	A01	3	3
2	A02	.	3
3	A03	4	7

- Approach:
  - Set the TOTAL to 0 at the first iteration of the execution
  - Then at each iteration of the execution, add values from SCORE to TOTAL

Retaining the Value of Newly-Created Variables

Consider the following dataset:

	ID	SCORE	TOTAL
1	A01	3	3
2	A02	.	3
3	A03	4	7

Problem:

- TOTAL is the variable that you are creating in the DATA step
- TOTAL will be set to *missing* at the beginning of the every iteration

- Approach:
  - Set the TOTAL to 0 at the first iteration of the execution
  - Then at each iteration of the execution, add values from SCORE to TOTAL

The RETAIN Statement

- To prevents the VARIABLE from being initialized each time the DATA step executes, use the RETAIN statement:

RETAIN variable <value>;

Name of the variable that you will want to retain

The RETAIN Statement

- To prevents the VARIABLE from being initialized each time the DATA step executes, use the RETAIN statement:

RETAIN variable <value>;

- A numeric value
- Used to initialize the VARIABLE *only* at the first iteration of the DATA step execution
- Not specifying an initial value → VARIABLE is initialized as *missing*

The Execution Phase of Program 3.3

```
data ex3_2;
     set sas3_1;
     retain total 0;
     total = sum(total, score);
run;
```

SAS3_1:

	ID	SCORE
1	A01	3
2	A02	.
3	A03	4

PDV

_N_	D	_ERROR_	D	ID	K	SCORE	K	TOTAL	K

- The execution phase begins immediately after the completion of the compilation phase

The Execution Phase of Program 3.3

```
data ex3_2;
     set sas3_1;
     retain total 0;
     total = sum(total, score);
run;
```

SAS3_1:

	ID	SCORE
1	A01	3
2	A02	.
3	A03	4

PDV

_N_	D	_ERROR_	D	ID	K	SCORE	K	TOTAL	K
1		0							

1st Iteration:

- _N_ ←1, _ERROR_ ← 0

The Execution Phase of Program 3.3

```
data ex3_2;
     set sas3_1;
     retain total 0;
     total = sum(total, score);
run;
```

SAS3_1:

	ID	SCORE
1	A01	3
2	A02	.
3	A03	4

PDV

_N_ D	_ERROR_ D	ID K	SCORE K	TOTAL K
1	0		.	

1st Iteration:

- _N_ ←1, _ERROR_ ← 0
- ID, SCORE ← missing

The Execution Phase of Program 3.3

```
data ex3_2;
     set sas3_1;
     retain total 0;
     total = sum(total, score);
run;
```

SAS3_1:

	ID	SCORE
1	A01	3
2	A02	.
3	A03	4

PDV

_N_	D	_ERROR_	D	ID	K	SCORE	K	TOTAL	K
1		0				.		0	

1st Iteration:

- _N_ ←1, _ERROR_ ← 0
- ID, SCORE ← missing
- TOTAL ← 0 because of the RETAIN statement

The Execution Phase of Program 3.3

```
data ex3_2;
   set sas3_1;
   retain total 0;
   total = sum(total, score);
run;
```

SAS3_1:

	ID	SCORE
1	A01	3
2	A02	.
3	A03	4

PDV

_N_	D	_ERROR_	D	ID	K	SCORE	K	TOTAL	K
1		0		A01		3		0	

1st Iteration:

- 1st observation from SAS3_1 → PDV

The Execution Phase of Program 3.3

```
data ex3_2;
    set sas3_1;
    retain total 0;
    total = sum(total, score);
run;
```

SAS3_1:

	ID	SCORE
1	A01	3
2	A02	.
3	A03	4

PDV

_N_ D	_ERROR_ D	ID K	SCORE K	TOTAL K
1	0	A01	3	0

1st Iteration:

- The RETAIN statement is declarative statement; it does not execute during the execution phase

The Execution Phase of Program 3.3

```
data ex3_2;
     set sas3_1;
     retain total 0;
     total = sum(total, score);
run;
```

SAS3_1:

	ID	SCORE
1	A01	3
2	A02	.
3	A03	4

PDV

_N_ D	_ERROR_ D	ID K	SCORE K	TOTAL K
1	0	A01	3	3

1st Iteration:

- TOTAL is calculated

The Execution Phase of Program 3.3

```
data ex3_2;
     set sas3_1;
     retain total 0;
     total = sum(total, score);
run;
```

SAS3_1:

	ID	SCORE
1	A01	3
2	A02	.
3	A03	4

PDV

_N_	D	_ERROR_	D	ID	K	SCORE	K	TOTAL	K
1		0		A01		3		3	

1st Iteration:

- The implicit OUTPUT statement tells the SAS system to write observations to the dataset

EX3_2:

	ID	SCORE	TOTAL
1	A01	3	3

The Execution Phase of Program 3.3

```
data ex3_2;
     set sas3_1;
     retain total 0;
     total = sum(total, score);
run;
```

SAS3_1:

	ID	SCORE
1	A01	3
2	A02	.
3	A03	4

PDV

_N_	D	_ERROR_	D	ID	K	SCORE	K	TOTAL	K
2		0		A01		3		3	

2nd Iteration:

- _N_ ↑2

EX3_2:

	ID	SCORE	TOTAL
1	A01	3	3

The Execution Phase of Program 3.3

```
data ex3_2;
     set sas3_1;
     retain total 0;
     total = sum(total, score);
run;
```

SAS3_1:

	ID	SCORE
1	A01	3
2	A02	.
3	A03	4

PDV

_N_ D	_ERROR_ D	ID K	SCORE K	TOTAL K
2	0	A01	3	3

2nd Iteration:

- ID and SCORE are retained from the previous iteration because they are read from an existing SAS data set

EX3_2:

	ID	SCORE	TOTAL
1	A01	3	3

The Execution Phase of Program 3.3

```
data ex3_2;
     set sas3_1;
     retain total 0;
     total = sum(total, score);
run;
```

SAS3_1:

	ID	SCORE
1	A01	3
2	A02	.
3	A03	4

PDV

_N_	D	_ERROR_	D	ID	K	SCORE	K	TOTAL	K
2		0		A01		3		3	

2nd Iteration:

- TOTAL is also retained because the RETAIN statement is used

EX3_2:

	ID	SCORE	TOTAL
1	A01	3	3

The Execution Phase of Program 3.3

```
data ex3_2;
   set sas3_1;
   retain total 0;
   total = sum(total, score);
run;
```

SAS3_1:

	ID	SCORE
1	A01	3
2	A02	.
3	A03	4

PDV

_N_	D	_ERROR_	D	ID	K	SCORE	K	TOTAL	K
2		0		A02		.		3	

2nd Iteration:

- 2nd observation from SAS3_1 → PDV

EX3_2:

	ID	SCORE	TOTAL
1	A01	3	3

The Execution Phase of Program 3.3

```
data ex3_2;
     set sas3_1;
     retain total 0;
     total = sum(total, score);
run;
```

SAS3_1:

	ID	SCORE
1	A01	3
2	A02	.
3	A03	4

PDV

_N_	D	_ERROR_	D	ID	K	SCORE	K	TOTAL	K
2		0		A02		.		3	

2nd Iteration:

- TOTAL is calculated

EX3_2:

	ID	SCORE	TOTAL
1	A01	3	3

The Execution Phase of Program 3.3

```
data ex3_2;
     set sas3_1;
     retain total 0;
     total = sum(total, score);
run;
```

SAS3_1:

	ID	SCORE
1	A01	3
2	A02	.
3	A03	4

PDV

_N_	D	_ERROR_	D	ID	K	SCORE	K	TOTAL	K
2		0		A02		.		3	

2nd Iteration:

- The implicit OUTPUT:
  The contents in PDV → EX3_2

EX3_2:

	ID	SCORE	TOTAL
1	A01	3	3
2	A02	.	3

The Execution Phase of Program 3.3

```
data ex3_2;
     set sas3_1;
     retain total 0;
     total = sum(total, score);
run;
```

SAS3_1:

	ID	SCORE
1	A01	3
2	A02	.
3	A03	4

PDV

_N_	D	_ERROR_	D	ID	K	SCORE	K	TOTAL	K
3		0		A02		.		3	

3rd Iteration:

- _N_ ↑ 3
- ID and SCORE are retained from the previous iteration
- TOTAL is also retained

EX3_2:

	ID	SCORE	TOTAL
1	A01	3	3
2	A02	.	3

The Execution Phase of Program 3.3

```
data ex3_2;
   set sas3_1;
   retain total 0;
   total = sum(total, score);
run;
```

SAS3_1:

	ID	SCORE
1	A01	3
2	A02	.
3	A03	4

PDV

_N_ D	_ERROR_ D	ID K	SCORE K	TOTAL K
3	0	A03	4	3

3rd Iteration:

- 3rd observation from SAS3_1 → PDV

EX3_2:

	ID	SCORE	TOTAL
1	A01	3	3
2	A02	.	3

The Execution Phase of Program 3.3

```
data ex3_2;
     set sas3_1;
     retain total 0;
     total = sum(total, score);
run;
```

SAS3_1:

	ID	SCORE
1	A01	3
2	A02	.
3	A03	4

PDV

_N_	D	_ERROR_	D	ID	K	SCORE	K	TOTAL	K
3		0		A03		4		7	

3rd Iteration:

- TOTAL is calculated

EX3_2:

	ID	SCORE	TOTAL
1	A01	3	3
2	A02	.	3

The Execution Phase of Program 3.3

```
data ex3_2;
     set sas3_1;
     retain total 0;
     total = sum(total, score);
run;
```

SAS3_1:

	ID	SCORE
1	A01	3
2	A02	.
3	A03	4

PDV

_N_	D	_ERROR_	D	ID	K	SCORE	K	TOTAL	K
3		0		A03		4		7	

3rd Iteration:

- The implicit OUTPUT:
  The contents in PDV → EX3_2

EX3_2:

	ID	SCORE	TOTAL
1	A01	3	3
2	A02	.	3
3	A03	4	7

The SUM Statement

- The SUM statement has the following form:

```
variable+expression;
```

- The numeric accumulator variable that is to be created.
- It is automatically set to 0 at the beginning of the first iteration of the DATA step execution.
- Retained in following iterations.

The SUM Statement

- The SUM statement has the following form:

```
variable+expression;
```

- Any SAS expression.
- If EXPRESSION is evaluated to a missing value, it is treated as 0.

The SUM Statement

```
data ex3_2;
     set sas3_1;
     retain total 0;
     total = sum(total, score);
run;
```

```
data ex3_3;
     set sas3_1;
     total + score;
run;
```

Conditional Processing In the DATA Step

- Many applications require to process only part of the observations that meet the condition of a specified expression.

- In this situation, you need to use the subsetting IF statement.

The Subsetting IF Statement

- The IF statement:

```
IF expression;
```

- If the EXPRESSION is true for the observation
  - SAS continues to execute statements in the DATA step and includes the current observation in the data set
  - The resulting SAS data set contains a subset of the external file or SAS data set

The Subsetting IF Statement

- The IF statement:

```
IF expression;
```

- If the EXPRESSION is false
  - no further statements are processed for that observation
  - the current observation is not written to the data set
  - the remaining program statements in the DATA step are not executed
  - SAS immediately returns to the beginning of the DATA step

The Subsetting IF Statement

Program 3.5

```
data ex3_4;
    set sas3_1;
    total + score;
    if not missing(score);
run;

title 'Keep observations only when SCORE is not missing';
proc print data=ex3_4;
run;
```

Keeping observations for SCORE is not missing

Obs	ID	score	total
1	A01	3	3
2	A03	4	7

The Subsetting IF Statement

- Sometimes it is more efficient (or easier) to specify a condition for excluding observations:

IF expression **THEN DELETE;**

Program 3.6

```
data ex3_5;
     set sas3_1;
     total + score;
     if missing(score) then delete;
run;
```

Detecting the End of a Data Set by Using the END= Option

- Sometimes you might want to determine when the last observation in an input data set has been read
- You can create a temporary variable by using the END= option in the SET statement

SET SAS-data-set **END**=variable;

- A temporary variable that contains an end-of-file indicator
- VARIABLE ← 0 at the beginning of the DATA step iteration
- VARIABLE ← 1 when SET reads the last observation of the input data set
- VARIABLE is not added to any new data seta

Detecting the End of a Data Set by Using the END= Option

- To calculate the total score and list total # of observations

Program 3.7

```
data total_score(keep = total n);
    set sas3_1 end = last;
    total + score;
    n + 1;
    if last;
run;
title 'Only keep the last observation';
proc print data=total_score;
run;
```

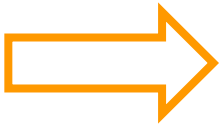
Only keep the last observation

Obs	total	n
1	7	3

Restructuring Data Sets from Wide Format to Long Format

- Restructuring datasets:

data with one observation per subject (the wide format)



data with multiple observations per subject (the long format)

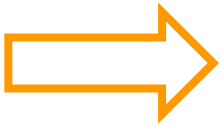
	ID	S1	S2	S3
1	A01	3	4	5
2	A02	4	.	2

	ID	TIME	SCORE
1	A01	1	3
2	A01	2	4
3	A01	3	5
4	A02	1	4
5	A02	3	2

Restructuring Data Sets from Wide Format to Long Format

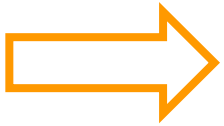
- Restructuring datasets:

data with one observation per subject (the wide format)



data with multiple observations per subject (the long format)

	ID	S1	S2	S3
1	A01	3	4	5
2	A02	4	.	2

S1 – S3  SCORE

	ID	TIME	SCORE
1	A01	1	3
2	A01	2	4
3	A01	3	5
4	A02	1	4
5	A02	3	2

Distinguish different measurements for each subject

Execution Phase of Program 3.8

```
data long (drop=s1-s3);
 set wide;
 time = 1;
 score = s1;
 if not missing(score) then output;
 time = 2;
 score = s2;
 if not missing(score) then output;
 time = 3;
 score = s3;
 if not missing(score) then output;
run;
```

Wide:

	ID	S1	S2	S3
1	A01	3	4	5
2	A02	4	.	2

PDV

_N_	D	ID	K	S1	D	S2	D	S3	D	TIME	K	SCORE	K

- The execution phase begins immediately after the completion of the compilation phase
- _ERROR_ is not shown for simplicity purpose

Execution Phase of Program 3.8

```
data long (drop=s1-s3);
 set wide;
 time = 1;
 score = s1;
 if not missing(score) then output;
 time = 2;
 score = s2;
 if not missing(score) then output;
 time = 3;
 score = s3;
 if not missing(score) then output;
run;
```

Wide:

	ID	S1	S2	S3
1	A01	3	4	5
2	A02	4	.	2

PDV

_N_	D	ID	K	S1	D	S2	D	S3	D	TIME	K	SCORE	K
1				.		.		.		.		.	

1st iteration:

- _N_ ← 1
- Other variables ← *missing*

Execution Phase of Program 3.8

```
data long (drop=s1-s3);
 set wide;
 time = 1;
 score = s1;
 if not missing(score) then output;
 time = 2;
 score = s2;
 if not missing(score) then output;
 time = 3;
 score = s3;
 if not missing(score) then output;
run;
```

Wide:

	ID	S1	S2	S3
1	A01	3	4	5
2	A02	4	.	2

PDV

_N_	D	ID	K	S1	D	S2	D	S3	D	TIME	K	SCORE	K
1		A01		3		4		5		.		.	

1st iteration:

- 1st observation from the wide → PDV

Execution Phase of Program 3.8

```
data long (drop=s1-s3);
 set wide;
 time = 1;
 score = s1;
 if not missing(score) then output;
 time = 2;
 score = s2;
 if not missing(score) then output;
 time = 3;
 score = s3;
 if not missing(score) then output;
run;
```

Wide:

	ID	S1	S2	S3
1	A01	3	4	5
2	A02	4	.	2

PDV

_N_	D	ID	K	S1	D	S2	D	S3	D	TIME	K	SCORE	K
1		A01		3		4		5		1		.	

1st iteration:

- Time ← 1

Execution Phase of Program 3.8

```
data long (drop=s1-s3);
 set wide;
 time = 1;
 score = s1;
 if not missing(score) then output;
 time = 2;
 score = s2;
 if not missing(score) then output;
 time = 3;
 score = s3;
 if not missing(score) then output;
run;
```

Wide:

	ID	S1	S2	S3
1	A01	3	4	5
2	A02	4	.	2

PDV

_N_	D	ID	K	S1	D	S2	D	S3	D	TIME	K	SCORE	K
1		A01		3		4		5		1		3	

1st iteration:

- Score ← value from S1(3)

Execution Phase of Program 3.8

```
data long (drop=s1-s3);
 set wide;
 time = 1;
 score = s1;
 if not missing(score) then output;
 time = 2;
 score = s2;
 if not missing(score) then output;
 time = 3;
 score = s3;
 if not missing(score) then output;
run;
```

Wide:

	ID	S1	S2	S3
1	A01	3	4	5
2	A02	4	.	2

Long:

	ID	TIME	SCORE
1	A01	1	3

PDV

_N_	D	ID	K	S1	D	S2	D	S3	D	TIME	K	SCORE	K
1		A01		3		4		5		1		3	

1st iteration:

- SCORE ≠ *missing*: ID, TIME, and SCORE → Long

Execution Phase of Program 3.8

```
data long (drop=s1-s3);
 set wide;
 time = 1;
 score = s1;
 if not missing(score) then output;
 time = 2;
 score = s2;
 if not missing(score) then output;
 time = 3;
 score = s3;
 if not missing(score) then output;
run;
```

Wide:

	ID	S1	S2	S3
1	A01	3	4	5
2	A02	4	.	2

Long:

	ID	TIME	SCORE
1	A01	1	3

PDV

_N_	D	ID	K	S1	D	S2	D	S3	D	TIME	K	SCORE	K
1		A01		3		4		5		2		3	

1st iteration:

- TIME ← 2

Execution Phase of Program 3.8

```
data long (drop=s1-s3);
 set wide;
 time = 1;
 score = s1;
 if not missing(score) then output;
 time = 2;
 score = s2;
 if not missing(score) then output;
 time = 3;
 score = s3;
 if not missing(score) then output;
run;
```

Wide:

	ID	S1	S2	S3
1	A01	3	4	5
2	A02	4	.	2

Long:

	ID	TIME	SCORE
1	A01	1	3

PDV

_N_	D	ID	K	S1	D	S2	D	S3	D	TIME	K	SCORE	K
1		A01		3		4		5		2		4	

1st iteration:

- Score ← value from S2(4)

Execution Phase of Program 3.8

```
data long (drop=s1-s3);
 set wide;
 time = 1;
 score = s1;
 if not missing(score) then output;
 time = 2;
 score = s2;
 if not missing(score) then output;
 time = 3;
 score = s3;
 if not missing(score) then output;
run;
```

Wide:

	ID	S1	S2	S3
1	A01	3	4	5
2	A02	4	.	2

Long:

	ID	TIME	SCORE
1	A01	1	3
2	A01	2	4

PDV

_N_	D	ID	K	S1	D	S2	D	S3	D	TIME	K	SCORE	K
1		A01		3		4		5		2		4	

1st iteration:

- SCORE ≠*missing*: ID, TIME, and SCORE → Long

Execution Phase of Program 3.8

```
data long (drop=s1-s3);
 set wide;
 time = 1;
 score = s1;
 if not missing(score) then output;
 time = 2;
 score = s2;
 if not missing(score) then output;
 time = 3;
 score = s3;
 if not missing(score) then output;
run;
```

Wide:

	ID	S1	S2	S3
1	A01	3	4	5
2	A02	4	.	2

Long:

	ID	TIME	SCORE
1	A01	1	3
2	A01	2	4

PDV

_N_	D	ID	K	S1	D	S2	D	S3	D	TIME	K	SCORE	K
1		A01		3		4		5		3		4	

1st iteration:

- TIME ← 3

Execution Phase of Program 3.8

```
data long (drop=s1-s3);
 set wide;
 time = 1;
 score = s1;
 if not missing(score) then output;
 time = 2;
 score = s2;
 if not missing(score) then output;
 time = 3;
 score = s3;
 if not missing(score) then output;
run;
```

Wide:

	ID	S1	S2	S3
1	A01	3	4	5
2	A02	4	.	2

Long:

	ID	TIME	SCORE
1	A01	1	3
2	A01	2	4

PDV

_N_	D	ID	K	S1	D	S2	D	S3	D	TIME	K	SCORE	K
1		A01		3		4		5		3		5	

1st iteration:

- SCORE ←value from S3(5)

Execution Phase of Program 3.8

```
data long (drop=s1-s3);
 set wide;
 time = 1;
 score = s1;
 if not missing(score) then output;
 time = 2;
 score = s2;
 if not missing(score) then output;
 time = 3;
 score = s3;
 if not missing(score) then output;
run;
```

Wide:

	ID	S1	S2	S3
1	A01	3	4	5
2	A02	4	.	2

Long:

	ID	TIME	SCORE
1	A01	1	3
2	A01	2	4
3	A01	3	5

PDV

_N_	D	ID	K	S1	D	S2	D	S3	D	TIME	K	SCORE	K
1		A01		3		4		5		3		5	

1st iteration:

- SCORE ≠*missing*: ID, TIME, and SCORE → Long

Execution Phase of Program 3.8

```
data long (drop=s1-s3);
 set wide;
 time = 1;
 score = s1;
 if not missing(score) then output;
 time = 2;
 score = s2;
 if not missing(score) then output;
 time = 3;
 score = s3;
 if not missing(score) then output;
run;
```

Wide:

	ID	S1	S2	S3
1	A01	3	4	5
2	A02	4	.	2

Long:

	ID	TIME	SCORE
1	A01	1	3
2	A01	2	4
3	A01	3	5

PDV

_N_	D	ID	K	S1	D	S2	D	S3	D	TIME	K	SCORE	K
1		A01		3		4		5		3		5	

1st iteration:

- There is no more implicit OUTPUT statement
- SAS returns to the beginning of the DATA step to begin the 2nd iteration

Execution Phase of Program 3.8

```
data long (drop=s1-s3);
 set wide;
 time = 1;
 score = s1;
 if not missing(score) then output;
 time = 2;
 score = s2;
 if not missing(score) then output;
 time = 3;
 score = s3;
 if not missing(score) then output;
run;
```

Wide:

	ID	S1	S2	S3
1	A01	3	4	5
2	A02	4	.	2

Long:

	ID	TIME	SCORE
1	A01	1	3
2	A01	2	4
3	A01	3	5

PDV

_N_	D	ID	K	S1	D	S2	D	S3	D	TIME	K	SCORE	K
2		A01		3		4		5		.		.	

2nd iteration:

- _N_ ↑2
- ID and S1-S3 are retained from the previous iteration
- TIME, SCORE ← missing

Execution Phase of Program 3.8

```
data long (drop=s1-s3);
 set wide;
 time = 1;
 score = s1;
 if not missing(score) then output;
 time = 2;
 score = s2;
 if not missing(score) then output;
 time = 3;
 score = s3;
 if not missing(score) then output;
run;
```

Wide:

	ID	S1	S2	S3
1	A01	3	4	5
2	A02	4	.	2

Long:

	ID	TIME	SCORE
1	A01	1	3
2	A01	2	4
3	A01	3	5

PDV

_N_	D	ID	K	S1	D	S2	D	S3	D	TIME	K	SCORE	K
2		A02		4		.		2		.		.	

2nd iteration:

- 2nd observation from the Wide → PDV

Execution Phase of Program 3.8

```
data long (drop=s1-s3);
 set wide;
 time = 1;
 score = s1;
 if not missing(score) then output;
 time = 2;
 score = s2;
 if not missing(score) then output;
 time = 3;
 score = s3;
 if not missing(score) then output;
run;
```

Wide:

	ID	S1	S2	S3
1	A01	3	4	5
2	A02	4	.	2

Long:

	ID	TIME	SCORE
1	A01	1	3
2	A01	2	4
3	A01	3	5

PDV

_N_	D	ID	K	S1	D	S2	D	S3	D	TIME	K	SCORE	K
2		A02		4		.		2		1		.	

2nd iteration:

- TIME ← 1

Execution Phase of Program 3.8

```
data long (drop=s1-s3);
 set wide;
 time = 1;
 score = s1;
 if not missing(score) then output;
 time = 2;
 score = s2;
 if not missing(score) then output;
 time = 3;
 score = s3;
 if not missing(score) then output;
run;
```

Wide:

	ID	S1	S2	S3
1	A01	3	4	5
2	A02	4	.	2

Long:

	ID	TIME	SCORE
1	A01	1	3
2	A01	2	4
3	A01	3	5

PDV

_N_	D	ID	K	S1	D	S2	D	S3	D	TIME	K	SCORE	K
2		A02		4		.		2		1		4	

2nd iteration:

- SCORE ← value from S1 (4)

Execution Phase of Program 3.8

```
data long (drop=s1-s3);
 set wide;
 time = 1;
 score = s1;
 if not missing(score) then output;
 time = 2;
 score = s2;
 if not missing(score) then output;
 time = 3;
 score = s3;
 if not missing(score) then output;
run;
```

Wide:

	ID	S1	S2	S3
1	A01	3	4	5
2	A02	4	.	2

Long:

	ID	TIME	SCORE
1	A01	1	3
2	A01	2	4
3	A01	3	5
4	A02	1	4

PDV

_N_	D	ID	K	S1	D	S2	D	S3	D	TIME	K	SCORE	K
2		A02		4		.		2		1		4	

2nd iteration:

- ID, TIME, and SCORE → Long

Execution Phase of Program 3.8

```
data long (drop=s1-s3);
 set wide;
 time = 1;
 score = s1;
 if not missing(score) then output;
 time = 2;
 score = s2;
 if not missing(score) then output;
 time = 3;
 score = s3;
 if not missing(score) then output;
run;
```

Wide:

	ID	S1	S2	S3
1	A01	3	4	5
2	A02	4	.	2

Long:

	ID	TIME	SCORE
1	A01	1	3
2	A01	2	4
3	A01	3	5
4	A02	1	4

PDV

_N_	D	ID	K	S1	D	S2	D	S3	D	TIME	K	SCORE	K
2		A02		4		.		2		2		4	

2nd iteration:

- TIME ← 2

Execution Phase of Program 3.8

```
data long (drop=s1-s3);
 set wide;
 time = 1;
 score = s1;
 if not missing(score) then output;
 time = 2;
 score = s2;
 if not missing(score) then output;
 time = 3;
 score = s3;
 if not missing(score) then output;
run;
```

Wide:

	ID	S1	S2	S3
1	A01	3	4	5
2	A02	4	.	2

Long:

	ID	TIME	SCORE
1	A01	1	3
2	A01	2	4
3	A01	3	5
4	A02	1	4

PDV

_N_	D	ID	K	S1	D	S2	D	S3	D	TIME	K	SCORE	K
2		A02		4		.		2		2		.	

2nd iteration:

- SCORE ← the value from S2 (*missing*)

Execution Phase of Program 3.8

```
data long (drop=s1-s3);
 set wide;
 time = 1;
 score = s1;
 if not missing(score) then output;
 time = 2;
 score = s2;
 if not missing(score) then output;
 time = 3;
 score = s3;
 if not missing(score) then output;
run;
```

Wide:

	ID	S1	S2	S3
1	A01	3	4	5
2	A02	4	.	2

Long:

	ID	TIME	SCORE
1	A01	1	3
2	A01	2	4
3	A01	3	5
4	A02	1	4

PDV

_N_	D	ID	K	S1	D	S2	D	S3	D	TIME	K	SCORE	K
2		A02		4		.		2		2		.	

2nd iteration:

- SCORE = *missing*: no output is generated

Execution Phase of Program 3.8

```
data long (drop=s1-s3);
 set wide;
 time = 1;
 score = s1;
 if not missing(score) then output;
 time = 2;
 score = s2;
 if not missing(score) then output;
 time = 3;
 score = s3;
 if not missing(score) then output;
run;
```

Wide:

	ID	S1	S2	S3
1	A01	3	4	5
2	A02	4	.	2

Long:

	ID	TIME	SCORE
1	A01	1	3
2	A01	2	4
3	A01	3	5
4	A02	1	4

PDV

_N_	D	ID	K	S1	D	S2	D	S3	D	TIME	K	SCORE	K
2		A02		4		.		2		3		2	

2nd iteration:

- SCORE ← the value from S3 (2)

Execution Phase of Program 3.8

```
data long (drop=s1-s3);
 set wide;
 time = 1;
 score = s1;
 if not missing(score) then output;
 time = 2;
 score = s2;
 if not missing(score) then output;
 time = 3;
 score = s3;
 if not missing(score) then output;
run;
```

Wide:

	ID	S1	S2	S3
1	A01	3	4	5
2	A02	4	.	2

Long:

	ID	TIME	SCORE
1	A01	1	3
2	A01	2	4
3	A01	3	5
4	A02	1	4
5	A02	3	2

PDV

_N_	D	ID	K	S1	D	S2	D	S3	D	TIME	K	SCORE	K
2		A02		4		.		2		3		2	

2nd iteration:

- ID, TIME, and SCORE → Long

Execution Phase of Program 3.8

```
data long (drop=s1-s3);
 set wide;
 time = 1;
 score = s1;
 if not missing(score) then output;
 time = 2;
 score = s2;
 if not missing(score) then output;
 time = 3;
 score = s3;
 if not missing(score) then output;
run;
```

Wide:

	ID	S1	S2	S3
1	A01	3	4	5
2	A02	4	.	2

Long:

	ID	TIME	SCORE
1	A01	1	3
2	A01	2	4
3	A01	3	5
4	A02	1	4
5	A02	3	2

PDV

_N_	D	ID	K	S1	D	S2	D	S3	D	TIME	K	SCORE	K
2		A02		4		.		2		3		2	

2nd iteration:

- SAS reaches the end of the DATA step.
- SAS returns to the beginning of the DATA step to begin the 3rd iteration
- With no more observations to read in the 3rd iteration, SAS goes to the next DATA or PROC step

Debugging Techniques

- Syntax errors are often easier to detect than logic errors
  - SAS not only stops programs
  - SAS generates detailed error messages
- Logic errors often result in generating an unintended data set and they are difficult to debug

Using the PUT Statement to Observe the Contents of the PDV

- One way to detect a logic error is to use the PUT statement in the DATA step:

PUT variable | variable-list | character-string;

Program 3.9

```
data ex3_4;
    put "1st PUT" _all_;
    set sas3_1;
    put "2nd PUT" _all_;
    total + score;
    put "3rd PUT" _all_;
    if not missing(score);
    put "4th PUT" _all_;
run;
```

Using the PUT Statement to Observe the Contents of the PDV

```
68   data ex3_4;
69       put "1st PUT" _all_;
70       set sas3_1;
71       put "2nd PUT" _all_;
72       total + score;
73       put "3rd PUT" _all_;
74       if not missing(score);
75       put "4th PUT" _all_;
76   run;

1st PUTID=  SCORE=. total=0 _ERROR_=0 _N_=1
2nd PUTID=A01 SCORE=3 total=0 _ERROR_=0 _N_=1
3rd PUTID=A01 SCORE=3 total=3 _ERROR_=0 _N_=1
4th PUTID=A01 SCORE=3 total=3 _ERROR_=0 _N_=1
1st PUTID=A01 SCORE=3 total=3 _ERROR_=0 _N_=2
2nd PUTID=A02 SCORE=. total=3 _ERROR_=0 _N_=2
3rd PUTID=A02 SCORE=. total=3 _ERROR_=0 _N_=2
1st PUTID=A02 SCORE=. total=3 _ERROR_=0 _N_=3
2nd PUTID=A03 SCORE=4 total=3 _ERROR_=0 _N_=3
3rd PUTID=A03 SCORE=4 total=7 _ERROR_=0 _N_=3
4th PUTID=A03 SCORE=4 total=7 _ERROR_=0 _N_=3
1st PUTID=A03 SCORE=4 total=7 _ERROR_=0 _N_=4
NOTE: There were 3 observations read from the data set WORK.SAS3_1.
NOTE: The data set WORK.EX3_4 has 2 observations and 3 variables.
NOTE: DATA statement used (Total process time):
      real time           0.01 seconds
      cpu time            0.03 seconds
```