

STAT604

Lesson SAS 07

Portions Copyright © 2009 SAS Institute Inc., Cary, NC, USA. All rights reserved. Reproduced with permission of SAS Institute Inc., Cary, NC, USA. SAS Institute Inc. makes no warranties with respect to these materials and disclaims all liability therefor.

**THE
POWER
TO KNOW®**

Chapter 5: Data Transformations

5.1 Introduction

5.2 Manipulating Character Values (Part 1)

5.3 Manipulating Character Values (Part 2)

5.4 Manipulating Numeric Values

5.5 Converting Variable Type

Objectives

- Use SAS functions to extract, edit, and search character values.

Business Scenario – Create Mailing List Data

The **orion.contacts** data set contains the contact information for each charity's representative.

Partial Listing of **orion.contacts**

ID	Title	Name	Address1	Address2
AQI	Ms.	Farr,Sue	15 Harvey Rd.	Macon, GA 31298
CCI	Dr.	Cox,Kay B.	163 McNeil Pl.	Kern, CA 93280
CNI	Mr.	Mason,Ron	442 Glen Ave.	Miami, FL 33054
CS	Ms.	Ruth,G. H.	2491 Brady St.	Munger, MI 48747

Address1 and **Address2** are in the correct form to use for a mailing address, but the **Title** and **Name** variables need to be combined into a new variable, **FullName**.

Business Scenario – Desired Output

Create a new data set, **labels**, that is suitable for creating mailing labels.

Partial Listing of **labels**

ID	FullName	Address1	Address2
AQI	Ms. Sue Farr	15 Harvey Rd.	Macon, GA 31298
CCI	Dr. Kay B. Cox	163 McNeil Pl.	Kern, CA 93280
CNI	Mr. Ron Mason	442 Glen Ave.	Miami, FL 33054
CS	Ms. G. H. Ruth	2491 Brady St.	Munger, MI 48747

Create Mailing List Data

Partial Listing of `orion.contacts`

Title	Name
Ms.	Farr,Sue
Dr.	Cox,Kay B.
Mr.	Mason,Ron
Ms.	Ruth,G. H.
Prof.	Florentino, Helen-Ashe H
Ms.	Van Allsburg, Jan F.
Mr.	Laff, Stanley X.
Mr.	Rizen, George Q.
Dr.	Mitchell, Marc J.
Ms.	Mills, Dorothy E.
Dr.	Webb, Jonathan W.
Mr.	Keenan, Maynard J.

Two steps need to be accomplished:

Step 1: Separate the last name from the first and middle names.

Step 2: Combine the title, the first and middle names, and the last name.

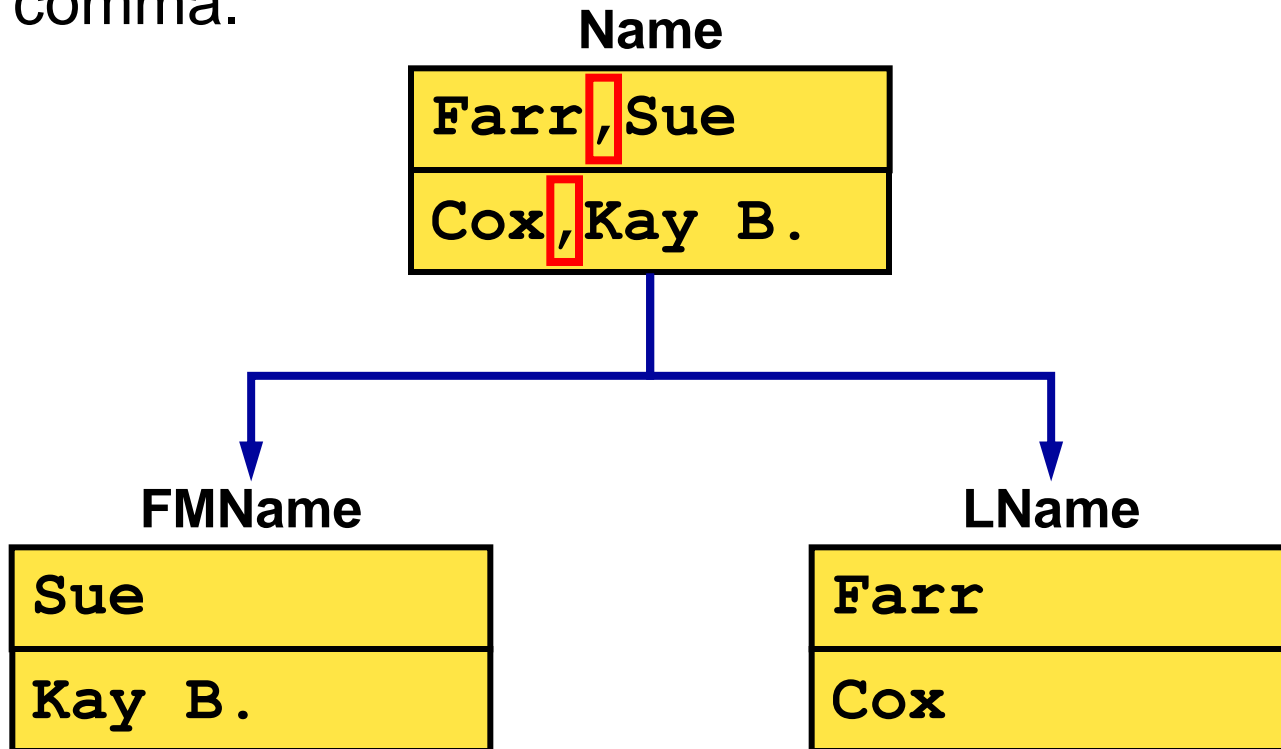
Poll



Quiz

Setup for the Poll

The first step in creating the mailing list is to separate the contact's name into two parts based on the position of the comma.



Would the SUBSTR function be appropriate for this?

5.06 Poll

Would the SUBSTR function be appropriate to separate the contact's name into two parts?

- ☐ Yes
- ☐ No

5.06 Poll – Correct Answer

Would the SUBSTR function be appropriate to separate the contact's name into two parts?

- ☐ Yes
- ☒ No

The SUBSTR function would be difficult to use because the comma is not in the same position for each value of name.

The SCAN function is a better choice to separate the last from the first and middle names.

The SCAN Function

The SCAN function returns the *n*th word of a character value.

General form of the SCAN function:

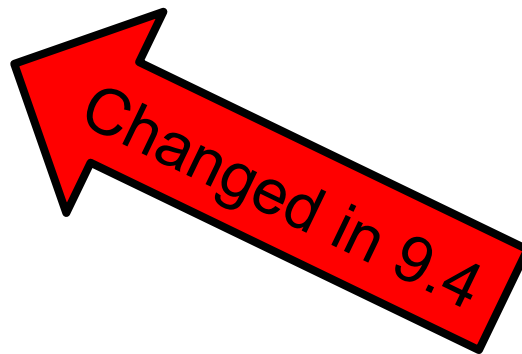
```
NewVar=SCAN(string,n<,charlist>);
```

<i>string</i>	can be a character constant, variable, or expression.
<i>n</i>	specifies the <i>n</i> th word to extract from <i>string</i> .
<i>charlist</i>	lists the character(s) that delimit words. If omitted, the default delimiters are as follows:
ASCII (PC, UNIX)	blank . < (+ & ! \$ *) ; - / , % ^
EBCDIC (z/OS)	blank . < (+ & ! \$ *) ; - / , % ¢ ¬

The SCAN Function – Details

When you use the SCAN function,

- a missing value is returned if there are fewer than n words in the string
- if n is negative, the SCAN function selects the word in the character string starting from the end of string
- the length of a new created variable is the length of the first argument.



- ✎ A good practice is to explicitly define the length of any created variable with a LENGTH statement.

The SCAN Function – Details

When you use the SCAN function,

- delimiters before the first word have no effect
- any character or set of characters can serve as delimiters
- two or more contiguous delimiters are treated as a single delimiter
- modifiers can be used as a fourth argument to change the default behavior.

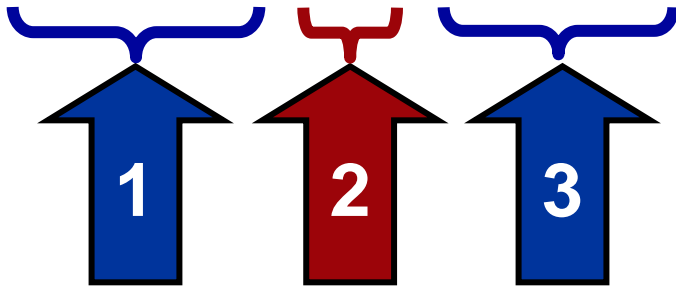
The SCAN Function – Example

Extract the second word of **Phrase**.

```
Second=scan (Phrase,2,' ' ) ;
```

PDV

Phrase \$ 21	Second \$ 21
software and services	and



Poll

Quiz



5.07 Quiz

Consider this PDV and assignment statement:

```
Second=scan (Phrase,2,' ','');
```

PDV

Phrase	Second
\$ 28	\$ 28
software, hardware, services	

What value will be stored in **Second**?

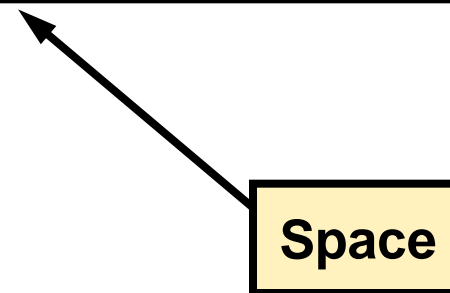
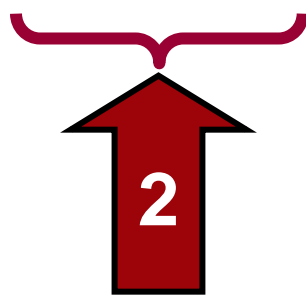
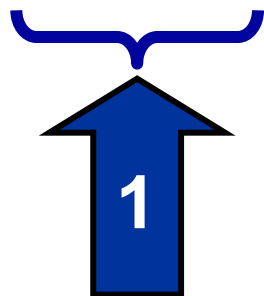
5.07 Quiz – Correct Answer

Consider this PDV and assignment statement:

```
Second=scan (Phrase,2,' ','');
```

PDV

Phrase \$ 28	Second \$ 28
software, hardware, services	hardware



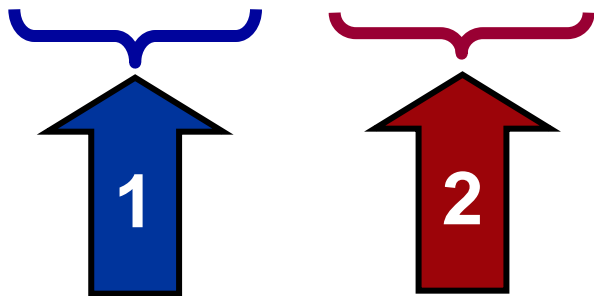
The SCAN Function – Example

Extract the second word of **Phrase** without the leading space.

```
Second=scan(Phrase,2,' ','');
```

PDV

Phrase \$ 28	Second \$ 28
software, hardware, services	hardware



Poll 

Quiz

Setup for the Poll

Consider this DATA step:

```
data Scan_Quiz;  
  Text = "New Year's Day, January 1st, 2007";  
  Year = ?;  
run;
```

5.08 Multiple Choice Poll

What expression completes the assignment statement to correctly extract 2007 from the **Text** variable?

- a. `scan(Text, -1) ;`
- b. `scan(Text, 6) ;`
- c. `scan(Text, 6, ' , ') ;`
- d. All of the above would work.

```
data Scan_Quiz;  
  Text = "New Year's Day, January 1st, 2007";  
  Year =  
run;
```

5.08 Multiple Choice Poll – Correct Answer

What expression completes the assignment statement to correctly extract 2007 from the **Text** variable?

- a. `scan(Text, -1) ;`
- b. `scan(Text, 6) ;`
- c. `scan(Text, 6, ' , ') ;`
- ☒ d. All of the above would work.

Create Mailing List Data

Using the SCAN function gives an easy way to separate the names for the mailing list.

```
data labels;  
  set orion.contacts;  
  length FMName LName $ 15;  
  FMName = scan(Name,2,' ');  
  LName = scan(Name,1,' ');  
run;
```

Create Mailing List Data

```
proc print data=labels noobs;  
    var ID Name Title FMName LName;  
run;
```

Partial PROC PRINT Output

ID	Name	Title	FMName	LName
AQI	Farr,Sue	Ms.	Sue	Farr
CCI	Cox,Kay B.	Dr.	Kay B.	Cox
CNI	Mason,Ron	Mr.	Ron	Mason
CS	Ruth,G. H.	Ms.	G. H.	Ruth

The next step is to join the values of **Title**, **FMName**, and **LName** into another variable.

The CATX Function

The CATX function joins or *concatenates* character strings.

General form of the CATX function:

$$NewVar = CATX(separator, string-1, \dots, string-n)$$

<i>separator</i>	Is a character string that is inserted between the concatenated <i>string-1</i> , ..., <i>string-n</i> arguments.
<i>string-1</i> , ..., <i>string-n</i>	can be a character constant, variable, or expression. Leading and trailing blanks are removed from each argument.

The size of the created variable, *NewVar*, is 200 bytes if it is not previously defined with a LENGTH statement.

The CATX Function – Example

Combine **FMName** and **LName** to create **FullName**.

```
FullName=catx(' ', FMName, LName) ;
```

PDV

FMName \$ 15	LName \$ 15	FullName \$ 200
Sue	Farr	Sue Farr

Other CAT Functions

There are three other CAT functions that concatenate character strings.

Function	Details
CAT(<i>string-1</i> , ... , <i>string-n</i>)	does not remove leading or trailing blanks from the arguments before concatenating them.
CATS(<i>string-1</i> , ... , <i>string-n</i>)	removes leading and trailing blanks from the arguments.
CATT(<i>string-1</i> , ... , <i>string-n</i>)	removes trailing blanks from the arguments.

Create Mailing List Data – Finished Program

Adding an assignment statement with the CATX function completes the program.

```
data labels;  
  set orion.contacts;  
  length FullName $ 35 FMName LName $ 15;  
  FMName = scan(Name,2,' ');  
  LName = scan(Name,1,' ');  
  FullName = catx(' ',Title,FMName,LName);  
run;
```

Create Mailing List Data – Finished Program

```
proc print data=labels noobs;  
    var ID FullName Address1 Address2;  
run;
```

Partial PROC PRINT Output

ID	FullName	Address1	Address2
AQI	Ms. Sue Farr	15 Harvey Rd.	Macon, GA 31298
CCI	Dr. Kay B. Cox	163 McNeil Pl.	Kern, CA 93280
CNI	Mr. Ron Mason	442 Glen Ave.	Miami, FL 33054
CS	Ms. G. H. Ruth	2491 Brady St.	Munger, MI 48747

Concatenation Operator

The *concatenation operator* is another way to join character strings.

General form of the concatenation operator:

```
NewVar=string1 !! string2;
```

Example: `Phone = ' (' !! area !! ') ' !! Number ;`

PDV

Area	Number	Phone
\$ 3	\$ 8	\$ 14
919	531-0000	(919) 531-0000



The operator can also be written as two vertical bars (||) or two broken vertical bars (|||).

Business Scenario: Data Clean Up

The Internet Sales Group accidentally used the wrong data files for the Orion Star Catalog Web site. They corrected the problem as soon as it was noticed, but some orders were created with data errors in them.

orion.clean_up has sample observations showing the problems.

Business Scenario: Data Clean Up

Listing of `orion.clean_up`

Product_ID	Product	Order_ID
21 02 002 00003	Sunfit Trunks, Blue	1231986335
21 02 002 00003	Luci Knit Mittens, Red	1232003930
21 02 002 00004	Luci Knit mittens, Blue	1232007693
21 02 002 00004	Sunfit Trunks, aqua	1232007700
21 02 002 00005	Sunfit Trunks, Yellow	1232087464
21 02 002 00005	Lucky Knit Mittens, Black	1232092527

- The **Product_ID** for mittens should have 5 instead of a 2 for the third group of numbers.
- `Luci` is a typo; the correct word is `Lucky`.
- **Product_ID** values should have no internal spaces.
- All words in the **Product** value should start with a capital letter.

Business Scenario – Desired Output

The **correct** data set shows what the data should be.

Listing of **correct**

Product_ID	Product	Order_ID
210200200003	Sunfit Trunks, Blue	1231986335
210200500003	Lucky Knit Mittens, Red	1232003930
210200500004	Lucky Knit Mittens, Blue	1232007693
210200200004	Sunfit Trunks, Aqua	1232007700
210200200005	Sunfit Trunks, Yellow	1232087464
210200500005	Lucky Knit Mittens, Black	1232092527

Data Clean Up – Step 1

The first step in creating the **correct** data set is to do the following:

- Find the observations with `Mittens` as part of the **Product** value.
- Change the middle characters of the **Product_ID** values for those observations.

The `FIND` and `SUBSTR` functions are useful for this.

The FIND Function

The FIND function searches a target string for a specified substring.

General form of the FIND function:

```
Position = FIND(string,substring<,modifiers,startpos>);
```

The FIND function returns a numeric value that is

- the starting position of the first occurrence of *substring* within *string*, if *substring* is found
- 0, if *substring* is not found.

The FIND Function

The FIND function searches a target *string* for a specified *substring*.

General form of the FIND function:

```
Position = FIND(string,substring<,modifiers,startpos>);
```

Modifiers can be

- I to indicate a case-insensitive search
- T to indicate to ignore trailing blanks in the string and substring values.

startpos indicates where in the *string* to start searching for the *substring*.

The FIND Function – Example

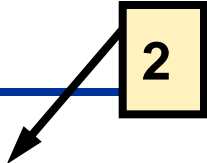
```
data find;  
  Text='AUSTRALIA, DENMARK, US';  
  Pos1=find(Text, 'US');  
  Pos2=find(Text, ' US');  
  Pos3=find(Text, 'us');  
  Pos4=find(Text, 'us', 'I');  
  Pos5=find(Text, 'us', 'I', 10);  
run;
```

PDV

Pos1
N 8

What value will SAS assign to **Pos1**?

The FIND Function – Example



```
data find;  
  Text='AUSTRALIA, DENMARK, US';  
  Pos1=find(Text, 'US');  
  Pos2=find(Text, ' US');  
  Pos3=find(Text, 'us');  
  Pos4=find(Text, 'us', 'I');  
  Pos5=find(Text, 'us', 'I', 10);  
run;
```

PDV

Pos1	
N	8
2	

The FIND Function – Example

20

```
data find;  
  Text='AUSTRALIA, DENMARK, US';  
  Pos1=find(Text, 'US');  
  Pos2=find(Text, ' US');  
  Pos3=find(Text, 'us');  
  Pos4=find(Text, 'us', 'I');  
  Pos5=find(Text, 'us', 'I', 10);  
run;
```

PDV

Pos1	Pos2
N 8	N 8
2	20

Poll

Quiz



5.10 Quiz

Complete the PDV for the values for **Pos3** and **Pos4**.

```
data find;  
  Text='AUSTRALIA, DENMARK, US';  
  Pos1=find(Text, 'US');  
  Pos2=find(Text, ' US');  
  Pos3=find(Text, 'us');  
  Pos4=find(Text, 'us', 'I');  
  Pos5=find(Text, 'us', 'I', 10);  
run;
```

PDV

Pos1	Pos2	Pos3	Pos4
N 8	N 8	N 8	N 8
2	20		

5.10 Quiz – Correct Answer

Complete the PDV for the values for **Pos3** and **Pos4**.

```
data find;  
  Text='AUSTRALIA, DENMARK, US';  
  Pos1=find(Text, 'US');  
  Pos2=find(Text, ' US');  
  Pos3=find(Text, 'us');  
  Pos4=find(Text, 'us', 'I');  
  Pos5=find(Text, 'us', 'I', 10);  
run;
```

PDV

Pos1	Pos2	Pos3	Pos4
N 8	N 8	N 8	N 8
2	20	0	2

The FIND Function – Example

21

```
data find;  
  Text='AUSTRALIA, DENMARK, US';  
  Pos1=find(Text, 'US');  
  Pos2=find(Text, ' US');  
  Pos3=find(Text, 'us');  
  Pos4=find(Text, 'us', 'I');  
  Pos5=find(Text, 'us', 'I', 10);  
run;
```

PDV

Pos1	Pos2	Pos3	Pos4	Pos5
N 8	N 8	N 8	N 8	N 8
2	20	0	2	21

The SUBSTR Function (Left Side)

This form of the SUBSTR function (left side of assignment statement) replaces characters in a character variable.

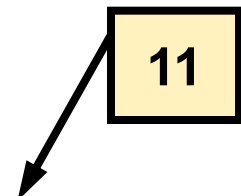
General form of the SUBSTR function (left side):

`SUBSTR(string,start<,length>)=value;`

<i>string</i>	specifies a character variable.
<i>start</i>	specifies the starting position to replace characters with the <i>value</i> .
<i>length</i>	<p>specifies the number of characters to replace in <i>string</i>. If omitted, all characters from the <i>start</i> position to the end of the <i>string</i> are replaced.</p> <p>The length value cannot be larger than the remaining length of <i>string</i> (including trailing blanks) after <i>start</i>.</p>

The SUBSTR Function (Left Side) – Example

Replace two characters starting at position 11.



A yellow box containing the number 11 has an arrow pointing to the start of the string 'Columbus, GA 43227' in the code below.

```
Location = 'Columbus, GA 43227';  
substr(Location,11,2)='OH';
```

PDV

Location		
\$ 18		
Columbus,	OH	43227

Data Clean Up – Step 1

Use the SUBSTR and FIND functions to change incorrect product IDs for mittens.

```
data correct;  
  set orion.clean_up;  
  if find(Product, 'Mittens', 'I') > 0 then do;  
    substr(Product_ID, 9, 1) = '5';  
  end;  
run;  
  
proc print data=correct noobs;  
run;
```

Data Clean Up – Step 1

PROC PRINT Output

Product_ID	Product	Order_ID
21 02 002 00003	Sunfit Trunks, Blue	1231986335
21 02 005 00003	Luci Knit Mittens, Red	1232003930
21 02 005 00004	Luci Knit mittens, blue	1232007693
21 02 002 00004	Sunfit Trunks, aqua	1232007700
21 02 002 00005	Sunfit Trunks, Yellow	1232087464
21 02 005 00005	Lucky Knit Mittens, Black	1232092527

The next step is to change the error `Luci` to `Lucky`.

The `TRANWRD` function is the best way to do this kind of change.

The TRANWRD Function

The TRANWRD function replaces or removes all occurrences of a given word (or a pattern of characters) within a character string.

General form for the TRANWRD function:

```
NewVar=TRANWRD(source,target,replacement);
```

<i>source</i>	specifies the source string that you want to change.
<i>target</i>	specifies the string searched for in <i>source</i> .
<i>replacement</i>	specifies the string that replaces <i>target</i> .

The TRANWRD Function – Details

General form for the TRANWRD function:

```
NewVar=TRANWRD(source,target,replacement);
```

These details apply when you use the TRANWRD function:

- The TRANWRD function does not remove trailing blanks from *target* or *replacement*.
- If *NewVar* was not previously defined, it is given a length of 200.
- If the target string is not found in the source, then no replacement occurs.

Data Clean Up – Step 2

Use the TRANWRD function to replace all occurrences of `Luci` with `Lucky`.

```
data correct;  
  set orion.clean_up;  
  if find(Product, 'Mittens', 'I') > 0 then do;  
    substr(Product_ID, 9, 1) = '5';  
    Product=Tranwrd(Product, 'Luci ', 'Lucky ');  
  end;  
run;  
  
proc print data=correct noobs;  
run;
```

Data Clean Up – Step 2

PROC PRINT Output

Product_ID	Product	Order_ID
21 02 002 00003	Sunfit Trunks, Blue	1231986335
21 02 005 00003	Lucky Knit Mittens, Red	1232003930
21 02 005 00004	Lucky Knit mittens, blue	1232007693
21 02 002 00004	Sunfit Trunks, aqua	1232007700
21 02 002 00005	Sunfit Trunks, Yellow	1232087464
21 02 005 00005	Lucky Knit Mittens, Black	1232092527

For step 3, removing the embedded blanks from **Product_ID** is easy with the COMPRESS function.

The COMPRESS Function

The COMPRESS function removes the characters listed in the *chars* argument from the *source*.

General form for the COMPRESS function:

```
NewVar=COMPRESS(source<,chars>);
```

If no *chars* are specified, the COMPRESS function removes all blanks from the *source*.

The COMPRESS Function

```
ID = '20 01-005 024' ;  
New_ID1=compress (ID) ;  
New_ID2=compress (ID, '-') ;  
New_ID3=compress (ID, ' -' ) ;
```

PDV

ID \$ 13	New_ID1 \$ 13
20 01-005 024	2001-005024

New_ID2 \$ 13	New_ID3 \$ 13
20 01005 024	2001005024

Other Functions That Remove Blanks

Function	Purpose
TRIM(<i>string</i>)	removes trailing blanks from a character string.
STRIP(<i>string</i>)	removes all leading and trailing blanks from a character string.
COMPBL(<i>string</i>)	removes multiple blanks from a character string by translating each occurrence of two or more consecutive blanks into a single blank.

Data Clean Up – Step 3

Use the COMPRESS and PROPCASE functions to eliminate blanks from **Product_ID** and ensure the proper case for **Product**.

```
data correct;  
  set orion.clean_up;  
  if find(Product,'Mittens','I') > 0 then do;  
    substr(Product_ID,9,1) = '5';  
    Product=tranwrd(Product,'Luci ','Lucky ');  
  end;  
  Product_ID = compress(Product_ID);  
  Product = propcase(Product);  
run;  
  
proc print data=correct noobs;  
run;
```

Data Clean Up – Step 3

PROC PRINT Output

Product_ID	Product	Order_ID
210200200003	Sunfit Trunks, Blue	1231986335
210200500003	Lucky Knit Mittens, Red	1232003930
210200500004	Lucky Knit Mittens, Blue	1232007693
210200200004	Sunfit Trunks, Aqua	1232007700
210200200005	Sunfit Trunks, Yellow	1232087464
210200500005	Lucky Knit Mittens, Black	1232092527

Chapter 5: Data Transformations

5.1 Introduction

5.2 Manipulating Character Values (Part 1)

5.3 Manipulating Character Values (Part 2)

5.4 Manipulating Numeric Values

5.5 Converting Variable Type

Objectives

- Use SAS functions to truncate numeric values.
- Use SAS functions to compute descriptive statistics of numeric values.

Truncation Functions

These functions truncate numeric values:

- ROUND
- CEIL
- FLOOR
- INT

The ROUND Function

The ROUND function returns a value rounded to the nearest multiple of the round-off unit.

General form of the ROUND function:

```
NewVar=ROUND(argument<,round-off-unit>);
```

<i>argument</i>	is a number or numeric expression.
<i>round-off-unit</i>	is numeric and positive. If <i>round-off-unit</i> is not provided, <i>argument</i> is rounded to the nearest integer.

The ROUND Function – Example

```
data truncate;  
    NewVar1=round(12.12) ;  
    NewVar2=round(42.65,.1) ;  
    NewVar3=round(-6.478) ;  
    NewVar4=round(96.47,10) ;  
run;
```

PDV

NewVar1	NewVar2	NewVar3	NewVar4
N 8	N 8	N 8	N 8
12	42.7	-6	100

The ROUND Function – Example

```
data truncate;  
    NewVar5=round(12.69,.25);  
    NewVar6=round(42.65,.5);  
run;
```

Round to the nearest multiple of .25

PDV

NewVar5	NewVar6
N 8	N 8
12.75	.

The ROUND Function – Example

```
data truncate;  
    NewVar5=round(12.69,.25);  
    NewVar6=round(42.65,.5);  
run;
```

Round to the nearest multiple of .5

PDV

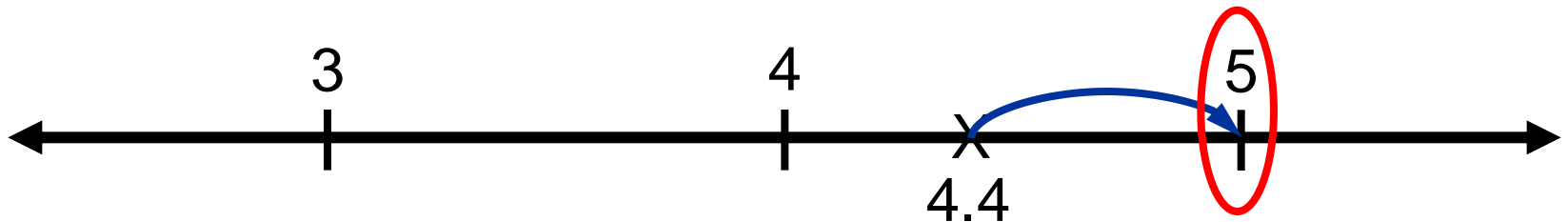
NewVar5	NewVar6
N 8	N 8
12.75	42.5

The CEIL Function

The CEIL function returns the smallest integer greater than or equal to the argument.

General form of the CEIL function:

```
NewVar=CEIL(argument);
```



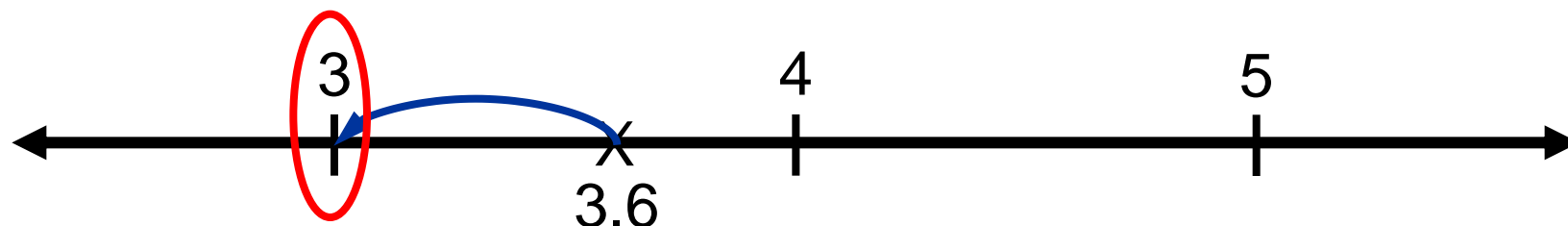
```
x=ceil(4.4);
```


The FLOOR Function

The FLOOR function returns the greatest integer less than or equal to the argument.

General form of the FLOOR function:

```
NewVar=FLOOR(argument);
```



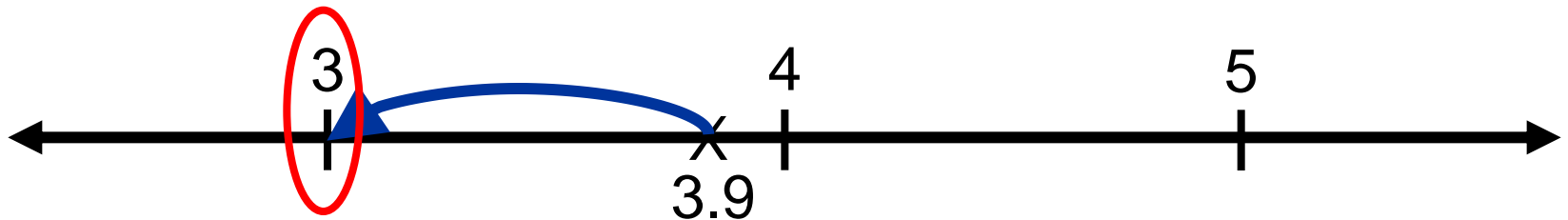
```
y=floor(3.6);
```

The INT Function

The INT function returns the integer portion of the argument.

General form of the INT function:

```
NewVar=INT(argument);
```



```
z=int (3.9) ;
```

Truncation Functions – Example

```
data truncate;  
    Var1=6.478;  
    CeilVar1=ceil(Var1);  
    FloorVar1=floor(Var1);  
    IntVar1=int(Var1);  
run;
```

PDV

Var1	CeilVar1	FloorVar1	IntVar1
6.478	7	6	6

Poll 

Quiz

Setup for the Poll

In this program, the values returned from the FLOOR and INT functions are the same.

```
data truncate;  
  Var1=6.478;  
  CeilVar1=ceil(Var1);  
  FloorVar1=floor(Var1);  
  IntVar1=int(Var1);  
run;
```

PDV

Var1	CeilVar1	FloorVar1	IntVar1
6.478	7	6	6

5.11 Poll

Given the same value as an argument, do the INT and the FLOOR functions always return the same result?

- ☐ Yes
- ☐ No

5.11 Poll – Correct Answer

Given the same value as an argument, do the INT and the FLOOR functions always return the same result?

☐ Yes

☒ No



The INT and the FLOOR functions give different results if the argument value is negative.

Truncation Functions

Compare the values from the CEIL, FLOOR, and INT functions with a negative argument.

```
data truncate;  
  Var1=-6.478;  
  CeilVar1=ceil(Var1);  
  FloorVar1=floor(Var1);  
  IntVar1=int(Var1);  
run;
```

PDV

Var1	CeilVar1	FloorVar1	IntVar1
-6.478	-6	-7	-6

Business Scenario – Donation Statistics

Create a new data set, **donation_stats**, based on the information in **orion.employee_donations**.

For each employee, calculate the following:

- the employee's total donation for the year.
- the average donation for the quarters in which the employee made a donation. Round the average to the nearest dollar.
- the number of quarters in which the employee made a donation.

Business Scenario – Input and Desired Output

Partial Listing of `orion.employee_donations`

Obs	Employee_ID	Qtr1	Qtr2	Qtr3	Qtr4
1	120265	.	.	.	25
2	120267	15	15	15	15
3	120269	20	20	20	20
4	120270	20	10	5	.

Partial Listing of `donation_stats`

Employee_ID	Total	Avg QT	Num Qt
120265	25	25	1
120267	60	15	4
120269	80	20	4
120270	35	12	3

Descriptive Statistics Functions

Using descriptive statistic functions will be the easiest way to calculate the values needed for **donation_stats**.

Function	Returns
SUM	Sum of arguments.
MEAN	Average of arguments.
MIN	Smallest value from arguments.
MAX	Largest value from arguments.
N	Count of non-missing arguments.
NMISS	Count of missing numeric arguments.
CMISS	Count of missing numeric or character arguments.

Descriptive Statistics Functions

These functions all share the same general syntax:

function-name(argument-1,argument-2,...,argument-n)

- *argument-1* through *argument-n* are numeric.
- An argument can be a variable list, which is preceded by OF.
- The SUM, MEAN, MAX, and MIN functions ignore missing values in their arguments.

Descriptive Statistics Functions

```
data descript;  
  Var1=12;  
  Var2=. ;  
  Var3=7;  
  Var4=5;  
  SumVars=sum(Var1,Var2,Var3,Var4) ;  
  AvgVars=mean(of Var1-Var4) ;  
  MissVars=cmiss(of Var1-Var4) ;  
run;
```

PDV

Var1	Var2	Var3	Var4
12	.	7	5

SumVars	AvgVars	MissVars
24	.	.

Descriptive Statistics Functions

```
data descript;  
  Var1=12;  
  Var2=. ;  
  Var3=7;  
  Var4=5;  
  SumVars=sum(Var1,Var2,Var3,Var4) ;  
  AvgVars=mean(of Var1-Var4) ;  
  MissVars=cmiss(of Var1-Var4) ;  
run;
```

PDV

Var1	Var2	Var3	Var4
12	.	7	5

SumVars	AvgVars	MissVars
24	8	.

Descriptive Statistics Functions

```
data descript;  
  Var1=12;  
  Var2=. ;  
  Var3=7;  
  Var4=5;  
  SumVars=sum(Var1,Var2,Var3,Var4) ;  
  AvgVars=mean(of Var1-Var4) ;  
  MissVars=cmiss(of Var1-Var4) ;  
run;
```

PDV

Var1	Var2	Var3	Var4
12	.	7	5

SumVars	AvgVars	MissVars
24	8	1

Poll 

Quiz

5.12 Multiple Choice Poll

Ord1, **Ord2**, and **Ord3** are variables that contain the sale amounts of the last three orders from a customer. Which of the following expressions can calculate the total of the two largest orders?

- a. `sum(max(of Ord1-Ord3), max(of Ord1-Ord3))`
- b. `sum(of Ord1-Ord3) - min(of Ord1-Ord3)`
- c. `max(of Ord1-Ord3) + min(of Ord1-Ord3)`
- d. None of the above

5.12 Multiple Choice Poll – Correct Answer

Ord1, **Ord2**, and **Ord3** are variables that contain the sale amounts of the last three orders from a customer. Which of the following expressions can calculate the total of the two largest orders?

- a. `sum(max(of Ord1-Ord3), max(of Ord1-Ord3))`
- ☒ b. `sum(of Ord1-Ord3) - min(of Ord1-Ord3)`
- c. `max(of Ord1-Ord3) + min(of Ord1-Ord3)`
- d. None of the above

Adding the amount from all three orders and then subtracting the amount of the smallest order leaves the sum of the two largest orders.

Business Scenario – Complete

Use the SUM, MEAN, and N functions to calculate the donation statistics.

```
data donation_stats;  
  set orion.employee_donations;  
  keep Employee_ID Total AvgQT NumQT;  
  Total = sum(of Qtr1-Qtr4);  
  AvgQT = round(Mean(of Qtr1-Qtr4));  
  NumQt = n(of Qtr1-Qtr4);  
run;
```

Business Scenario – Complete

```
proc print data=donation_stats noobs;  
    var Employee_ID Total AvgQt NumQt;  
run;
```

Partial PROC PRINT Output

Employee_ID	Total	Avg QT	Num Qt
120265	25	25	1
120267	60	15	4
120269	80	20	4
120270	35	12	3

Chapter 5: Data Transformations

5.1 Introduction

5.2 Manipulating Character Values (Part 1)

5.3 Manipulating Character Values (Part 2)

5.4 Manipulating Numeric Values

5.5 Converting Variable Type

Objectives

- Explain the automatic conversion that SAS uses to convert values between data types.
- Explicitly convert values between data types.

Business Scenario – Convert HR Data

Orion Star recently acquired a small marketing firm and needs to convert the firm's personnel data into a data set that can be easily transferred into Orion's HR system.

The data set **orion.convert** has a sample of the marketing firm's personnel data.

Listing of **orion.convert**

ID \$ 5	GrossPay \$ 6	Code N 8	Mobile \$ 8	Hired \$ 10
36	52,000	303	393-0956	04/13/2004
48	32,000	919	770-8292	08/25/2006
52	49,000	301	449-5239	06/08/2005

Business Scenario – Desired Output

Store the converted personnel data in **hrdata**.

Listing of **hrdata**

EmpID N 8	GrossPay N 8	Bonus N 8	Phone \$ 14	HireDate N 8
11036	52000	5200	(303) 393-0956	16174
11048	32000	3200	(919) 770-8292	17038
11052	49000	4900	(301) 449-5239	16595

- 11000 is added to **ID** to create **EmpID**. (This avoids conflicts with existing Orion Star Employee IDs.)
- **GrossPay** is the only variable name being kept.
- **Bonus** is a retention bonus and is 10% of **GrossPay**.

Business Scenario – Desired Output

Store the converted personnel data in **hrdata**.

Listing of **hrdata**

EmpID N 8	GrossPay N 8	Bonus N 8	Phone \$ 14	HireDate N 8
11036	52000	5200	(303) 393-0956	16174
11048	32000	3200	(919) 770-8292	17038
11052	49000	4900	(301) 449-5239	16595

- **Phone** is a combination of **Code** and **Mobile**.
- **HireDate** is a SAS date value.

Data Conversion

For this business scenario, SAS needs to convert one data type to another.

Character-to-Numeric

- The character values in **ID**, **GrossPay**, and **Hired** need to be transformed into numeric values.

Numeric-to-Character

- The numeric values in **Code** need to be transformed into character values before being concatenated with the values in **Mobile**.

Data Conversion

Data types can be converted two ways:

- *automatically* by allowing SAS to do it for you.
- *explicitly* with these functions:

INPUT	character-to-numeric conversion
PUT	numeric-to-character conversion

Automatic Character-to-Numeric Conversion

What will happen when the character values of **ID** are used in an arithmetic expression?

```
data hrdata;  
    keep EmpID;  
    set orion.convert;  
    EmpID = ID + 11000;  
run;
```

Automatic Character-to-Numeric Conversion

Partial Log

```
28  data hrdata;  
29      keep EmpID;  
30      set orion.convert;  
31      EmpID = ID + 11000;  
32  run;
```

NOTE: Character values have been converted to numeric values at the places given by:

(Line):(Column).

31:11

NOTE: There were 3 observations read from the data set ORION.CONVERT.

NOTE: The data set WORK.HRDATA has 3 observations and 1 variables.

Automatic Character-to-Numeric Conversion

```
proc print data=hrdata noobs;  
run;
```

PROC PRINT Output

EmpID

11036

11048

11052

The automatic conversion worked great for **ID**. Now see what happens with **GrossPay**.

Automatic Character-to-Numeric Conversion

What happens when the character values of **GrossPay** are used in an arithmetic expression?

```
data hrdata;  
    keep GrossPay Bonus;  
    set orion.convert;  
    Bonus = GrossPay * .10;  
run;
```

Automatic Character-to-Numeric Conversion

```
proc print data=hrdata noobs;  
run;
```

PROC PRINT Output

Gross Pay	Bonus
52,000	.
32,000	.
49,000	.

Why did the automatic conversion not work for the values of **GrossPay**?

Automatic Character-to-Numeric Conversion

SAS automatically converts a character value to a numeric value when the character value is used in a numeric context, such as the following:

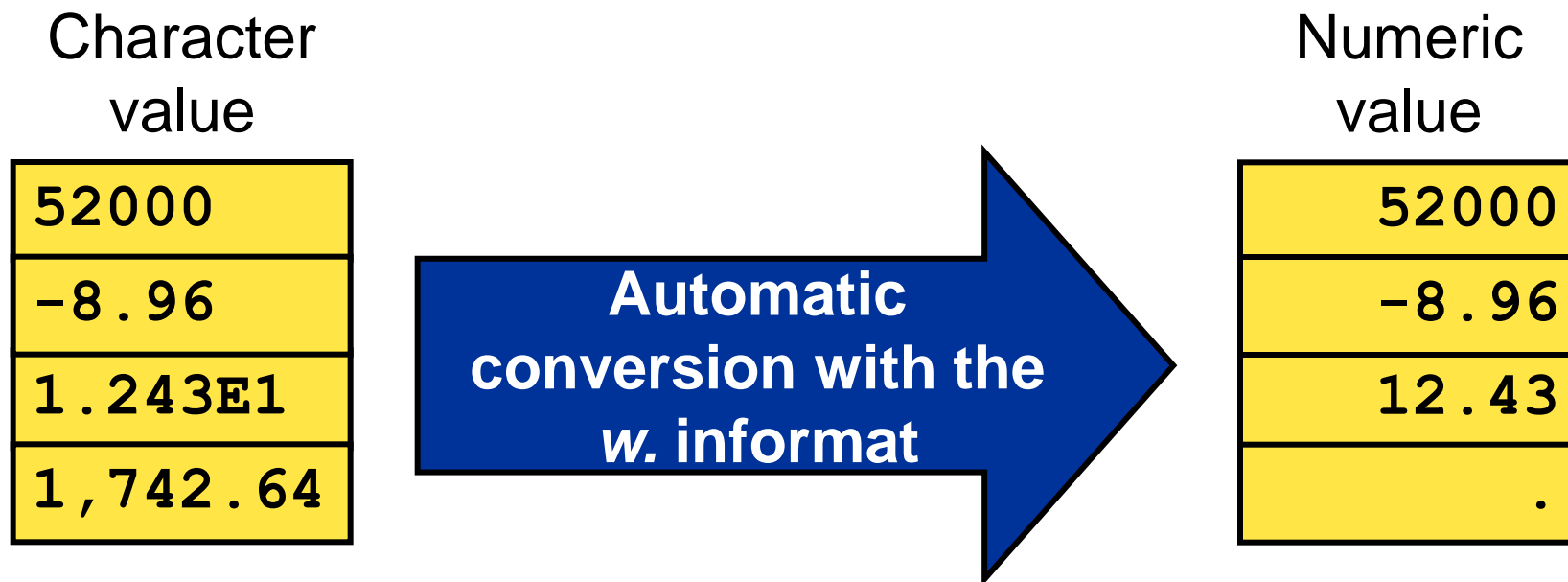
- assignment to a numeric variable
- an arithmetic operation
- logical comparison with a numeric value
- a function that takes numeric arguments

Automatic Character-to-Numeric Conversion

The automatic conversion

- uses the w. informat
- converts only standard numeric data (digits exclusively or digits with optional decimal point and/or leading sign and/or E-notation).
- produces a numeric missing value from a character value that does not conform to standard numeric notation

Automatic Character-to-Numeric Conversion



- The values in **GrossPay** contained commas, which could not be converted by the w. informat, so **GrossPay** was assigned a missing value.
- To explicitly convert the values in **GrossPay**, use the INPUT function.

The INPUT Function

The INPUT function returns the value produced when the source is read with a specified informat.

General form of the INPUT function:

```
NumVar=INPUT(source,informat);
```

<i>source</i>	contains a SAS character expression
<i>informat</i>	is the SAS informat to apply to the <i>source</i> .

 No conversion messages are written to the log by the INPUT function.

The INPUT Function – Example

This DATA step shows examples of the INPUT function.

```
data conversions;  
  CVar1='32000';  
  CVar2='32.000';  
  CVar3='03may2008';  
  CVar4='030508';  
  NVar1=input(CVar1,5.);  
  NVar2=input(CVar2,commax6.);  
  NVar3=input(CVar3,date9.);  
  NVar4=input(CVar4,ddmmyy6.);  
run;
```

The INPUT Function – Example

```
proc contents data=conversions;  
run;
```

Partial PROC CONTENTS Output

Alphabetic List of Variables and Attributes

#	Variable	Type	Len
1	CVar1	Char	5
2	CVar2	Char	6
3	CVar3	Char	9
4	CVar4	Char	6
5	NVar1	Num	8
6	NVar2	Num	8
7	NVar3	Num	8
8	NVar4	Num	8

The INPUT Function – Example

```
proc print data=conversions noobs;  
run;
```

PROC PRINT Output

CVar1	CVar2	CVar3	CVar4	NVar1	NVar2	NVar3	NVar4
32000	32.000	03may2008	030508	32000	32000	17655	17655

Poll

Quiz



5.13 Quiz

Fill in the missing expression in the DATA step below. The expression should calculate **TotalValue** by multiplying **SharePrice** by **MyShares**.

```
data Input_Quiz;  
    SharePrice = "$130.25";  
    MyShares = 125;  
    TotalValue = ?  
  
run;
```

5.13 Quiz – Correct Answer

Fill in the missing expression in the DATA step below. The expression should calculate **TotalValue** by multiplying **SharePrice** by **MyShares**.

```
data Input_Quiz;  
    SharePrice = "$130.25";  
    MyShares = 125;  
    TotalValue = input(SharePrice, comma7.) *  
                  MyShares;  
run;
```

The COMMAw.d informat reads numeric values and removes embedded commas, blanks, dollar signs, percent signs, hyphens, and close parentheses from the input data. The COMMAw.d informat converts an open parenthesis at the beginning of a field to a minus sign.

Explicit Character-to-Numeric Conversion

Continue with the business scenario by creating the variables **EmpID**, **Bonus**, and **HireDate**.

Use the INPUT function to explicitly convert character values to numeric.

```
data hrdata;  
  keep EmpID GrossPay Bonus HireDate;  
  set orion.convert;  
  EmpID = input(ID,5.)+11000;  
  Bonus = input(GrossPay,comma6.)*.10;  
  HireDate = input(Hired,mmdyy10.);  
run;
```


Explicit Character-to-Numeric Conversion

```
proc print data=hrdata noobs;  
    var EmpID GrossPay Bonus HireDate;  
run;
```

PROC PRINT Output

SAS date values

EmpID	Gross Pay	Bonus	Hire Date
11036	52,000	5200	16174
11048	32,000	3200	17038
11052	49,000	4900	16595



Explicit Character-to-Numeric Conversion

```
proc print data=hrdata noobs;  
  var EmpID GrossPay Bonus HireDate;  
  format HireDate mmddyy10.;  
run;
```

PROC PRINT Output

EmpID	Gross Pay	Bonus	HireDate
11036	52,000	5200	04/13/2004
11048	32,000	3200	08/25/2006
11052	49,000	4900	06/08/2005

What data type is **GrossPay**?

Converting a Variable to Another Data Type

```
proc contents data=hrdata;  
run;
```

Partial PROC CONTENTS Output

Alphabetic List of Variables and Attributes

#	Variable	Type	Len
3	Bonus	Num	8
2	EmpID	Num	8
1	GrossPay	Char	6
4	HireDate	Num	8

How can you convert **GrossPay** to a numeric variable with the same name?

Poll

Quiz



5.14 Quiz

Will this statement convert **GrossPay** to numeric?

```
GrossPay=input (GrossPay,comma6. ) ;
```

Open and run the program **p205a02**. Did **GrossPay** become a numeric variable?

5.14 Quiz – Correct Answer

Will this statement convert **GrossPay** to numeric?

```
GrossPay=input (GrossPay,comma6. ) ;
```

Open and run the program **p205a02**. Did **GrossPay** become a numeric variable?

No, GrossPay remained a character variable.

Converting a Variable to Another Data Type

```
GrossPay=input (GrossPay, comma6. ) ;
```



This assignment statement does **not** change **GrossPay** from a character variable to a numeric variable.

A variable is character or numeric. After the variable's type is established, it cannot be changed.

By following three steps, you can create a new variable with the same name and a different type.

Converting a Variable to Another Data Type

Step 1: Use the RENAME= data set option to rename the variable that you want to convert.

```
data hrdata;  
    set orion.convert(rename=(GrossPay=  
                             CharGross)) ;  
run;
```

General form of the RENAME data set option:

```
SAS-data-set(RENAME=(old-name=new-name))
```

Converting a Variable to Another Data Type

Step 2: Use the INPUT function in an assignment statement to create a new variable with the original name of the variable that you renamed.

```
data hrdata;  
    set orion.convert(rename=(GrossPay=  
                                CharGross)) ;  
    GrossPay=input(CharGross,comma6.) ;  
run;
```

Converting a Variable to Another Data Type

Step 3: Use a DROP= data set option in the DATA statement to exclude the original variable from the output SAS data set.

```
data hrdata(drop=CharGross) ;  
    set orion.convert(rename=(GrossPay=  
                             CharGross)) ;  
    GrossPay=input(CharGross,comma6.) ;  
run;
```

The compilation for this program shows the PDV being created with a numeric **GrossPay** variable.

Converting a Variable: Compilation

```
data hrdata(drop=CharGross) ;  
    set orion.convert(rename=(GrossPay=  
                             CharGross)) ;  
    GrossPay=input(CharGross,comma6.) ;  
run;
```

Partial PDV

ID	CharGross	Hired
\$ 5	\$ 6	\$ 7

Converting a Variable: Compilation

```
data hrdata(drop=CharGross) ;  
    set orion.convert(rename=(GrossPay=  
                                CharGross)) ;  
    GrossPay=input(CharGross,comma6.) ;  
run ;
```

Partial PDV

ID	CharGross	Hired	GrossPay
\$ 5	\$ 6	\$ 7	N 8

Converting a Variable: Compilation

```
data hrdata(drop=CharGross) ;  
    set orion.convert(rename=(GrossPay=  
                                CharGross)) ;  
    GrossPay=input(CharGross,comma6.) ;  
run;
```

Partial PDV

ID	CharGross	Hired	GrossPay
\$ 5	\$ 6	\$ 7	N 8