

SAS Advanced Programming

SAP3 SAS Advanced Programming Techniques

SAP305 Creating Picture Formats with the FORMAT Procedure, Creating Functions with the FCMP Procedure

Libname.sas

```
%let path=~ /EPG3M6;
```

```
%let pathout=&path/output;
```

```
libname pg3 "&path/data" filelockwait=20;
```

* FILELOCKWAIT=20 specifies SAS will wait up to 20 seconds

for a locked file to become available. Use this option

to avoid a lock error when using the FCMP procedure. */

create
format

```
proc format;  
  value $scale 'A' = 'Excellent'  
              'B' = 'Good'  
              'C' = 'Fair'  
              other = 'Miscoded';  
  value age_range low-13 = 'Pre-teen'  
                 12-17  = 'Teen'  
                 18-high = 'Adult';  
run;
```

apply
format

```
proc print data=work.students noobs;  
  var Name Grade Age;  
  format Grade $scale. Age age_range.;  
run;
```

PROC FORMAT;

PICTURE *format-name* (*format-options*)

value-or-range-1 = *'template-value'* (*template-options*)

value-or-range-2 = *'template-value'* (*template-options*)

...;

RUN;

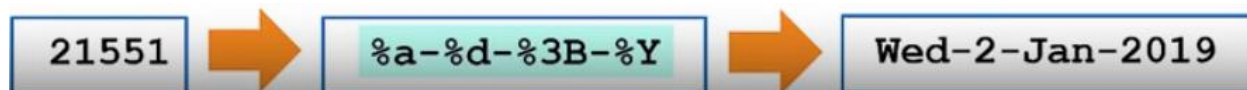
```

PROC FORMAT;
    PICTURE format-name (DEFAULT=length)
        value-or-range-1 = 'directives'
        (DATATYPE=DATE | TIME | DATETIME);
RUN;

```

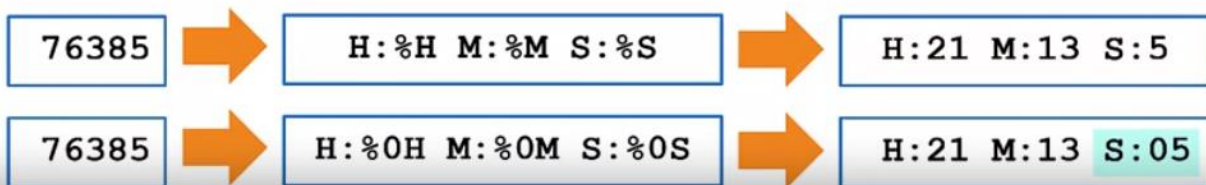
%A	Weekday name (full)	Wednesday
%a	Weekday name (first three letters)	Wed
%d	Day of month (one or two digits)	2
%0d	Day of month (two digits)	02
%B	Month name (full)	January
%3B	Month name (first three letters)	Jan
%m	Month number (one or two digits)	1
%0m	Month number (two digits)	01
%Y	Year (four digits)	2019
%0y	Year (two digits)	19

NOTE: Both SAS On Demand for Academics and SAS University Edition sessions use UTF-8 encoding, a common double-byte character set (DBCS). In SAS DBCS sessions, the %B directive does not permit limiting text length. Use the %b directive instead.



%H or %0H	Hour, 24-hour clock (one or two digits)	21
%I or %0I	Hour, 12-hour clock (one or two digits)	9 or 09
%M or %0M	Minute (one or two digits)	13
%S or %0S	Second (one or two digits)	5 or 05
%p	AM or PM	PM

To add a leading zero before a single digit, insert a 0 before the directive.



```

proc format;
  picture MyDate (default=15)
    low-high = '%a-%d-%3B-%Y'
    (datatype=date);

  picture MyTime (default=14)
    low-high = 'H:%0H M:%0M S:%0S'
    (datatype=time);
run;

```

StartDate
Wed-2-Jan-2019
StartTime
H:21 M:13 S:05

*****,

* Specifying a Template for Datetime Values *;

*****,

```

proc format;

  value $genfmt 'F' = 'Female'

    'M' = 'Male'

    other = 'Miscoded';

  value hrange low-<58 = 'Below Average'

    58-60 = 'Average'

    60<-high = 'Above Average';

run;

```

```
proc print data=pg3.class_birthdate noobs;
  var Name Gender Height;
  format Gender $genfmt. Height hrange.;
run;
```

```
proc format;
  picture MyDate (default=15)
    low-high = '%a-%d-%3B-%Y'
    (datatype=date);
  picture MyTime (default=14)
    low-high = 'H:%0H M:%0M S:%0S'
    (datatype=time);
  picture MyDateTime (default=24)
    low-high = '%Y.%0m.%0d @ %l:%0M:%0S %p'
    (datatype=datetime);
run;
```

```
data work.datetimetable;
  StartDate='02JAN2019'd;
  StartTime='21:13:5't;
  StartDateTime='02JAN2019:21:13:5'dt;
  format StartDate MyDate. StartTime MyTime. StartDateTime MyDateTime.;
run;
```

```
proc print data=work.datetimetable;
run;
```

```
proc format;
```

```

picture MyDate (default=15)

    low-'31DEC1999'd = '%3B-%Y' (datatype=date)

    '01JAN2000'd-high = '%a-%d-%3B-%Y' (datatype=date);

run;

data work.datetimetable;

    StartDate='02JAN2019'd; output;

    StartDate='15MAY1995'd; output;

    format StartDate MyDate.;

run;

proc print data=work.datetimetable;

run;

*****

* Demo                                *;

* 1) Highlight and run the TITLE statements and the PROC    *;

*   PRINT step. Notice the formatted value of ISO_time.    *;

* 2) Notice the syntax of the PROC FORMAT step. The date and *;

*   time directives are specifying a one- or two-digit day  *;

*   of month, a three-letter proper case month name, a     *;

*   four-digit year, a colon, a one- or two-digit 24-clock  *;

*   hour, and the letter H (for example, 7Aug1980:18H).    *;

* 3) Modify the FORMAT statement in the PROC PRINT step to  *;

*   use the custom format created in the PROC FORMAT step.  *;

*   format ISO_time dateh.;                                *;

* 4) Highlight and run the demo program. Notice that the    *;

*   ISO_time values are not formatted properly because SAS   *;

*   did not recognize the date and time directives.          *;

```

```

* 5) Add the DATATYPE= option to the PICTURE statement after *;
*   the directives.                *;
*   picture dateh                  *;
*       low-high='%d%3B%Y:%HH'      *;
*       (datatype=datetime);        *;
* 6) Highlight and run the program. Notice that the ISO_time *;
*   values are formatted but are being truncated. A length *;
*   of 11 is being used by default because that is the *;
*   length of the template.        *;
* 7) Add the DEFAULT= option to the PICTURE statement after *;
*   the name of the format.        *;
*   picture dateh (default=13)      *;
*       low-high='%d%3B%Y:%HH'      *;
*       (datatype=datetime);        *;
* 8) Highlight and run the demo program. Verify the formatted *;
*   values of ISO_time.            *;
*****

```

```

title 'Storms with Category 5 Winds';
proc print data=pg3.storm_detail noobs;
    var Name ISO_time Wind Pressure Region;
    where Wind>155;
    format ISO_time datetime.;
run;
title;

```

Storms with Category 5 Winds

Name	ISO_time	Wind	Pressure	Region
ALLEN	07AUG80:18:00:00	165	899	Atlantic
GILBERT	14SEP88:00:00:00	160	888	Atlantic
LINDA	12SEP97:06:00:00	160	902	Pacific
WILMA	19OCT05:12:00:00	160	882	Atlantic
PATRICIA	23OCT15:06:00:00	180	886	Pacific
PATRICIA	23OCT15:12:00:00	185	872	Pacific
PATRICIA	23OCT15:18:00:00	180	878	Pacific

```
proc format;
```

```
    picture dateh
```

```
        low-high='%d%3B%Y:%HH' (datatype=datetime);
```

```
run;
```

```
title 'Storms with Category 5 Winds';
```

```
proc print data=pg3.storm_detail noobs;
```

```
    var Name ISO_time Wind Pressure Region;
```

```
    where Wind>155;
```

```
    format ISO_time dateh.;
```

```
run;
```

```
title;
```

Storms with Category 5 Winds

Name	ISO_time	Wind	Pressure	Region
ALLEN	7August1980	165	899	Atlantic
GILBERT	14September	160	888	Atlantic
LINDA	12September	160	902	Pacific
WILMA	19October20	160	882	Atlantic
PATRICIA	23October20	180	886	Pacific
PATRICIA	23October20	185	872	Pacific
PATRICIA	23October20	180	878	Pacific

```
proc format;
```

```
    picture dateh (default=13)
```

```
        low-high='%d%3b%Y:%HH' (datatype=datetime);
```

```
run;
```

```
title 'Storms with Category 5 Winds';
```

```
proc print data=pg3.storm_detail noobs;
```

```
var Name ISO_time Wind Pressure Region;
```

```
where Wind>155;
```

```
format ISO_time dateh.;
```

```
run;
```

```
title;
```

Storms with Category 5 Winds				
Name	ISO_time	Wind	Pressure	Region
ALLEN	7Aug1980:18H	165	899	Atlantic
GILBERT	14Sep1988:0H	160	888	Atlantic
LINDA	12Sep1997:6H	160	902	Pacific
WILMA	19Oct2005:12H	160	882	Atlantic
PATRICIA	23Oct2015:6H	180	886	Pacific
PATRICIA	23Oct2015:12H	185	872	Pacific
PATRICIA	23Oct2015:18H	180	878	Pacific

```
*****,
```

```
* Activity 5.02 *;
```

```
* 1) Run the program and notice the formatting of *;
```

```
* BeginDate. *;
```

```
* 2) Add a PICTURE statement to create another date *;
```

```
* format. *;
```

```
* - Name the format MyMonth. *;
```

```
* - Be sure to specify a default length. *;
```

```
* - All date values should have a format layout *;
```

```
* similar to November of 2017. *;
```

```
* Hint: Use the %B and %Y directives. *;
```

```
* - Be sure to declare the data type. *;
```

```
* 3) Modify the FORMAT statement in the PROC FREQ step *;
```



```

*   to use the new format. Run the program.           *;
*   Which month has the most tornados?               *;
*****;

```

```

proc format;
  picture MyDate (default=14)
    low-high = '%A-%Y' (datatype=date);
  /* add PICTURE statement here */

```

```
run;
```

```

title 'Frequency of 2017 US Tornadoes';
proc freq data=pg3.tornado_2017 order=freq;
  table BeginDate;
  format BeginDate MyMonth.;
run;
title;

```

Frequency of 2017 US Tornadoes				
The FREQ Procedure				
BeginDate				
BeginDate	Frequency	Percent	Cumulative Frequency	Cumulative Percent
20840	74	14.80	74	14.80
20884	66	13.20	140	28.00
20879	44	8.80	184	36.80
20821	42	8.40	226	45.20
20878	38	7.60	264	52.80
20841	29	5.80	293	58.60
20887	20	4.00	313	62.60

```

proc format;
  picture MyDate (default=14)

```

```

low-high = '%A-%Y' (datatype=date);
/* add PICTURE statement here */
picture MyMonth (default=17)
low-high = '%B of %Y' (datatype=date);

run;

title 'Frequency of 2017 US Tornadoes';
proc freq data=pg3.tornado_2017 order=freq;
table BeginDate;
format BeginDate MyMonth.;
run;
title;

```

Frequency of 2017 US Tornadoes				
The FREQ Procedure				
BeginDate				
BeginDate	Frequency	Percent	Cumulative Frequency	Cumulative Percent
March of 2017	201	40.20	201	40.20
January of 2017	170	34.00	371	74.20
February of 2017	83	16.60	454	90.80
April of 2017	16	3.20	470	94.00
May of 2017	12	2.40	482	96.40
November of 2017	4	0.80	486	97.20
July of 2017	3	0.60	489	97.80
June of 2017	3	0.60	492	98.40
October of 2017	3	0.60	495	99.00
August of 2017	2	0.40	497	99.40
December of 2017	2	0.40	499	99.80
September of 2017	1	0.20	500	100.00

```
PROC FORMAT;
  PICTURE format-name (ROUND DEFAULT=length)
           value-or-range-1 = 'digit selectors with non-numeric characters'
           (MULT | MULTIPLIER=n PREFIX='prefix') ;
RUN;
```

Digit Selectors

'009.9'

- numeric characters, 0 through 9
- Nonzero digit selectors display leading zeros.
- A digit selector of 0 does not print leading zeros.

	A '999'	B '099'	C '009'
54	054	54	54
7	007	07	7

- 1 Apply the multiplier to the numeric value.

```
picture mypct (round) low-high='009.9%' (multiplier=10);
```

Initial Value	multiplied by 10
0.3165	3.165
4.1139	41.139
6.656	66.56
11.7089	117.089
18.9873	189.873
100	1000

- 2 If specified, round the numeric value to the nearest integer.

```
picture mypct (round) low-high='009.9%' (multiplier=10);
```

Initial Value	multiplied by 10	without ROUND	with ROUND
0.3165	3.165	3	3
4.1139	41.139	41	41
6.656	66.56	66	67
11.7089	117.089	117	117
18.9873	189.873	189	190
100	1000	1000	1000

3

Place the numeric value into the template beginning on the *right*.

```
picture mypct (round) low-high='009.9%' (multiplier=10);
```

Initial Value	multiplied by 10	without ROUND	with ROUND	Final Value with ROUND
0.3165	3.165	3	3	0.3%
4.1139	41.139	41	41	4.1%
6.656	66.56	66	67	6.7%
11.7089	117.089	117	117	11.7%
18.9873	189.873	189	190	19.0%
100	1000	1000	1000	100.0%

Custom Large Number

```
picture dollar_KM(round default=7)
    low-<1000='009' (prefix='$' mult=1)
    1000-<1000000='009.9K' (prefix='$' mult=.01)
    1000000-high='009.9M' (prefix='$' mult=.00001);
```

Unformatted Values	Multiplier	Multiplied Values	Multiplied Values with Round	Formatted Values
23	1.00000	23.00000	23	\$23
234	1.00000	234.00000	234	\$234
2345	0.01000	23.45000	23	\$2.3K
23456	0.01000	234.56000	235	\$23.5K
234567	0.01000	2345.67000	2346	\$234.6K
2345678	0.00001	23.45678	23	\$2.3M
23456789	0.00001	234.56789	235	\$23.5M
234567890	0.00001	2345.67890	2346	\$234.6M

```
*****,
```

```
* Specifying a Template for Large Numbers *;
```

```
*****,
```

```
proc freq data=pg3.tornado_2017 noprint;
```

```
    where PropertyDamage>0;
```

```

tables State / out=work.tornado_2017_pct;

run;

proc format;

  *picture mypct low-high='009.9%' (multiplier=10);

  picture mypct (round) low-high='009.9%' (multiplier=10);

run;

```

```

title1 '2017 US Tornadoes';

title2 'with Property Damage by State';

proc print data=work.tornado_2017_pct noobs;

  *format Percent percent8.2;

  format Percent mypct.;

run;

title;

```

```

data sample;

  input UnformattedValues Multiplier;

  MultiValues=UnformattedValues*Multiplier;

  MultiValuesRound=round(MultiValues);

  FormattedValues=UnformattedValues;

  label UnformattedValues='Unformatted Values'

        MultiValues='Multiplied Values'

        MultiValuesRound='Multiplied Values*with Round'

        FormattedValues='Formatted Values';

  datalines;

23 1

234 1

2345 .01

```

```

23456 .01
234567 .01
2345678 .00001
23456789 .00001
234567890 .00001
;
run;

proc format;
  picture dollar_KM (round default=7)
    low-<1000='009' (prefix='$' mult=1)
    1000-<1000000='009.9K' (prefix='$' mult=.01)
    1000000-high='009.9M' (prefix='$' mult=.00001);
run;

proc print data=sample split='*' noobs style(data)={cellwidth=1.5in};
  var UnformattedValues Multiplier MultiValues
    MultiValuesRound FormattedValues;
  format MultiValues 12.5
    FormattedValues dollar_KM.;
run;

*****
* Demo *;
* 1) Highlight and run the PROC SORT and PROC PRINT steps. *;
* Notice the formatted value of PropertyDamage. *;
* 2) Notice the syntax of the PROC FORMAT step. The number of *;
* digits for large numbers will be reduced. *;
* 3) Modify the FORMAT statement in the PROC PRINT step to *;

```

```

* use the custom format created in the PROC FORMAT step.  *;
* format PropertyDamage dollar_km.;  *;
* 4) Highlight and run the demo program. Notice that the  *;
* PropertyDamage value for the first row is being  *;
* displayed without the dollar sign. A length of 6 is  *;
* being used by default because that is the length of the  *;
* longest template.  *;
* 5) Add the DEFAULT= option to the PICTURE statement after  *;
* the name of the format.  *;
* picture dollar_km (round default=7)  *;
* low-<1000='009' (prefix='$' mult=1)  *;
* 1000-<1000000='009.9K' (prefix='$' mult=.01)  *;
* 1000000-high='009.9M' (prefix='$' mult=.00001);  *;
* 6) Highlight and run the demo program. Verify the formatted *;
* values of PropertyDamage.  *;
* 7) Modify the PICTURE statement to eliminate digits after  *;
* the decimal point.  *;
* picture dollar_km (round default=7)  *;
* low-<1000='009' (prefix='$' mult=1)  *;
* 1000-<1000000='009K' (prefix='$' mult=.001)  *;
* 1000000-high='009M' (prefix='$' mult=.000001);  *;
* 8) Highlight and run the demo program. Verify the formatted *;
* values of PropertyDamage.  *;
*****.
proc sort data=pg3.tornado_2017 out=work.tornado_2017;
    by descending PropertyDamage;
run;

title1 '2017 US Tornadoes';

```



```

title2 'by Descending Property Damage';
proc print data=work.tornado_2017;

    var State BeginDate Scale Deaths Injuries PropertyDamage;

    format PropertyDamage dollar20.;

run;

title;

```

2017 US Tornadoes by Descending Property Damage						
Obs	State	BeginDate	Scale	Deaths	Injuries	PropertyDamage
1	GA	22JAN2017	EF3	5	32	\$300,000,000
2	OK	06AUG2017	EF2	0	30	\$50,000,000
3	OK	16MAY2017	EF2	1	10	\$25,000,000
4	WI	16MAY2017	EF2	1	25	\$10,100,000
5	MS	21JAN2017	EF3	4	56	\$9,000,000
6	MO	28FEB2017	EF4	1	12	\$8,000,000
7	TN	01MAR2017	EF1	0	0	\$7,270,000
8	IL	28FEB2017	EF3	0	0	\$6,000,000
9	GA	22JAN2017	EF3	0	31	\$5,000,000
10	GA	22JAN2017	EF3	0	25	\$5,000,000
11	OH	05NOV2017	EF2	0	8	\$5,000,000
12	MS	30APR2017	EF2	1	0	\$3,500,000
13	IN	28FEB2017	EF3	0	1	\$3,200,000
14	LA	02JAN2017	EF1	0	0	\$2,500,000
15	IL	28FEB2017	EF3	1	1	\$2,000,000

```

proc format;

    picture dollar_KM (round)

        low-<1000='009' (prefix='$' mult=1)

        1000-<1000000='009.9K' (prefix='$' mult=.01)

        1000000-high='009.9M' (prefix='$' mult=.00001);

run;

proc sort data=pg3.tornado_2017 out=work.tornado_2017;

    by descending PropertyDamage;

run;

```

```

title1 '2017 US Tornadoes';
title2 'by Descending Property Damage';
proc print data=work.tornado_2017;
    var State BeginDate Scale Deaths Injuries PropertyDamage;
    format PropertyDamage dollar_KM.;
run;
title;

```

2017 US Tornadoes by Descending Property Damage						
Obs	State	BeginDate	Scale	Deaths	Injuries	PropertyDamage
1	GA	22JAN2017	EF3	5	32	300.0M
2	OK	06AUG2017	EF2	0	30	\$50.0M
3	OK	16MAY2017	EF2	1	10	\$25.0M
4	WI	16MAY2017	EF2	1	25	\$10.1M
5	MS	21JAN2017	EF3	4	56	\$9.0M
6	MO	28FEB2017	EF4	1	12	\$8.0M
7	TN	01MAR2017	EF1	0	0	\$7.3M

```

proc format;
    picture dollar_KM (round default=7)
        low-<1000='009' (prefix='$' mult=1)
        1000-<1000000='009.9K' (prefix='$' mult=.01)
        1000000-high='009.9M' (prefix='$' mult=.00001);
run;

proc sort data=pg3.tornado_2017 out=work.tornado_2017;
    by descending PropertyDamage;
run;

title1 '2017 US Tornadoes';
title2 'by Descending Property Damage';

```

```
proc print data=work.tornado_2017;

    var State BeginDate Scale Deaths Injuries PropertyDamage;

    format PropertyDamage dollar_KM.;

run;

title;
```

2017 US Tornadoes by Descending Property Damage						
Obs	State	BeginDate	Scale	Deaths	Injuries	PropertyDamage
1	GA	22JAN2017	EF3	5	32	\$300.0M
2	OK	06AUG2017	EF2	0	30	\$50.0M
3	OK	16MAY2017	EF2	1	10	\$25.0M
4	WI	16MAY2017	EF2	1	25	\$10.1M
5	MS	21JAN2017	EF3	4	56	\$9.0M
6	MO	28FEB2017	EF4	1	12	\$8.0M
7	TN	01MAR2017	EF1	0	0	\$7.3M

```
proc format;

    picture dollar_KM (round default=7)

        low-<1000='009' (prefix='$' mult=1)

        1000-<1000000='009K' (prefix='$' mult=.001)

        1000000-high='009M' (prefix='$' mult=.000001);

run;

proc sort data=pg3.tornado_2017 out=work.tornado_2017;

    by descending PropertyDamage;

run;

title1 '2017 US Tornadoes';

title2 'by Descending Property Damage';

proc print data=work.tornado_2017;

    var State BeginDate Scale Deaths Injuries PropertyDamage;

    format PropertyDamage dollar_KM.;
```

run;

title;

2017 US Tornadoes by Descending Property Damage						
Obs	State	BeginDate	Scale	Deaths	Injuries	PropertyDamage
1	GA	22JAN2017	EF3	5	32	\$300M
2	OK	06AUG2017	EF2	0	30	\$50M
3	OK	16MAY2017	EF2	1	10	\$25M
4	WI	16MAY2017	EF2	1	25	\$10M
5	MS	21JAN2017	EF3	4	56	\$9M
6	MO	28FEB2017	EF4	1	12	\$8M
7	TN	01MAR2017	EF1	0	0	\$7M

*****,

* LESSON 5, PRACTICE 1 *;

*****,

/*Practice Level 1: Specifying a Template Based on Date Directives

The pg3.storm_final table contains storm information such as StartDate and EndDate for storms from the 1980 through 2017 storm seasons.

Create a custom date format with the following layout:

three-letter-weekday.full-month-name.two-digit-day-of-month.two-digit-year (for example, Sat.September.09.17).

Open the p305p01.sas program in the practices folder.

Run the program. Notice the formatted values of StartDate and EndDate based on the WORDDATE format.

In the PROC FORMAT step, add a PICTURE statement to create a custom date format. Name the format NewDate.

Set an appropriate default length for NewDate that accommodates the formatted date values, such as: Sat.September.09.17.

Specify a date range from LOW to HIGH.

Use the appropriate date directives to produce a template with the following layout:

three-letter-weekday.full-month-name.two-digit-day-of-month.two-digit-year.

Specify the appropriate DATATYPE= option.

Modify the FORMAT statement in the PROC PRINT step to use the custom date format for the StartDate and EndDate columns.

Run the program and verify the results.

What is the formatted StartDate value for the last storm that occurred in the year 2016?

```
*/
```

```
proc format;
```

```
run;
```

```
proc sort data=pg3.storm_final out=work.storm_final;
```

```
by descending StartDate;
```

```
run;
```

```
title 'Detail Storm Report by Descending Start Date';
```

```
proc print data=work.storm_final;
```

```
var Name BasinName StartDate EndDate MaxWindMPH MinPressure;
```

```
format StartDate EndDate worddate.;
```

```
run;
```

```
title;
```

Detail Storm Report by Descending Start Date						
Obs	Name	BasinName	StartDate	EndDate	MaxWindMPH	MinPressure
1	HILDA	South Indian	December 24, 2017	December 30, 2017	60	980
2	OCKHI	North Indian	November 29, 2017	December 6, 2017	100	976
3	DAHLIA	South Indian	November 24, 2017	December 5, 2017	60	985
4	CEMPAKA	South Indian	November 22, 2017	December 1, 2017	40	998
5	RINA	North Atlantic	November 6, 2017	November 9, 2017	60	995
6	PHILIPPE	North Atlantic	October 28, 2017	October 29, 2017	60	997
7	SELMA	East Pacific	October 27, 2017	October 28, 2017	40	1005

```
proc format;
```

```

picture NewDate (default=19)

low-high = '%a.%B.%0d.%0y' (datatype=date);

run;

proc sort data=pg3.storm_final out=work.storm_final;

    by descending StartDate;

run;

title 'Detail Storm Report by Descending Start Date';

proc print data=work.storm_final;

    var Name BasinName StartDate EndDate MaxWindMPH MinPressure;

    format StartDate EndDate NewDate.;

run;

title;

```

Detail Storm Report by Descending Start Date

Obs	Name	BasinName	StartDate	EndDate	MaxWindMPH	MinPressure
1	HILDA	South Indian	Sun.December.24.17	Sat.December.30.17	60	980
2	OCKHI	North Indian	Wed.November.29.17	Wed.December.06.17	100	976
3	DAHLIA	South Indian	Fri.November.24.17	Tue.December.05.17	60	985
4	CEMPAKA	South Indian	Wed.November.22.17	Fri.December.01.17	40	998
5	RINA	North Atlantic	Mon.November.06.17	Thu.November.09.17	60	995
6	PHILIPPE	North Atlantic	Sat.October.28.17	Sun.October.29.17	60	997
7	SELMA	East Pacific	Fri.October.27.17	Sat.October.28.17	40	1005
8	OPHELIA	North Atlantic	Mon.October.09.17	Mon.October.16.17	115	960
9	NATE	North Atlantic	Wed.October.04.17	Mon.October.09.17	90	981
10	RAMON	East Pacific	Tue.October.03.17	Wed.October.04.17	45	1002
11	PILAR	East Pacific	Sat.September.23.17	Mon.September.25.17	45	1002
12	MARIA	North Atlantic	Sat.September.16.17	Sat.September.30.17	175	908

```

*****
* LESSON 5, PRACTICE 2
*
*****
/*Practice Level 2: Specifying a Template Based on Digit Selectors

```

The pg3.stocks table contains stock market data for the first weekday of the month for the years 2010 through 2017.

Create custom formats to display the DailyChange value with a suffix of USD and the Volume value with a suffix of shares.

Open the p305p02.sas program in the practices folder.

Run the DATA step and PROC PRINT step. View the results. Notice how the values of VolumeChar and DailyChangeChar are

left-justified because they are character columns based on the CATX function.

Because of this left justification, the commas, decimal points, and words are not aligned.

In the PROC FORMAT step, add an additional PICTURE statement to create a custom format named usd.

Set the default length to 11.

For the usd format, specify a numeric range of LOW through less than 0 to account for negative values.

Use a template of '009.99 USD' with a prefix of a negative sign.

For the usd format, specify a numeric range of 0 to HIGH to account for positive values. Use a template of '009.99 USD'.

Add a FORMAT statement to the DATA step to format Volume with the shares format and DailyChange with the usd format.

Run the program and view the results. Notice how the values of Volume and DailyChange are right-justified because they are numeric columns. Because of the right justification, the commas, decimal points, and words line up.

Do the DailyChange values match the DailyChangeChar values in the results, other than in regard to justification?

```
*/
```

```
proc format;
```

```
    picture shares
```

```
        low-high='000,000,009 shares';
```

```
run;
```

```
data work.stock_report;
```

```
set pg3.stocks(drop=High Low);
```

```
VolumeChar=catx(' ',put(Volume,comma18.),'shares');
```

```
DailyChange=Close-Open;
```

```
DailyChangeChar=catx(' ',DailyChange,'USD');
```

```
run;
```

```
title 'Monthly Stock Market Data for 2010 to 2017';
```

```
proc print data=work.stock_report;
```

```
run;
```

```
title;
```

Monthly Stock Market Data for 2010 to 2017

Obs	Stock	Date	Open	Close	Volume	VolumeChar	DailyChange	DailyChangeChar
1	ABC Company	01DEC2017	89.15	82.20	5976252	5,976,252 shares	-6.95	-6.95 USD
2	ABC Company	01NOV2017	81.85	88.90	5556471	5,556,471 shares	7.05	7.05 USD
3	ABC Company	02OCT2017	80.22	81.88	7019666	7,019,666 shares	1.66	1.66 USD
4	ABC Company	01SEP2017	80.16	80.22	5772280	5,772,280 shares	0.06	0.06 USD
5	ABC Company	01AUG2017	83.00	80.62	4801386	4,801,386 shares	-2.38	-2.38 USD
6	ABC Company	03JUL2017	74.30	83.46	8056590	8,056,590 shares	9.16	9.16 USD
7	ABC Company	01JUN2017	75.57	74.20	6439536	6,439,536 shares	-1.37	-1.37 USD

```
proc format;
```

```
picture shares
```

```
low-high='000,000,009 shares';
```

```
picture usd (default=11)
```

```
low-<0='-009.99 USD' (prefix='-')
```

```
0-high='009.99 USD';
```

```
run;
```



```

data work.stock_report;

  set pg3.stocks(drop=High Low);

  VolumeChar=catx(' ',put(Volume,comma18.),'shares');

  DailyChange=Close-Open;

  DailyChangeChar=catx(' ',DailyChange,'USD');

  format Volume shares. DailyChange usd.;

run;

```

```

title 'Monthly Stock Market Data for 2010 to 2017';

proc print data=work.stock_report;

run;

title;

```

Monthly Stock Market Data for 2010 to 2017

Obs	Stock	Date	Open	Close	Volume	VolumeChar	DailyChange	DailyChangeChar
1	ABC Company	01DEC2017	89.15	82.20	5,976,252 shares	5,976,252 shares	-6.95 USD	-6.95 USD
2	ABC Company	01NOV2017	81.85	88.90	5,556,471 shares	5,556,471 shares	7.05 USD	7.05 USD
3	ABC Company	02OCT2017	80.22	81.88	7,019,666 shares	7,019,666 shares	1.66 USD	1.66 USD
4	ABC Company	01SEP2017	80.16	80.22	5,772,280 shares	5,772,280 shares	0.06 USD	0.06 USD
5	ABC Company	01AUG2017	83.00	80.62	4,801,386 shares	4,801,386 shares	-2.38 USD	-2.38 USD
6	ABC Company	03JUL2017	74.30	83.46	8,056,590 shares	8,056,590 shares	9.16 USD	9.16 USD
7	ABC Company	01JUN2017	75.57	74.20	6,439,536 shares	6,439,536 shares	-1.37 USD	-1.37 USD
8	ABC Company	01MAY2017	76.88	75.55	6,896,904 shares	6,896,904 shares	-1.33 USD	-1.33 USD
9	ABC Company	03APR2017	91.49	76.38	10,709,200 shares	10,709,200 shares	-15.11 USD	-15.11 USD
10	ABC Company	01MAR2017	92.64	91.38	5,025,627 shares	5,025,627 shares	-1.26 USD	-1.26 USD

```

*****
* Creating Functions Containing One Argument *;
*****
*****
* Demo *;
* 1) Highlight and run the PROC FCMP step. View the output *;
* table storing the new function. *;
* 2) Highlight and run the DATA step. View the error in the *;

```

```

* SAS log that is related to SAS not finding the FtoC    *;
* function.                                           *;
* 3) Highlight and run the OPTIONS statement and the DATA    *;
* step. View the new column TavgC in the NewYork table.    *;
* 4) Open pg3.weather_sydney_daily2017. Notice that the Tavg    *;
* values are in the unit of Celsius.                *;
* 5) Add syntax to the PROC FCMP step to create the CtoF    *;
* function.                                           *;
* function CtoF(TempC);                               *;
* TempF=round(TempC*9/5+32,.01);                     *;
* return(TempF);                                       *;
* endsub;                                             *;
* 6) Highlight and run the PROC FCMP step. Confirm that there    *;
* are no errors. View the output table storing the new    *;
* function.                                           *;
* 7) Copy and paste the DATA step for the New York data.    *;
* Modify the new DATA step to use the Sydney data with the    *;
* CtoF function.                                       *;
* data work.Sydney;                                   *;
* set pg3.weather_sydney_daily2017;                 *;
* TavgF=CtoF(TAvg);                                   *;
* run;                                                *;
* 8) Highlight and run the new DATA step. View the new column    *;
* TavgF in the Sydney table.                         *;
*****

```

```

proc fcmp outlib=pg3.funcs.weather;

function FtoC(TempF);

TempC=round((TempF-32)*5/9,.01);

```

```

return(TempC);

endsub;

run;

```

Table: PG3.FUNCS | View: Column names | Filter: (none)

Columns: Total rows: 8 Total columns: 10

	Key	Owner	Sequence	Type	Subtype	Name	Continue	NValue
1	WEATHER	CMP	0	Header	Package		0	.
2	F.WEATHER.FTOC	CMP	0	Prototype	FCmp	weather	0	.
3	F.WEATHER.FTOC	CMP	1	Header	Function		0	.
4	F.WEATHER.FTOC	CMP	2	Statement Source	Executable	FUNCTION	0	65
5	F.WEATHER.FTOC	CMP	3	Statement Source	Executable	ASSIGN	0	1
6	F.WEATHER.FTOC	CMP	4	Statement Source	Executable	RETURN	0	1
7	F.WEATHER.FTOC	CMP	5	Statement Source	Executable	ENDSUB	0	14
8	F.WEATHER.FTOC	CMP	6	Symbol		_HOSTNAME_	0	.

NOTE: Function FtoC saved to pg3.funcs.weather.

```
options cmplib=pg3.funcs;
```

```
data work.NewYork;
```

```
    set pg3.weather_ny_daily2017;
```

```
    TavGC=FtoC(Tavg);
```

```
run;
```

Table: WORK.NEWYORK View: Column names Filter: (none)

Total rows: 365 Total columns: 8

	City	Country	Date	Precip	Punit	Tavg	Tunit	TavgC
1	New York	United States	01JAN2017	0	IN	46	F	7.78
2	New York	United States	02JAN2017	0.31	IN	40	F	4.44
3	New York	United States	03JAN2017	0.46	IN	43	F	6.11
4	New York	United States	04JAN2017	0	IN	46	F	7.78
5	New York	United States	05JAN2017	0	IN	34	F	1.11
6	New York	United States	06JAN2017	0.03	IN	32	F	0
7	New York	United States	07JAN2017	0.22	IN	25	F	-3.89
8	New York	United States	08JAN2017	0	IN	21	F	-6.11
9	New York	United States	09JAN2017	0	IN	20	F	-6.67
10	New York	United States	10JAN2017	0	IN	27	F	-2.78
11	New York	United States	11JAN2017	0.46	IN	45	F	7.22
12	New York	United States	12JAN2017	0.1	IN	50	F	10

```
proc fcmp outlib=pg3.funcs.weather;
```

```
function FtoC(TempF);
```

```
TempC=round((TempF-32)*5/9,.01);
```

```
return(TempC);
```

```
endsub;
```

```
function CtoF(TempC);
```

```
TempF=round(TempC*9/5+32,.01);
```

```
return(TempF);
```

```
endsub;
```

```
run;
```





WARNING: Function 'FtoC' was defined in a previous package. Function 'FtoC' as defined in the current program will be used as default when the package is not specified.

```
data work.Sydney;
```

```
set pg3.weather_sydney_daily2017;
```

TavgF=CtoF(Tavg);

run;

Table: WORK.SYDNEY | View: Column names |     | Filter: (none)

Total rows: 365 Total columns: 8

	City	Country	Date	Precip	Punit	Tavg	Tunit	TavgF
1	Sydney	Australia	01JAN2017	0	CM	23.33	C	73.99
2	Sydney	Australia	02JAN2017	0.1	CM	21.11	C	70
3	Sydney	Australia	03JAN2017	0.23	CM	21.67	C	71.01
4	Sydney	Australia	04JAN2017	0.15	CM	21.67	C	71.01
5	Sydney	Australia	05JAN2017	0.51	CM	22.22	C	72
6	Sydney	Australia	06JAN2017	0.81	CM	22.22	C	72
7	Sydney	Australia	07JAN2017	1.45	CM	24.44	C	75.99
8	Sydney	Australia	08JAN2017	0	CM	25	C	77

*****;

- * Activity 5.04 *;
- * 1) Create another custom function in the PROC FCMP *;
- * step. *;
- * - Add a FUNCTION statement to create a function *;
- * named INTOCM that has a numeric argument of Pin. *;
- * - Add the following assignment statement: *;
- * Pcm=Pin*2.54; *;
- * - Add a RETURN statement to return the value of *;
- * Pcm. *;
- * - Add an END SUB statement. *;
- * 2) Run the program and verify that the PrecipCM values *;
- * are 2.54 times bigger than the Precip values. Does *;
- * the PROC SQL step use the custom functions *;
- * successfully? *;

*****;

```

proc fcmp outlib=pg3.funcs.weather;

    function FtoC(TempF);
        TempC=round((TempF-32)*5/9,.01);
        return(TempC);
    endsub;

    /* add FUNCTION, assignment, RETURN, and ENDSUB statements here */
    function INtoCM(Pin);
        Pcm=Pin*2.54;
        return(Pcm);
    endsub;
run;

options cmplib=pg3.funcs;

proc sql;
    select Date, Tavg, Tunit, FtoC(Tavg) as TavgC,
        Precip, Punit, INtoCM(Precip) as PrecipCM
    from pg3.weather_ny_daily2017;
quit;

```

Date	Tavg	Tunit	TavgC	Precip	Punit	PrecipCM
01JAN2017	46	F	7.78	0	IN	0
02JAN2017	40	F	4.44	0.31	IN	0.7874
03JAN2017	43	F	6.11	0.46	IN	1.1684
04JAN2017	46	F	7.78	0	IN	0
05JAN2017	34	F	1.11	0	IN	0
06JAN2017	32	F	0	0.03	IN	0.0762
07JAN2017	25	F	-3.89	0.22	IN	0.5588
08JAN2017	21	F	-6.11	0	IN	0
09JAN2017	20	F	-6.67	0	IN	0
10JAN2017	27	F	-2.78	0	IN	0
11JAN2017	45	F	7.22	0.46	IN	1.1684
12JAN2017	50	F	10	0.1	IN	0.254
13JAN2017	50	F	10	0	IN	0
14JAN2017	32	F	0	0.15	IN	0.381

Use a comma to separate multiple arguments.

A dollar sign indicates a character argument.

```
function CnvTemp(Temp, Unit $);
  if upcase(Unit)='F'
    then NewTemp=round((Temp-32)*5/9,.01);
  else if upcase(Unit)='C'
    then NewTemp=round(Temp*9/5+32,.01);
  return(NewTemp);
endsub;
```

```
function CharTemp(Temp, Unit $, FinalUnit $) $ 10;
```

A length can be specified for the character value. If a length is not specified, the length defaults to 8.

*****;

* Creating Functions Containing Multiple Arguments *;

*****;

*****;

```
* Demo                                *;
* 1) Notice the syntax for the PROC FCMP step. The Unit    *;
*   argument specifies whether the temperature is currently *;
*   a Fahrenheit or Celsius value. If the current          *;
*   temperature is Fahrenheit, it is converted to Celsius.  *;
*   Conversely, if the current temperature is Celsius, it is *;
*   converted to Fahrenheit.                                *;
* 2) Highlight and run the PROC FCMP step, the OPTIONS     *;
*   statement, and the DATA step. View the new column     *;
*   TavgConverted in the SydneyNewYork table. The          *;
*   temperature is in Fahrenheit for the Sydney rows and    *;
*   Celsius for the New York rows.                          *;
* 3) Add a third argument to the CnvTemp function that      *;
*   specifies the desired final unit.                        *;
*   function CnvTemp(Temp, Unit $, FinalUnit $);           *;
* 4) Delete the two conditional statements for the CnvTemp  *;
*   function and uncomment the four conditional statements. *;
* 5) In the DATA step, delete the TavgConverted assignment *;
*   statement and uncomment the TavgF and TavgC assignment  *;
*   statements.                                             *;
*   TavgF=CnvTemp(TAvg,Tunit,'F');                          *;
*   TavgC=CnvTemp(TAvg,Tunit,'C');                          *;
* 6) Run the demo program that includes the SGPLOT procedure.*;
*   View the new columns TavgF and TavgC in the             *;
*   SydneyNewYork table. Also view the graph of the TavgF  *;
*   column.                                                 *;
```



```

* 7) In the PROC FCMP step, copy and paste the syntax for    *;
*   creating the CnvTemp function within the step. Modify    *;
*   the syntax to create a function that returns a character *;
*   value that concatenates the letter F or C after the      *;
*   temperature value.                                     *;
*   function CharTemp(Temp, Unit $, FinalUnit $) $ 20;      *;
*   ...                                                     *;
*   return(catx(' ',put(NewTemp,8.2),FinalUnit));          *;
* 8) In the DATA step, add the following two assignment    *;
*   statements:                                             *;
*   TavgFchar=CharTemp(TAvg,Tunit,'F');                    *;
*   TavgCchar=CharTemp(TAvg,Tunit,'C');                    *;
* 9) Highlight and run the PROC FCMP step, the OPTIONS      *;
*   statement, and the DATA step. View the new columns    *;
*   TavgFchar and TavgCchar in the SydneyNewYork table.    *;
*****

```

```

proc fcmp outlib=pg3.funcs.temps;

function CnvTemp(Temp, Unit $);
  if upcase(Unit)='F'
    then NewTemp=round((Temp-32)*5/9,.01);
  else if upcase(Unit)='C'
    then NewTemp=round(Temp*9/5+32,.01);
/*
  if upcase(Unit)='F' and upcase(FinalUnit)='C'
    then NewTemp=round((Temp-32)*5/9,.01);
  else if upcase(Unit)='F' and upcase(FinalUnit)='F'
    then NewTemp=Temp;
  else if upcase(Unit)='C' and upcase(FinalUnit)='F'

```

```

        then NewTemp=round(Temp*9/5+32,.01);
    else if upcase(Unit)='C' and upcase(FinalUnit)='C'
        then NewTemp=Temp;
    */
    return(NewTemp);
endsub;

run;

options cmplib=pg3.funcs;

data work.SydneyNewYork;
    set pg3.weather_sydney_ny_monthly2017;
    TavgConverted=CnvTemp(TAvg,Tunit);
    *TavgF=CnvTemp(TAvg,Tunit,'F');
    *TavgC=CnvTemp(TAvg,Tunit,'C');
run;

```

: WORK.SYDNEYNEWYORK | View: Column names |     | Filter: (none)

Total rows: 24 Total columns: 8

	City	Country	Date	Precip	Punit	Tavg	Tunit	TavgConverted
1	Sydney	Australia	JAN2017	4.85	CM	25.67	C	78.21
2	Sydney	Australia	FEB2017	15.8	CM	24.78	C	76.6
3	Sydney	Australia	MAR2017	18.92	CM	22.83	C	73.09
4	Sydney	Australia	APR2017	9.45	CM	19.11	C	66.4
5	Sydney	Australia	MAY2017	3.25	CM	16.78	C	62.2
6	Sydney	Australia	JUN2017	11.35	CM	14.11	C	57.4
7	Sydney	Australia	JUL2017	1.8	CM	13.78	C	56.8
8	Sydney	Australia	AUG2017	2.72	CM	14.28	C	57.7
9	Sydney	Australia	SEP2017	0.03	CM	18.06	C	64.51
10	Sydney	Australia	OCT2017	5.97	CM	20.17	C	68.31
11	Sydney	Australia	NOV2017	3.78	CM	20.72	C	69.3
12	Sydney	Australia	DEC2017	4.45	CM	24.44	C	75.99
13	New York	United States	JAN2017	4.65	IN	38.6	F	3.67
14	New York	United States	FEB2017	1.58	IN	40.4	F	4.67
15	New York	United States	MAR2017	5.8	IN	39.4	F	4.11
16	New York	United States	APR2017	4.07	IN	55.1	F	12.83
17	New York	United States	MAY2017	5.83	IN	60.5	F	15.83
18	New York	United States	JUN2017	4.2	IN	70.9	F	21.61
19	New York	United States	JUL2017	4.66	IN	76.4	F	24.67

```
proc fcmp outlib=pg3.funcs.temps;
```

```
function CnvTemp(Temp, Unit $, FinalUnit $);
```

```
    if upcase(Unit)='F' and upcase(FinalUnit)='C'
```

```
        then NewTemp=round((Temp-32)*5/9,.01);
```

```
    else if upcase(Unit)='F' and upcase(FinalUnit)='F'
```

```
        then NewTemp=Temp;
```

```
    else if upcase(Unit)='C' and upcase(FinalUnit)='F'
```

```
        then NewTemp=round(Temp*9/5+32,.01);
```

```
    else if upcase(Unit)='C' and upcase(FinalUnit)='C'
```

```





        then NewTemp=Temp;
    return(NewTemp);
endsub;
run;

options cmplib=pg3.funcs;

data work.SydneyNewYork;
    set pg3.weather_sydney_ny_monthly2017;
    TavgF=CnvTemp(TAvg,Tunit,'F');
    TavgC=CnvTemp(TAvg,Tunit,'C');
run;

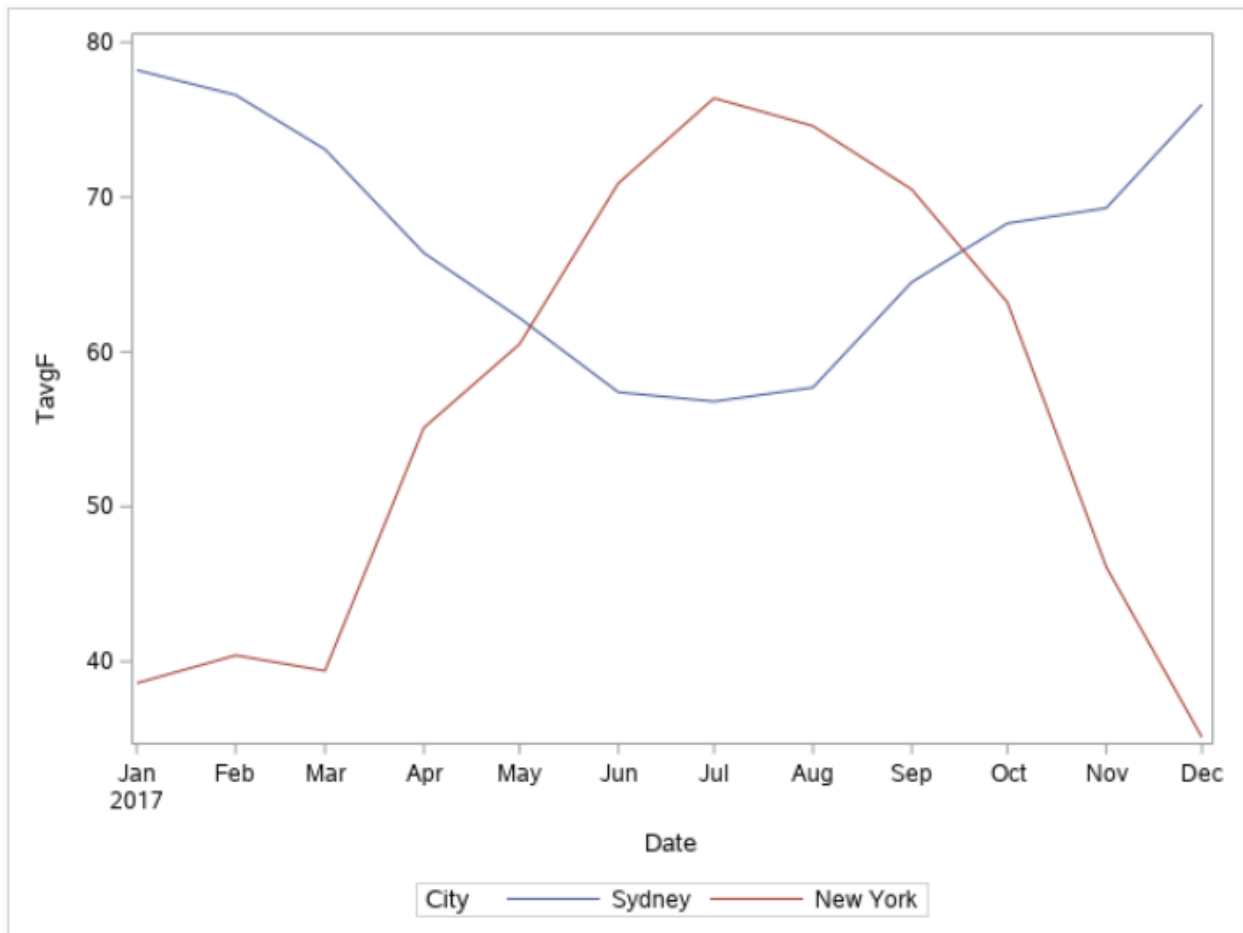
proc sgplot data=work.SydneyNewYork;
    series x=Date y=TavgF / group=city;
run;

```

Table: WORK.SYDNEYNEWYORK | View: Column names |     | Filter: (none)

Total rows: 24 Total columns: 9

	City	Country	Date	Precip	Punit	Tavg	Tunit	TavgF	TavgC
1	Sydney	Australia	JAN2017	4.85	CM	25.67	C	78.21	25.67
2	Sydney	Australia	FEB2017	15.8	CM	24.78	C	76.6	24.78
3	Sydney	Australia	MAR2017	18.92	CM	22.83	C	73.09	22.83
4	Sydney	Australia	APR2017	9.45	CM	19.11	C	66.4	19.11
5	Sydney	Australia	MAY2017	3.25	CM	16.78	C	62.2	16.78
6	Sydney	Australia	JUN2017	11.35	CM	14.11	C	57.4	14.11
7	Sydney	Australia	JUL2017	1.8	CM	13.78	C	56.8	13.78
8	Sydney	Australia	AUG2017	2.72	CM	14.28	C	57.7	14.28
9	Sydney	Australia	SEP2017	0.03	CM	18.06	C	64.51	18.06
10	Sydney	Australia	OCT2017	5.97	CM	20.17	C	68.31	20.17
11	Sydney	Australia	NOV2017	3.78	CM	20.72	C	69.3	20.72
12	Sydney	Australia	DEC2017	4.45	CM	24.44	C	75.99	24.44
13	New York	United States	JAN2017	4.65	IN	38.6	F	38.6	3.67
14	New York	United States	FEB2017	1.58	IN	40.4	F	40.4	4.67
15	New York	United States	MAR2017	5.8	IN	39.4	F	39.4	4.11



```

proc fcmp outlib=pg3.funcs.temps;
function CnvTemp(Temp, Unit $, FinalUnit $);
  if upcase(Unit)='F' and upcase(FinalUnit)='C'
    then NewTemp=round((Temp-32)*5/9,.01);
  else if upcase(Unit)='F' and upcase(FinalUnit)='F'
    then NewTemp=Temp;
  else if upcase(Unit)='C' and upcase(FinalUnit)='F'
    then NewTemp=round(Temp*9/5+32,.01);
  else if upcase(Unit)='C' and upcase(FinalUnit)='C'
    then NewTemp=Temp;
  return(NewTemp);
endsub;

```

```

function CharTemp(Temp, Unit $, FinalUnit $) $ 20;
  if upcase(Unit)='F' and upcase(FinalUnit)='C'
    then NewTemp=round((Temp-32)*5/9,.01);
  else if upcase(Unit)='F' and upcase(FinalUnit)='F'
    then NewTemp=Temp;
  else if upcase(Unit)='C' and upcase(FinalUnit)='F'
    then NewTemp=round(Temp*9/5+32,.01);
  else if upcase(Unit)='C' and upcase(FinalUnit)='C'
    then NewTemp=Temp;
  return(catx(' ',put(NewTemp,8.2),FinalUnit));
endsub;
run;

```

```

options cmplib=pg3.funcs;

```

```

data work.SydneyNewYork;
  set pg3.weather_sydney_ny_monthly2017;
  TavgF=CnvTemp(TAvg,Tunit,'F');
  TavgC=CnvTemp(TAvg,Tunit,'C');
  TavgFchar=CharTemp(TAvg,Tunit,'F');
  TavgCchar=CharTemp(TAvg,Tunit,'C');
run;

```

```

proc sgplot data=work.SydneyNewYork;
  series x=Date y=TavgF / group=city;
run;

```

: WORK.SYDNEYNEWYORK View: Column names Filter: (none)

Total rows: 24 Total columns: 11

	City	Country	Date	Precip	Punit	Tavg	Tunit	TavgF	TavgC	TavgFchar	TavgCchar
1	Sydney	Australia	JAN2017	4.85	CM	25.67	C	78.21	25.67	78.21 F	25.67 C
2	Sydney	Australia	FEB2017	15.8	CM	24.78	C	76.6	24.78	76.60 F	24.78 C
3	Sydney	Australia	MAR2017	18.92	CM	22.83	C	73.09	22.83	73.09 F	22.83 C
4	Sydney	Australia	APR2017	9.45	CM	19.11	C	66.4	19.11	66.40 F	19.11 C
5	Sydney	Australia	MAY2017	3.25	CM	16.78	C	62.2	16.78	62.20 F	16.78 C
6	Sydney	Australia	JUN2017	11.35	CM	14.11	C	57.4	14.11	57.40 F	14.11 C
7	Sydney	Australia	JUL2017	1.8	CM	13.78	C	56.8	13.78	56.80 F	13.78 C
8	Sydney	Australia	AUG2017	2.72	CM	14.28	C	57.7	14.28	57.70 F	14.28 C
9	Sydney	Australia	SEP2017	0.03	CM	18.06	C	64.51	18.06	64.51 F	18.06 C
10	Sydney	Australia	OCT2017	5.97	CM	20.17	C	68.31	20.17	68.31 F	20.17 C
11	Sydney	Australia	NOV2017	3.78	CM	20.72	C	69.3	20.72	69.30 F	20.72 C
12	Sydney	Australia	DEC2017	4.45	CM	24.44	C	75.99	24.44	75.99 F	24.44 C
13	New York	United States	JAN2017	4.65	IN	38.6	F	38.6	3.67	38.60 F	3.67 C
14	New York	United States	FEB2017	1.58	IN	40.4	F	40.4	4.67	40.40 F	4.67 C
15	New York	United States	MAR2017	5.8	IN	39.4	F	39.4	4.11	39.40 F	4.11 C

```
PROC FCMP OUTLIB=libref.table.package;
  SUBROUTINE subroutine-name(arg-1 <, arg-2, ...>);
    OUTARGS out-arg1 <, out-arg-2, ...>;
    ... programming statements ...
  ENDSUB;
RUN;
```

```
CALL subroutine-name(arg-1, arg-2, ... , arg-n);
```

call a subroutine
from a DATA step

Custom CALL routines
are often referred to
as subroutines.




```

proc fcmp outlib=pg3.funcs.weather;
subroutine ConvTemp2(Temp, Unit $);
  outargs Temp, Unit;
  if upcase(Unit)='F' then do;
    Temp=round((Temp-32)*5/9,.01);
    Unit='C';
  end;
  else if upcase(Unit)='C' then do;
    Temp=round(Temp*9/5+32,.01);
    Unit='F';
  end;
endsub;
run;

```

```

options cmplib=pg3.funcs;

data work.NewYork;
  keep City Country Date
      Tavg Tunit;
  set pg3.weather_ny_daily2017;
  call ConvTemp2(Tavg, Tunit);
run;

proc print data=NewYork (obs=5);
run;

```

pg3.weather_ny_daily2017

Obs	City	Country	Date	Tavg	Tunit
1	New York	United States	01JAN2017	46	F
2	New York	United States	02JAN2017	40	F
3	New York	United States	03JAN2017	43	F
4	New York	United States	04JAN2017	46	F
5	New York	United States	05JAN2017	34	F



work.NewYork

Obs	City	Country	Date	Tavg	Tunit
1	New York	United States	01JAN2017	7.78	C
2	New York	United States	02JAN2017	4.44	C
3	New York	United States	03JAN2017	6.11	C
4	New York	United States	04JAN2017	7.78	C
5	New York	United States	05JAN2017	1.11	C

Advantages to Custom Functions and Call Routines

There are several advantages to using custom functions and custom CALL routines:

- 1) Programs are easier to read, write, and modify.
- 2) Functions and CALL routines are independent and reusable.
 - A program that calls a routine is not affected by the routine's implementation.
 - Any program that has access to the data set where the function or routine is stored can call the routine.

Note: PROC FCMP routines that you create cannot have the same name as built-in SAS functions. If the names are the same, then SAS generates an error message.

```

*****
* LESSON 5, PRACTICE 4
*
*****

```

`/*Practice Level 1: Creating a Custom Function That Returns a Numeric Value`

The `pg3.class_tests` table contains student scores (ranging from 1 to 10) for four tests and a final exam.

Create a function that calculates each student's final score. The final score is the average of six scores: the four tests and the final exam, which is counted twice.

1) Open the `p305p04.sas` program in the practices folder.

Highlight and run the DATA step and the PROC PRINT step to view the student scores.

2) Create a custom function that calculates each student's final score:

Add a PROC FCMP statement that has an OUTLIB= option to store the custom function in `pg3.myfunctions.class`.

Add a FUNCTION statement to name the function `CalcScore`. The function should contain five arguments: `T1`, `T2`, `T3`, `T4`, and `F`.

Add this assignment statement:

```
FScore=round(sum(of T1-T4, 2*F)/6,.01);
```

Add a RETURN statement to return the value of `FScore`.

Add an ENDSUB statement and a RUN statement.

3) Highlight and run the PROC FCMP step. View the log and confirm that the function is saved.

4) After the PROC FCMP step and before the DATA step, add a global OPTIONS statement with the CMPLIB= option

to specify the table location of the function, `pg3.myfunctions`.

5) In the DATA step, add an assignment to create `FinalScore`. This column is equal to the `CalcScore` function

with the arguments `Test1`, `Test2`, `Test3`, `Test4`, and `Final`.

6) Run the program and verify the results.

How many students have a final score greater than 9.00? Note: Type a numeric value.

```
*/
```

```
/* add a PROC FCMP step */
```

```
/* add an OPTIONS statement */
```

```
data work.scores;

    set pg3.class_tests;

    /* add an assignment statement */
```

```
run;
```

```
title 'Student Scores';

proc print data=work.scores;

run;

title;
```

Student Scores						
Obs	Name	Test1	Test2	Test3	Test4	Final
1	Alfred	8	7	6	9	8
2	Alice	7	6	4	9	8
3	Barbara	9	8	7	.	7
4	Carol	6	5	5	8	8
5	Henry	8	.	6	10	7
6	James	9	8	8	10	10
7	Jane	8	7	6	9	6
8	Janet	7	7	5	9	6
9	Jeffrey	5	6	4	8	7
10	John	6	7	5	9	6
11	Joyce	8	7	.	10	9
12	Judy	9	9	7	10	10
13	Louise	9	10	6	.	9
14	Mary	8	8	6	10	10
15	Philip	7	8	4	9	8
16	Robert	7	7	5	9	7
17	Ronald	.	6	5	8	9
18	Thomas	7	6	6	9	8
19	William	8	8	7	10	9

```
/* add a PROC FCMP step */

proc fcmp outlib=pg3.myfunctions.class;
```

```

        function CalcScore (T1, T2, T3, T4, F);
            FScore=round(sum(of T1-T4, 2*F)/6,.01);
            return(FScore);
        endsub;
/* add an OPTIONS statement */
options cmplib=pg3.myfunctions;

data work.scores;
    set pg3.class_tests;
/* add an assignment statement */
    FinalScore=CalcScore(Test1,Test2,Test3,Test4,Final);
run;

title 'Student Scores';
proc print data=work.scores;
run;
title;

```

Student Scores

Obs	Name	Test1	Test2	Test3	Test4	Final	FinalScore
1	Alfred	8	7	6	9	8	7.67
2	Alice	7	6	4	9	8	7.00
3	Barbara	9	8	7	.	7	6.33
4	Carol	6	5	5	8	8	6.67
5	Henry	8	.	6	10	7	6.33
6	James	9	8	8	10	10	9.17
7	Jane	8	7	6	9	6	7.00
8	Janet	7	7	5	9	6	6.67
9	Jeffrey	5	6	4	8	7	6.17
10	John	6	7	5	9	6	6.50
11	Joyce	8	7	.	10	9	7.17
12	Judy	9	9	7	10	10	9.17
13	Louise	9	10	6	.	9	7.17
14	Mary	8	8	6	10	10	8.67
15	Philip	7	8	4	9	8	7.33
16	Robert	7	7	5	9	7	7.00
17	Ronald	.	6	5	8	9	6.17
18	Thomas	7	6	6	9	8	7.33
19	William	8	8	7	10	9	8.50

*Solution;

```
proc fcmp outlib=pg3.myfunctions.class;
```

```
function CalcScore(T1, T2, T3, T4, F);
```

```
    FScore=round(sum(of T1-T4, 2*F)/6,.01);
```

```
    return(FScore);
```

```
endsub;
```

```
run;
```

```
options cmplib=pg3.myfunctions;
```

```
data work.scores;
```

```
    set pg3.class_tests;
```

```
    FinalScore=CalcScore(of Test1-Test4, Final);
```

```
run;
```

```
title 'Student Scores';
```

```
proc print data=work.scores;
```

```
run;
```

```
title;
```

```
*****,
```

```
* LESSON 5, PRACTICE 5          *;
```

```
*****,
```

```
/*Practice Level 2: Creating a Custom Function That Returns a Character Value
```

The sashelp.baseball table contains baseball statistics per each player.

The Name column contains the player's name with the last name appearing before the first name (for example, Davis, Alan).

Create a function that switches the order of the first and last names.

1) Open the p305p05.sas program in the practices folder.

2) Add a PROC FCMP step to create a custom function.

Store the function in a package named baseball within the pg3.myfunctions table.

Name the function flip.

The function should contain a character argument named LastFirst and return a character value with an appropriate length.

Use the following RETURN statement:

```
return(catx(' ',scan(LastFirst,2,' '),  
          scan(LastFirst,1,''));
```

Highlight and run the PROC FCMP step. View the log and confirm that the function is saved.

3) Add an OPTIONS statement with the CMPLIB= option to specify the table that SAS searches for the baseball package.

4) In the DATA step, add an assignment to create Player. This column is equal to the flip function with the argument of Name.

5) Run the program and verify the results.

What is the name of the baseball player in row 21? Note: Type the answer as shown in the results.

```
*/  
/* add a PROC FCMP step */  
proc fcmp outlib=pg3.myfunctions.baseball;  
    function flip>LastFirst $) $ 32;  
        return(catx(' ',scan>LastFirst,2,','),scan>LastFirst,1,','));  
    endsub;  
run;  
  
/* add an OPTIONS statement */  
options cmplib=pg3.myfunctions;  
  
data work.FlipNames;  
    set sashelp.baseball(keep=Name Team);  
    /* add an assignment statement */  
        Player=flip(Name);  
    drop Name;  
run;  
  
title 'Baseball Players and Teams';  
proc print data=work.FlipNames;  
    var Player Team;  
run;  
title;
```

Baseball Players and Teams

Obs	Player	Team
1	Andy Allanson	Cleveland
2	Alan Ashby	Houston
3	Alan Davis	Seattle
4	Andre Dawson	Montreal
5	Andres Galarraga	Montreal
6	Alfredo Griffin	Oakland
7	Al Newman	Montreal
8	Argenis Salazar	Kansas City
9	Andres Thomas	Atlanta
10	Andre Thornton	Cleveland
11	Alan Trammell	Detroit
12	Alex Trevino	Los Angeles
13	Andy Van Slyke	St Louis
14	Alan Wiggins	Baltimore
15	Bill Almon	Pittsburgh