SAS® Functions - Simple But Powerful Techniques

Steve Holson, General Dynamics
Jenny Putman, General Dynamics

Introduction
This tutorial introduces SAS® functions. If you have ever asked:

- What is a SAS® function?
- What exactly do SAS® functions do?
- How are SAS® functions used?

you've come to the right tutorial. These questions, as well as several examples of popular functions, will be discussed. Familiarity with the DATA step is assumed.

What Is A SAS® Function?
A SAS® function is a routine that acts upon one or more arguments (variables, constants, expressions) and returns a value. The syntax for a function is:

FUNCTIONNAME(argument,argument,...)

Functions are used in programming statements and are useful as programming shortcuts for:

- Performing numeric calculations
- Manipulating character and numeric data
- Creating new values from existing data

As previously mentioned, an argument can either be a constant, a variable or an expression. The following examples demonstrate the three argument forms.

The INT function returns the integer value of an argument. In this example, the argument is a constant. The value is 2.5.

The statements

Data;
   Wholenum = INT(2.5);
Run;

give the resulting variable, Wholenum, the value 2.

The MEAN function computes the average of the arguments. The arguments X, Y and Z in this example are variables.

The statements

Data;
   X = 2;
   Y = 6;
   Z = 4;
   Avg = MEAN(X,Y,Z);
Run;

give the resulting variable, Avg, the value 4.

The SUM function adds the values of the arguments. In this example, the first argument is a variable and the second, an expression.

Data:
   Set IP.Expenses;
   Interest = SUM(Home_Int,(Pers_Int *.65));
Run;

Functions By Categories
The following functions will be discussed in this paper.

Arithmetic
- MAX
- MIN

Truncation
- CEIL
- FLOOR
- INT
- ROUND

Sample statistics
- MEAN
- SUM

Character
- INDEX
- SCAN
- SUBSTR
- TRIM

Date & time
- DAY
- JULDATE
- MDY
- MONTH
- TODAY
- YEAR

Special
- PUT

As previously mentioned, an argument can either be a constant, a variable or an expression. The following examples demonstrate the three argument forms.

The INT function returns the integer value of an argument. In this example, the argument is a constant. The value is 2.5.

The statements

Data;
   Wholenum = INT(2.5);
Run;

give the resulting variable, Wholenum, the value 2.

The MEAN function computes the average of the arguments. The arguments X, Y and Z in this example are variables.

The statements

Data;
   X = 2;
   Y = 6;
   Z = 4;
   Avg = MEAN(X,Y,Z);
Run;

give the resulting variable, Avg, the value 4.

The SUM function adds the values of the arguments. In this example, the first argument is a variable and the second, an expression.

Data:
   Set IP.Expenses;
   Interest = SUM(Home_Int,(Pers_Int *.65));
Run;

The MAX function returns the largest value among the nonmissing values of the arguments. Arguments must be numeric. MAX requires two or more nonmissing values.

The following program outputs the highest pay rate in each department.

Input: IP.Payfile

<table>
<thead>
<tr>
<th>Dept</th>
<th>Payrate</th>
</tr>
</thead>
<tbody>
<tr>
<td>525-0</td>
<td>11.25</td>
</tr>
<tr>
<td>525-0</td>
<td>9.50</td>
</tr>
<tr>
<td>525-0</td>
<td>13.25</td>
</tr>
<tr>
<td>526-0</td>
<td>15.75</td>
</tr>
<tr>
<td>526-0</td>
<td>10.30</td>
</tr>
</tbody>
</table>
Program

Data Highpay (Keep = Dept Highrate);
   Set IP.Payfile;
   By Dept;
   Highrate = MAX(Highrate,Payrate);
   If Last.Dept Then
      Do;
         Output Highpay;
         Highrate = .;
      End;
   Retain Highrate;
Run;

Output: Highpay
Dept Highrate
525-0 13.25
526-0 15.75

The MIN function returns the smallest value among the nonmissing values of the arguments. Arguments must be numeric. MIN requires two or more nonmissing values.

The following example outputs the lowest pay rate in each department.

Input: IP.Payfile
Dept Payrate
525-0 11.25
525-0 9.50
526-0 13.25
526-0 15.75

Program

Data Lowpay (Keep = Dept Lowrate);
   Set IP.Payfile;
   By Dept;
   Lowrate = MIN(Lowrate,Payrate);
   If Last.Dept Then
      Do;
         Output Lowpay;
         Lowrate = .;
      End;
   Retain Lowrate;
Run;

Output: Lowpay
Dept Lowrate
525-0 9.50
526-0 10.30

The CEIL function returns the next largest or equal integer value of the argument. The argument must be numeric.

The next largest/equal integer will be returned from the argument Pay in the following example.

Input: IP.Data
Name Pay
Graham 374.00
Hinkle 399.01
Hull 425.99

Program

Data Out;
   Set IP.Data;
   High_Amt = CEIL(Pay);
Run;

Output: Out
Name Pay High_Amt
Graham 374.00 374
Hinkle 399.01 400
Hull 425.99 426

The FLOOR function returns the next smallest or equal integer value of the argument. The argument must be numeric.

The smallest/equal integer will be returned from the argument Pay in the program that follows.

Input: IP.Data
Name Pay
Graham 374.00
Hinkle 399.01
Hull 425.99

Program

Data Out;
   Set IP.Data;
   Low_Amt = FLOOR(Pay);
Run;

Output: Out
Name Pay Low_Amt
Graham 374.00 374
Hinkle 399.01 399
Hull 425.99 425
The ROUND function rounds a value to the nearest roundoff unit. The following program will round the argument Pay to the hundreds, tens, ones, and tenths.

**Program**

<table>
<thead>
<tr>
<th>Data Out</th>
<th>Set IP.Emphist;</th>
</tr>
</thead>
<tbody>
<tr>
<td>T1 = ROUND(Pay,100); T2 = ROUND(Pay,10); T3 = ROUND(Pay,1); T4 = ROUND(Pay,.1);</td>
<td></td>
</tr>
<tr>
<td>Run;</td>
<td></td>
</tr>
</tbody>
</table>

**Output:**

<table>
<thead>
<tr>
<th>Pay</th>
<th>T1</th>
<th>T2</th>
<th>T3</th>
<th>T4</th>
</tr>
</thead>
<tbody>
<tr>
<td>375.56</td>
<td>400</td>
<td>380</td>
<td>376</td>
<td>375.6</td>
</tr>
<tr>
<td>434.42</td>
<td>400</td>
<td>430</td>
<td>434</td>
<td>434.4</td>
</tr>
<tr>
<td>628.28</td>
<td>600</td>
<td>630</td>
<td>628</td>
<td>628.3</td>
</tr>
</tbody>
</table>

The SUM function calculates the sum of the arguments. Arguments must be numeric. **SUM** requires two or more arguments.

If variable names are numbered, the list of arguments can be abbreviated as shown in Program2. Results from Program1 and Program2 are the same.

**Program**

<table>
<thead>
<tr>
<th>Data Out</th>
<th>Set IP.Payroll;</th>
</tr>
</thead>
<tbody>
<tr>
<td>Tot = SUM(Day1,Day2,Day3,Day4,Day5);</td>
<td></td>
</tr>
<tr>
<td>Run;</td>
<td></td>
</tr>
</tbody>
</table>

**Output:**

<table>
<thead>
<tr>
<th>Empno</th>
<th>Day1</th>
<th>Day2</th>
<th>Day3</th>
<th>Day4</th>
<th>Day5</th>
<th>Tot</th>
</tr>
</thead>
<tbody>
<tr>
<td>40696</td>
<td>7</td>
<td>8</td>
<td>8</td>
<td>8</td>
<td>8</td>
<td>39</td>
</tr>
<tr>
<td>35788</td>
<td>9</td>
<td>8</td>
<td>8</td>
<td>10</td>
<td>5</td>
<td>40</td>
</tr>
<tr>
<td>39777</td>
<td>10</td>
<td>8</td>
<td>.</td>
<td>8</td>
<td>8</td>
<td>34</td>
</tr>
</tbody>
</table>

The INDEX function searches the first argument, from left to right, for the character string specified by the second argument. The start position of the string is returned. If the string is not found, **INDEX** returns a 0.

In the following program, the variable Jobtitle will be searched for the string 'Programmer/Analyst'. If the string is found, the observation will be written to the data set Out.

**Program**

<table>
<thead>
<tr>
<th>Data Out</th>
<th>Set IP.Pers;</th>
</tr>
</thead>
<tbody>
<tr>
<td>X = INDEX(Jobtitle,'Programmer/Analyst'); If X &gt; 0 Then Output Out;</td>
<td></td>
</tr>
<tr>
<td>Run;</td>
<td></td>
</tr>
</tbody>
</table>

**Output:**

<table>
<thead>
<tr>
<th>Name</th>
<th>Jobtitle</th>
</tr>
</thead>
<tbody>
<tr>
<td>Graham</td>
<td>Sr. Programmer/Analyst</td>
</tr>
<tr>
<td>Hinkle</td>
<td>Project Manager</td>
</tr>
<tr>
<td>Hull</td>
<td>Assoc. Programmer/Analyst</td>
</tr>
<tr>
<td>Jones</td>
<td>Data Entry Specialist</td>
</tr>
<tr>
<td>Munger</td>
<td>Data Entry Specialist</td>
</tr>
</tbody>
</table>

The SCAN function returns a given word from a character expression. The argument is separated into words by the defined delimiter, and the nth word is returned. If a delimiter is not specified, the following characters are default delimiters: `blank . < ( + | & | ! | $ | * ) ; ~ - / % \ €`

In the following program, a '" is used as the delimiter to separate Address into City, State and Zip.
Input: IP.Location
Address
San Diego/CA/92115
New York/NI/89123
Burlington/NJ/08060

Program
Data Out (Drop = Address);
Set IP.Location;
City = SCAN(Address,1,'/');
State = SCAN(Address,2,'/');
Zip = SCAN(Address,3,'/');
Run;

Output: Out
City State Zip
San Diego CA 92115
New York NY 89123
Burlington NJ 08060

SUBSTR(argument1,position,n)
The SUBSTR function extracts a substring of a value beginning from the character position indicated. The substring will be n characters long. If the length is omitted, the substring consists of the remainder of the character value.
The SUBSTR function is used in the next example to extract from the variable Partno a string starting at position 4 for a length of 5 characters.

Input: IP.Partfile
Partno
12-TR137-45
14-LA229-43
32-GM188-89

Program
Data Out;
Set IP.Partfile;
Code = SUBSTR(Partno,4,5);
Run;

Output: Out
Partno Code
12-TR137-45 TR137
14-LA229-43 LA229
32-GM188-89 GM188

TRIM(argument)
The TRIM function returns the argument with trailing blanks removed. TRIM is useful when concatenating,
because concatenation does not remove trailing blanks.

For example, the TRIM function is used in the following program to remove trailing blanks from the values of F_Name. Without the TRIM function, the values of Name have more than one space between the first and last name.

Input: IP.Profile
F_Name L_Name
DorcuGraham
Cliff Hinkle
Mildred Hull

Program
Data Out (Drop F_Name L_Name);
Set IP. Profile;
Name = F_Name || L_Name;
Tname = TRIM(F_Name) || ' ' || L_Name;
Run;

Output: Out
Name Tname
DorcuGraham DorcuGraham
Cliff Hinkle Cliff Hinkle
Mildred Hull Mildred Hull

TODAYO)
The TODAY function returns the current date as a SAS® date value (the number of days after or before January 1, 1960).
The statements
Data;
Curdate = TODAYO;
Put Curdate= Curdate= MMDDYY8.;
Run;
produce the line
Curdate= 10186 Curdate= 11/21/87
On November 21, 1987 the SAS® date value was 10186.

JULDATE(date)
The JULDATE function returns the Julian date from a SAS® date value.
The statements
Data;
  Currdate = TODAY();
  Julian = JULDATE(Currdate);
  Put Currdate=
  Julian=;
Run;
produce the line
Currdate= 10186  Julian= 87325
The Julian date 87325 is returned from Currdate's SAS® date value.

MONTH(date)

The MONTH function returns the month from a SAS® date value.
The statements
Data;
  Currdate = TODAY();
  Curr_Mo = MONTH(Currdate);
  Put Curr_Mo=;
Run;
produce the line
Curr_Mo= 11
(The current date is November 21, 1987).

DAY(date)

The DAY function returns the day from a SAS® date value.
The statements
Data;
  Currdate = TODAY();
  Curr_Day = DAY(Currdate);
  Put Curr_Day=;
Run;
produce the line
Curr_Day= 21
(The current date is November 21, 1987).

YEAR(date)

The YEAR function returns the year from a SAS® date value.
The statements
Data;
  Currdate = TODAY();
  Curr_Yr = YEAR(Currdate);
  Put Curr_Yr=;
Run;
produce the line
Currr_Yr= 1987
(The current date is November 21, 1987).

MDY(month,day,year)

The MDY function returns a SAS® date value from the month, day and year.
The statements
Data;
  Sasdate = MDY(11,21,87);
  Put Sasdate=;
Run;
produce the line
Sasdate= 10186

PUT(argument,format)

The PUT function specifies an output format for a value. The result is always a character string. The type (numeric/character) of the first argument and the format must be the same.
The following program demonstrates a table lookup using the variable Dept. If the department is found, the Put function returns the format specified. If not found, the string 'Invalid Dept' is returned.

Input: IP.Depart
Dept
  1953
  250
  5560
  5857

Program
Proc Format;
  Value $Dep '5650'-'5859' = 'DSD'
    '106'-'299' = 'Electronics'
    '1950'-'1959' = 'Convair'
    '250'-'350' = 'Pomona'
    Other = 'Invalid Dept';
Data Out;
  Set IP.Depart;
  Div = PUT(Dept,$Dep.);
Run;
The next example also demonstrates a table lookup but with a different twist. Observations are written out based upon the result of the table lookup.

**Input:** IP.Department

<table>
<thead>
<tr>
<th>Dept</th>
<th>Name</th>
</tr>
</thead>
<tbody>
<tr>
<td>1095</td>
<td>Graham</td>
</tr>
<tr>
<td>1107</td>
<td>Hinkle</td>
</tr>
<tr>
<td>4010</td>
<td>Hull</td>
</tr>
</tbody>
</table>

**Program**

```sas
Proc Format;
  Value Dep 1001-2222 = 'Jones'
               3001-4444 = 'Munger'
               Other = 'Error';
```

```sas
Data Jones Munger Error;
  Set IP.Department;
  If PUT(Dept,Dep.) = 'Jones' Then
    Output Jones;
  Else If PUT(Dept,Dep.) = 'Munger' Then
    Output Munger;
  Else If PUT(Dept,Dep.) = 'Error' Then
    Output Error;
  Run;
```

**Output:** Jones

<table>
<thead>
<tr>
<th>Dept</th>
<th>Name</th>
</tr>
</thead>
<tbody>
<tr>
<td>1095</td>
<td>Graham</td>
</tr>
<tr>
<td>1107</td>
<td>Hinkle</td>
</tr>
</tbody>
</table>

**Output:** Munger

<table>
<thead>
<tr>
<th>Dept</th>
<th>Name</th>
</tr>
</thead>
<tbody>
<tr>
<td>4010</td>
<td>Hull</td>
</tr>
</tbody>
</table>

**Conclusion**

By now your programming biceps should be rippling due to a high intake of SAS® Function samples. If you are not yet satiated, a vast array of functions are yet to be discovered in Chapter 6 of the SAS® User's Guide: Basics.

SAS is a registered trademark of SAS Institute Inc., Cary, NC, USA.