**FUNCTIONS EXAMPLES**

**CHARACTER FUNCTIONS**

/\*To extract partial text from a character value\*/

SUBSTR(variable, position<,length>)=characters-to-replace \*/

**data** lname;

input name & $25.;

cards;

CICHOCK, ELIZABETH MARIE

BENINCASA, HANNAH LEE

;

**run**;

**data** c;

set lname;

jj=substr(name,**2**,**5**);

**run**;

**data** lname;

input name &$25.;

cards;

CICHOCK, ELIZABETH MARIE

BENINCASA, HANNAH LEE

;

**run**;

**data** lname1;

set lname;

LastName=scan(name,**1**);

FirstName=scan(name,**2**);

MiddleName=scan(name,**3**);

;

**run**;

**proc** **print** data=lname1;

**run**;

**data** firstlast;

input String $60.;

First\_Word = scan(string, **1**);

Last\_Word = scan(string, -**1**);

datalines4;

Jack and Jill

& Bob & Carol & Ted & Alice &

Leonardo

! $ % & ( ) \* + , - . / ;

;;;;

**proc** **print** data=firstlast;

**run**;

data all;

length word $20;

drop string;

string = ' The quick brown fox jumps over the lazy dog. ';

do until(word=' ');

count+1;

word = scan(string, count);

output;

end;

run;

proc print data=all noobs;

run;

**data** xy;

a='KIDNAP';

substr(a,**1**,**3**)='CAT';

put a;

**run**;

**data** bb;

arg='ABC.DEF(X=Y)';

word=scan(arg,**3**);

\*word=scan(arg,3);

put word;

**run**;

**data** y;

a='AB C D ';

b=compress(a);

put b;

**run**;

**data** y;

a='AB C D ';

b=compress(trim(a));

put b;

**run**;

/\*Example Use COMPRESS to Remove Specified Characters\*/

**data** zipcode;

input zipcode$14.;

zipcode1 = compress(zipcode); /\* to remove blanks \*/

zipcode2 = compress(zipcode,' ()?'); /\* to remove blanks, () and ? \*/

zipcode3 = compress(zipcode,'- ()?'); /\* to remove dash, blanks, () and ? \*/

datalines;

22168- 12 34

22168- (1234?)

;

**proc** **print** data=zipcode noobs;

title2"Listing of Zipcodes";

**run**;

/\*TRANSLATE(source,to-1,from-1<,...to-n,from-n>)

source

specifies a character constant, variable, or expression that contains the original character string.

to

specifies the characters that you want TRANSLATE to use as substitutes.

from

specifies the characters that you want TRANSLATE to replace.

\*/

**data** z;

x=translate('XYZW','XY','VW');

put x;

**run**;

**proc** **print** data= z;

**run**;

**DATA** MULTIPLE;

INPUT QUES : $1. @@;

QUES = TRANSLATE(QUES,'ABCDE','12345');

DATALINES;

1 4 3 2 5

5 3 4 2 1

;

**PROC** **PRINT** DATA=MULTIPLE NOOBS;

TITLE "Listing of Data Set MULTIPLE";

**RUN**;

**data** n;

input name $;

cards;

ASDF

dg

dgfd

;

**run**;

**data** n1;

set n;

a=upcase(name);

**run**;

**data** n2;

set n;

a=lowcase(name);

**run**;

**data** n3;

set n;

a=propcase(name);

**run**;

/\*\*Removing Blanks From the Search String\*\*\*/

**data** list;

input salelist $;

length target $**10** replacement $**3**;

target='FISH';

replacement='NIP';

salelist=tranwrd(salelist,trim(target),replacement);

put salelist;

datalines;

CATFISH

;

**run**;

**proc** **print** data= list;

**run**;

DATA CONVERT;

INPUT @1 ADDRESS $20. ;

\*\*\* Convert Street, Avenue and Road to their abbreviations;

ADDRESS = TRANWRD(ADDRESS,'Street','St.');

ADDRESS = TRANWRD (ADDRESS,'Avenue','Ave.');

ADDRESS = TRANWRD (ADDRESS,'Road','Rd.');

DATALINES;

89 Lazy Brook Road

123 River Rd.

12 Main Street

;

PROC PRINT DATA=CONVERT;

TITLE 'Listing of Data Set CONVERT';

RUN;

**DATA** CONVERT;

INPUT @**1** ADDRESS $20. ;

DATALINES;

89 Lazy Brook Road

123 River Rd.

12 Main Street

;

**DATA** CONVERT1;

set convert ;

\*\*\* Convert Street, Avenue and Road to their abbreviations;

ADDRESS = TRANWRD(ADDRESS,'Street','St.');

ADDRESS = TRANWRD (ADDRESS,'Avenue','Ave.');

ADDRESS = TRANWRD (ADDRESS,'Road','Rd.');

**run**;

**PROC** **PRINT** DATA=CONVERT1;

**RUN**;

**data** \_null\_;

input @**1** name $20.;

name=compbl(name);

put name=;

datalines;

Amadeus Software

**run**;

/\*Example 1: Create Sample Data and Compare TRIM and TRIMN\*/

The TRIM and TRIMN functions are similar. TRIM returns one blank for a blank string. TRIMN returns a string with a length of zero for a blank string.\*/

/\* TRIMN Removes trailing blanks from character expressions and returns a null string (zero blanks) if the expression is missing\*/

**data** k;

x="A"||trimn("")||"B";

put x;

**run**;

**data** k;

x="A"||trim("")||"B";

put x;

**run**;

**data** k1;

x=" ";

z=">"||trimn(x)||"<";

put z;

**run**;

**data** k2;

x=" ";

z=">"||trim(x)||"<";

put z;

**run**;

**data** test;

input part1 $ **1**-**10** part2 $ **11**-**20**;

hasblank=part1||part2;

noblank=trim(part1)||part2;

put hasblank;

put noblank;

datalines;

abc xyz

;

**run**;

**data** sample;

input string $char14.;

datalines;

Mary Smith / contains trailing blanks /

John Brown / contains leading blanks /

Alice Park / contains leading and trailing blanks /

Tom Wang / contains leading, trailing and multiple blanks in between /

/ contains a blank string /

;

**data** sample;

set sample;

original = '\*' || string || '\*';

trim = '\*' || trim(string) || '\*' ;

trimn = '\*' || trimn(string) || '\*';

**run**;

**proc** **print** data=sample (drop=string) noobs;

title2'Output of TRIM and TRIMN';

**run**;

/\*Example 2: Compare STRIP, TRIM(LEFT) and TRIMN(LEFT)\*/

**data** sample1;

set sample;

strip = '\*' || strip(string) || '\*';

trim\_left = '\*' || trim(left(string)) || '\*';

trimn\_left = '\*' || trimn(left(string)) || '\*';

**run**;

**proc** **print** data=sample1 noobs;

title2'Output of STRIP, TRIM(LEFT) and TRIMN(LEFT)';

\*var original strip trim\_lefttrimn\_left;

**run**;

/\*Example 3.1: Use COMPRESS to Remove Blanks and Compare with COMPBL\*/

**data** sample;

set sample;

compress = '\*' || compress(string) || '\*';

compbl = '\*' || compbl(string) || '\*';

**run**;

**proc** **print** data=sample noobs;

title2'Output of COMPRESS and COMPBL';

var original compress compbl;

**run**;

/\*Example Compare CAT, CATT, CATS, CATX\*/

**data** sample;

input string $ string1 $;

datalines;

Mary Smith

John Brown

;

**data** sample;

set sample;

cat =cat (string,string1); /\* (= || --- Concatenate) \*/

catt=catt(string,string1); /\* (= TRIM || or TRIMN ||) \*/

cats=cats(string,string1); /\* (= STRIP ||)) \*/

catx=catx('-',string,string1); /\* (= STRIP || separator) \*/

**run**;

**proc** **print** data=sample noobs;

**run**;

/\*Example CATX\*/

data \_null\_;

separator='%%$%%';

x='The Olympic ';

y=' Arts Festival ';

z=' includes works by ';

a='Dale Chihuly.';

result=catx(separator,x,y,z,a);

put result $char.;

run;

/\*Example 4.1: Compare CAT, CATT, CATS, CATX\*/

**data** sample;

set sample;

length cat catt cats $**16** catx $**20**;

text='Hello';

cat =cat ('\*',string,'\*'); /\* (= ||) \*/

catt=catt('\*',string,'\*'); /\* (= TRIM || or TRIMN ||) \*/

cats=cats('\*',string,'\*'); /\* (= STRIP ||)) \*/

catx=catx('!',text,string); /\* (= STRIP || separator) \*/

**run**;

**proc** **print** data=sample noobs;

var cat catt cats catx;

title2"Output of Concatenation Functions";

**run**;

/\*INPUT -to convert a character value into a numeric value\*/

/\*input(string,informat)

**data** testin;

input sale $9.;

fmtsale=input(sale,comma9.);

datalines;

2,115,353

;

**proc** **print** data=testin;

**run**;

**proc** **contents** data=testin;

**run**;

/\* PUT -to convert a anumeric value into a character value\*/

put(string,format)\*/

**data** test;

input name ;

fmtname=put(name,**5.**);

datalines;

100

;

**proc** **print** data=test;

**run**;

**proc** **contents** data=test;

**run**;

**data** ex;

numdate=**122591**;

chardate=put(numdate,z6.);

sasdate=input(chardate,mmddyy6.);

put chardate= sasdate=;

**run**;

/\* Searches a character expression for a string of characters, and returns the position of the string's first character for the first occurrence of the string. \*/

**INDEX**(source,excerpt)

/\*To identify the position where the specified text is found from a character value\*/

source

specifies a character constant, variable, or expression to search.

excerpt

is a character constant, variable, or expression that specifies the string of characters to search for in source.

**data** test;

b='have a good day';

x=index(b,'good');

put x;

**run**;

**data** \_null\_;

a = 'ABC.DEF(X=Y)';

b = 'X=Y';

x = index(a,b);

put x=;

**run**;

options nodate nostimer ls=78 ps=60;

data \_null\_;

length a b $14;

a='ABC.DEF (X=Y)';

b='X=Y';

q=index(a,b);

w=index(a,trim(b));

put q= w=;

run;

/\*The INDEXC function searches for the first occurrence of any individual character that is present within the character string, whereas the INDEX function searches for the first occurrence of the character string as a substring. The FINDC function provides more options. \*/

**data** test;

a='ABC.DEP (X2=Y1)';

x=indexc(a,'0123',';()=.');

put x;

**run**;

**data** test;

b='have a good day';

x=indexc(b,'pleasant','very');

put x;

**run**;

**DATA** MIXED\_DATES;

INPUT @**1** DUMMY $15. ;

IF INDEXC(DUMMY,'/-:') ;

DATE = INPUT(DUMMY,MMDDYY10.);

FORMAT DATE DATE9.;

DATALINES;

10/21/1946

5-10-1950

7:9:57

;

**PROC** **PRINT** DATA=MIXED\_DATES NOOBS;

TITLE "Listing of Data Set MIXED\_DATES";

VAR DATE;

**RUN**;

**DATA** MIXED\_DATES;

INPUT @**1** DUMMY $15. ;

IF INDEXC(DUMMY,'/-:') ;

DATE = INPUT(DUMMY,MMDDYY10.);

FORMAT DATE DATE9.;

DATALINES;

10/21/1946

06JUN2002

5-10-1950

7:9:57

;

**PROC** **PRINT** DATA=MIXED\_DATES NOOBS;

TITLE "Listing of Data Set MIXED\_DATES";

VAR DATE;

**RUN**;

**DATA** MIXED\_DATES;

INPUT @**1** DUMMY $15.;

IF INDEXC(DUMMY,'/-:') NE **0** THEN DATE = INPUT(DUMMY,MMDDYY10.);

ELSE DATE = INPUT(DUMMY,DATE9.);

FORMAT DATE WORDDATE.;

DROP DUMMY;

DATALINES;

10/21/1946

06JUN2002

5-10-1950

7:9:57

;

**PROC** **PRINT** DATA=MIXED\_DATES NOOBS;

TITLE "Listing of Data Set MIXED\_DATES";

VAR DATE;

**RUN**;

**NUMERIC FUNCTIONS**

/\*abs function

The ABS function returns a nonnegative number that is equal in magnitude to the magnitude of the argument.\*/

**data** have;

input var1;

var2=abs(var1);

cards;

1

-2

3

-4

-2.5

;

**proc** **print**;**run**;

/\*The INT function truncates the decimal portion of the value of the argument.

The integer portion of the value of the argument remains.

The INT function takes the integer value of each element of the argument matrix. \*/

**data** a;

c=**2.8**;

b=int(c);

**run**;

**proc** **print**;**run**;

**data** a;

c=**2.8**;

c1=-**2.8**;

b=int(c);

c=int(c1);

d=abs(int(c1));

**run**;

**proc** **print**;**run**;

/\*Returns the remainder from the division of the first argument by the second argument,

fuzzed to avoid most unexpected floating-point results.\*/

**data** b;

c = (**14**);

b=mod(c,**4**);

**run**;

**proc** **print**;**run**;

**data** b;

c = (-**9**);

b=mod(c,**4**);

**run**;

**proc** **print**;**run**;

**data** aa;

x1=mod(**10**,**3**);

**run**;

**proc** **print**;**run**;

**data** c;

c=**2.46**;

b=round(c);

**run**;

**proc** **print**;**run**;

**data** cc;

c=**2.5**;

b=round(c);

**run**;

**proc** **print**;**run**;

**data** ccc;

c=**2.65**;

b=round(c);

**run**;

**proc** **print**;**run**;

/\*Using the floor function to always round down to the nearest integer

Returns the smallest integer that is less than or equal to the argument, fuzzed to avoid unexpected floating-point results.\*/

**data** d;

input num;

round\_down=floor(num);

datalines;

123.456

123.789

34

5.1111

7.68

10.98

-3.1

-19.0

-4.8

;

**proc** **print**;

**run**;

/\*Using the CEIL function to always round up to the nearest integer \*/

/\*Returns the largest integer that is less than or equal to the argument, fuzzed to avoid unexpected floating-point results.\*/

**data** f ;

input num;

round\_up=ceil(num);

datalines;

123.456

34

5.1111

7.68

10.98

-3.1

-19.0

-4.8

;

**proc** **print**;

**run**;

/\*RAND - Generates random numbers from a distribution that you specify. \*/

/\*The RAND function generates random numbers from various continuous and discrete distributions.

Wherever possible, the simplest form of the distribution is used.\*/

/\*x --is an observation from the distribution with the following probability density function:

Range: 0 < x < 1

The uniform random number generator that the RAND function uses is the Mersenne-Twister (Matsumoto and Nishimura 1998).

This generator has a period of [equation] and 623-dimensional equidistribution up to 32-bit accuracy.

This algorithm underlies the generators for the other available distributions in the RAND function.\*/

**data** a;

x=rand('UNIFORM');

put x;

**run**;

/\*x --is an observation from the distribution with the following probability density function:

Range: x > 0 \*/

**data** b;

x=rand('EXPO');

put x;

**run**;

/\*LOG -to convert the numeric value into logarithm\*/

**data** a;

x=**30**;

b=log(x);

**run**;

**AGGREGATE FUNCTIONS**

**data** g;

xyz='This is a thistle? Yes, this is a thistle.';

howmanythis=count(xyz,'this');

put howmanythis;

**run**;

**proc** **print**;

**run**;

**data** g1;

xyz='This is a thistle? Yes, this is a thistle.';

howmanythis=count(xyz,'This','this');

put 'howmanythis ' howmanythis;

**run**;

**proc** **print**;

**run**;

**data** g;

input name & $35. @**37** age;

cards;

This is a thistle this is a thistle 34

thistle this is a thistle 55

;

**run**;

**data** b;

set g;

howmanythis=count(name,'this');

put howmanythis;

**run**;

**proc** **print**;

**run**;

/\*Returns the sum of the nonmissing arguments.\*/

**data** h;

x2=sum(**4**,**9**,**3**,**8**,**.**);

**run**;

**proc** **print**;

**run**;

**data** h;

x1=**9**;

x2=**39**;

x3=**4**;

x4=sum(of x1-x3);

**run**;

**proc** **print**;

**run**;

**data** i;

x2=min(**4**,**9**,**3**,**8**,**.**);

**run**;

**proc** **print**;

**run**;

**data** ii;

x2=min(**4**,**9**,**3**,**8**,**0**,**.**);

**run**;

**proc** **print**;

**run**;

**data** j;

x2=max(**4**,**9**,**3**,**8**,**.**);

**run**;

**proc** **print**;

**run**;

**data** k;

x2=mean(**4**,**9**,**3**,**8**,**.**);

**run**;

**proc** **print**;

**run**;

/\*To compute the mean, minimum and maximum of a set of numeric values\*/

**data** test;

a=**10**;

b=**20**;

var1=mean(a,b);

var2=min(a,b);

var3=max(a,b);

**run**;

**proc** **print** data=test;**run**;

**DATE FUNCTIONS**

**data** a;

X=today();

put X date9.;

**run**;

**proc** **print**;

format x date9.;

**run**;

**data** b;

a=**'24aug2010'd**;

X=year(a);

put X ;

**run**;

**data** c;

a=**'24aug2010'd**;

X=month(a);

put X;

**run**;

**data** d;

a=**'24aug2010'd**;

X=day(a);

put X;

**run**;

**data** e;

birthday=mdy(**8**,**27**,**90**);

put birthday;

put birthday= worddate.;

**run**;

**proc** **print**;

format birthday worddate.;

**run**;

/\*

Converts a Julian date to a SAS date value.

julian-date

specifies a SAS numeric expression that represents a Julian date.

A Julian date in SAS is a date in the form yyddd or yyyyddd,

where yy or yyyy is a two-digit or four-digit integer that represents the year and ddd is the number of the day of the year.

The value of ddd must be between 1 and 365 (or 366 for a leap year).

\*/

**data** d;

bc=datejul(**60005**);

put bc / Xstart date9.;

**run**;

**data** e;

Xend=datejul(**2001001**);

put Xend / Xend date9.;

**run**;

/\*

Returns the Julian date from a SAS date value.

A SAS date value is a number that represents the number of days from January 1, 1960 to a specific date.

The JULDATE function converts a SAS date value to a Julian date.

If date falls within the 100-year span defined by the system option YEARCUTOFF=,

the result has three, four, or five digits. In a five digit result, the first two digits represent the year,

and the next three digits represent the day of the year (1 to 365, or 1 to 366 for leap years).

Because leading zeros are dropped from the result, the year portion of a Julian date might be omitted

(for years ending in 00), or it might have only one digit (for years ending 01-09).

Otherwise, the result has seven digits: the first four digits represent the year, and the next three digits

represent the day of the year. For example, if YEARCUTOFF=1920, JULDATE would return 97001 for January 1, 1997,

and return 1878365 for December 31, 1878.

\*/

**data** f;

julian=juldate(**'31dec99'd**);

put julian / julian date9.;

**run**;

**data** ff;

julian=juldate(**'01jan2099:hh:mm:ss'dt**);

put julian / julian date9.;

**run**;

/\*Specifies the first year of a 100-year span that is used by date informats and functions to read a two-digit year.

YEARCUTOFF= System Option

YEARCUTOFF= nnnn | nnnnn

nnnn | nnnnn

specifies the first year of the 100-year span.

The YEARCUTOFF= value is the default that is used by various date and datetime informats and functions.

If the default value of nnnn (1920) is in effect, the 100-year span begins with 1920 and ends with 2019. Therefore, any informat or function that uses a two-digit year value that ranges from 20 to 99 assumes a prefix of 19. For example, the value 92 refers to the year 1992.

The value that you specify in YEARCUTOFF= can result in a range of years that span two centuries. For example, if you specify YEARCUTOFF=1950, any two-digit value between 50 and 99 inclusive refers to the first half of the 100-year span, which is in the 1900s. Any two-digit value between 00 and 49, inclusive, refers to the second half of the 100-year span, which is in the 2000s. The following figure illustrates the relationship between the 100-year span and the two centuries if YEARCUTOFF=1950.

A 100-Year Span with Values in Two Centuries

\*/

**INTCK function**

/\*The INTCK function counts the number of interval boundaries between two dates or between two datetime values.

INTCK( interval, from, to ) \*/

**data** a;

qtr=intck('qtr',**'10jan95'd**,**'30oct95'd**);

put qtr;

**run**;

**data** b;

year=intck('year',**'31dec94'd**,**'01jan95'd**);

put year;

**run**;

**data** c;

year=intck('year',**'01jan94'd**,**'31dec94'd**);

put year;

**run**;

**data** d;

semi=intck('semiyear',**'01jan95'd**,**'01jan98'd**);

put semi;

**run**;

**data** kk;

years=intck('year',**'01jan2009'd**,**'01jan2010'd**);

SEMIYEAR=intck('SEMIYEAR',**'01jan2009'd**,**'01jan2010'd**);

quarters=intck('qtr',**'01jan2009'd**,**'01jan2010'd**);

months=intck('month',**'01jan2009'd**,**'01jan2010'd**);

weeks=intck('week3.2',**'01jan2009'd**,**'01jan2010'd**);

days=intck('day',**'01jan2009'd**,**'01jan2010'd**);

**run**;

**proc** **print** data=kk;

**run**;

**data** e;

weekvar=intck('week3.2',**'04jul2016'd**,**'30Jul2016'd**);

put weekvar;

**run**;

**data** f;

weekvar=intck('week3.2',**'01jul2016'd**,**'30Jul2016'd**);

put weekvar;

**run**;

**data** f;

wdvar=intck('weekday7w',**'01jan97'd**,

**'01feb97'd**);

put wdvar;

**run**;

**data** g;

y='year';

date1=**'1sep1991'd**;

date2=**'1sep2001'd**;

newyears=intck(y,date1,date2);

put newyears;

**run**;

**data** h;

y=trim('year ');

date1=**'1sep1991'd** + **300**;

date2=**'1sep2001'd** - **300**;

newyears=intck(y,date1,date2);

put newyears;

**run**;

**INTNX function**

/\*The INTNX function increments dates by intervals.

INTNX computes the date or datetime of the start of the interval a specified

number of intervals from the interval containing a given date or datetime value.

INTNX( interval, from, n <, alignment > ) \*/

**data** a;

w=intnx('week', **'17oct03'd**, **6**);

format w date7.;

put w;

**run**;

**data** b;

x=intnx('month', **'15mar2000'd**, **5** , 'sameday'); \* returns 15AUG2000;

y=intnx('year', **'29feb2000'd**, **2** , 'sameday'); \* returns 28FEB2002;

z=intnx('month', **'31aug2001'd**, **1**, 'sameday'); \* returns 30SEP2001;

format x y z date7.;

put x y z;

**run**;

**data** c;

yr=intnx('year',**'05feb94'd**,**3**);

put yr / yr date7.;

**run**;

**data** d;

x=intnx('month',**'05jan95'd**,**0**);

put x / x date7.;

next=intnx('semiyear',**'01jan97'd**,**1**);

put next / next date7.;

past=intnx('month2',**'01aug96'd**,-**1**);

put past / past date7.;

sm=intnx('semimonth2.2',**'01apr97'd**,**4**);

put sm / sm date7.;

m='month';

date=**'1jun1990'd**;

nextmon=intnx(m,date,**1**);

put nextmon / nextmon date7.;

x1='month ';

x2=trim(x1);

date=**'1jun1990'd** - **100**;

nextmonth=intnx(x2,date,**1**);

put nextmonth / nextmonth date7.;

**run**;

**data** e;

date1=intnx('month',**'01jan95'd**,**5**,'beginning');

put date1 / date1 date7.;

date2=intnx('month',**'01jan95'd**,**5**,'middle');

put date2 / date2 date7.;

date3=intnx('month',**'01jan95'd**,**5**,'end');

put date3 / date3 date7.;

date4=intnx('month',**'01jan95'd**,**5**,'sameday');

put date4 / date4 date7.;

date5=intnx('month',**'15mar2000'd**,**5**,'same');

put date5 / date5 date9.;

interval='month';

date=**'1sep2001'd**;

align='m';

date4=intnx(interval,date,**2**,align);

put date4 / date4 date7.;

x1='month ';

x2=trim(x1);

date=**'1sep2001'd** + **90**;

date5=intnx(x2,date,**2**,'m');

put date5 / date5 date7.;

**run**;