STA 311 Statistical Computing & Data Management

Instructor: Cheng Peng, Ph.D.
Department of Mathematics
West Chester University
West Chester, PA 19383

Office: 25 University Avenue, RM 111
Phone: 610.436.2369
Email: cpeng@wcupa.edu

Topic 10. SAS String Functions, Loops & Longitudinal Data

Topics for This Week

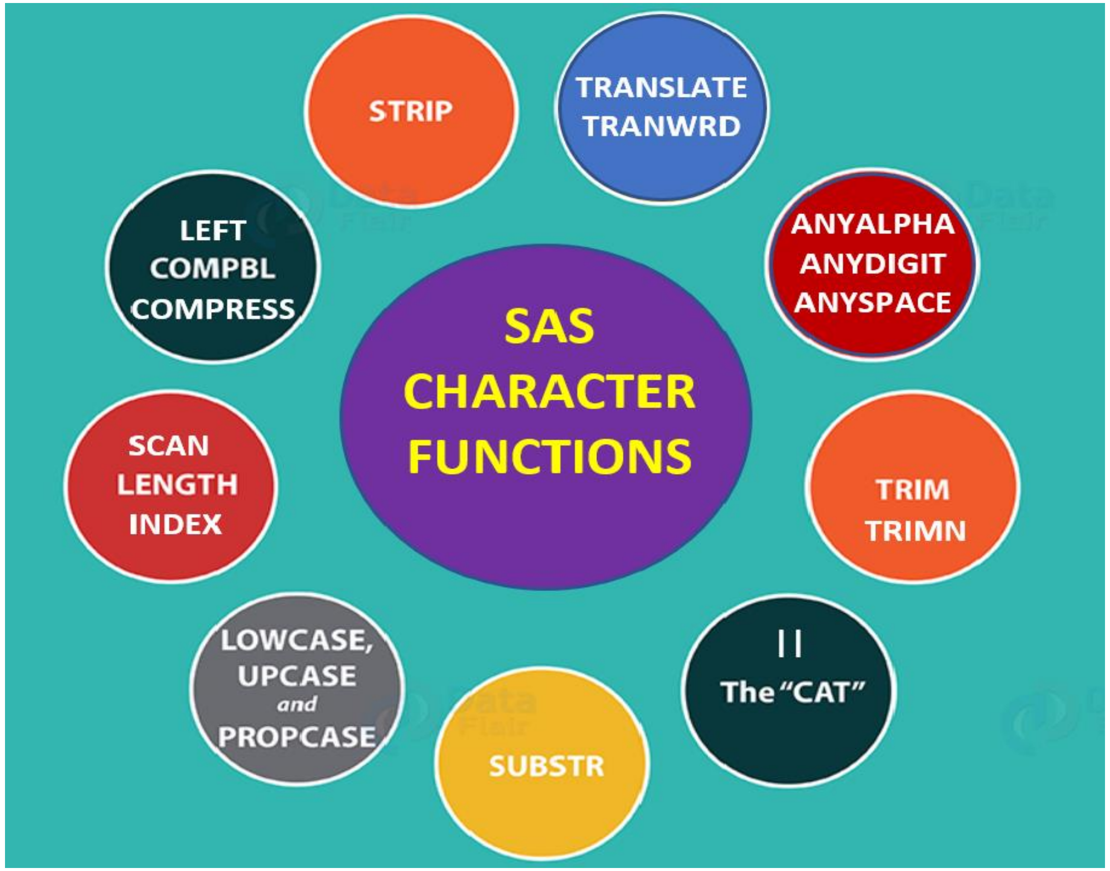
Topics

1. SAS String Functions
2. SAS Loop
3. RETAIN statement
4. Operations within Groups in Longitudinal Data

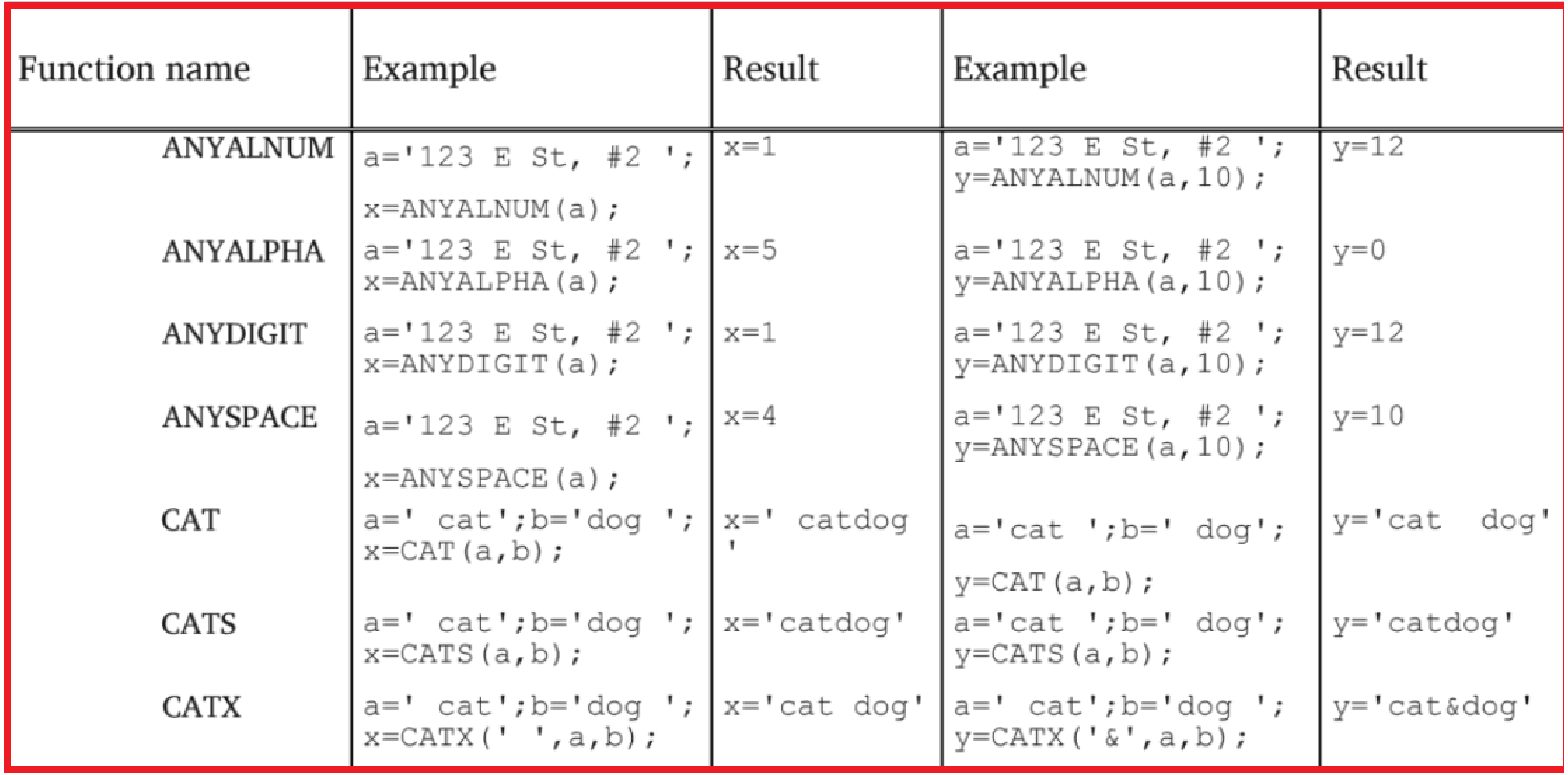
String Functions: Overview

There are many character functions in SAS for string manipulation. Here are a few commonly used ones.

Explanations and examples of using these function are given in the few slides



String Functions: Overview



Function name	Example	Result	Example	Result

ANYALNUM	`a='123 E St, #2 ';`
	`x=ANYALNUM(a);`
		`x=1`	`a='123 E St, #2 ';`
			`y=ANYALNUM(a,10);`
				`y=12`

ANYALPHA	`a='123 E St, #2 ';`
	`x=ANYALPHA(a);`
		`x=5`	`a='123 E St, #2 ';`
			`y=ANYALPHA(a,10);`
				`y=0`

ANYDIGIT	`a='123 E St, #2 ';`
	`x=ANYDIGIT(a);`
		`x=1`	`a='123 E St, #2 ';`
			`y=ANYDIGIT(a,10);`
				`y=12`

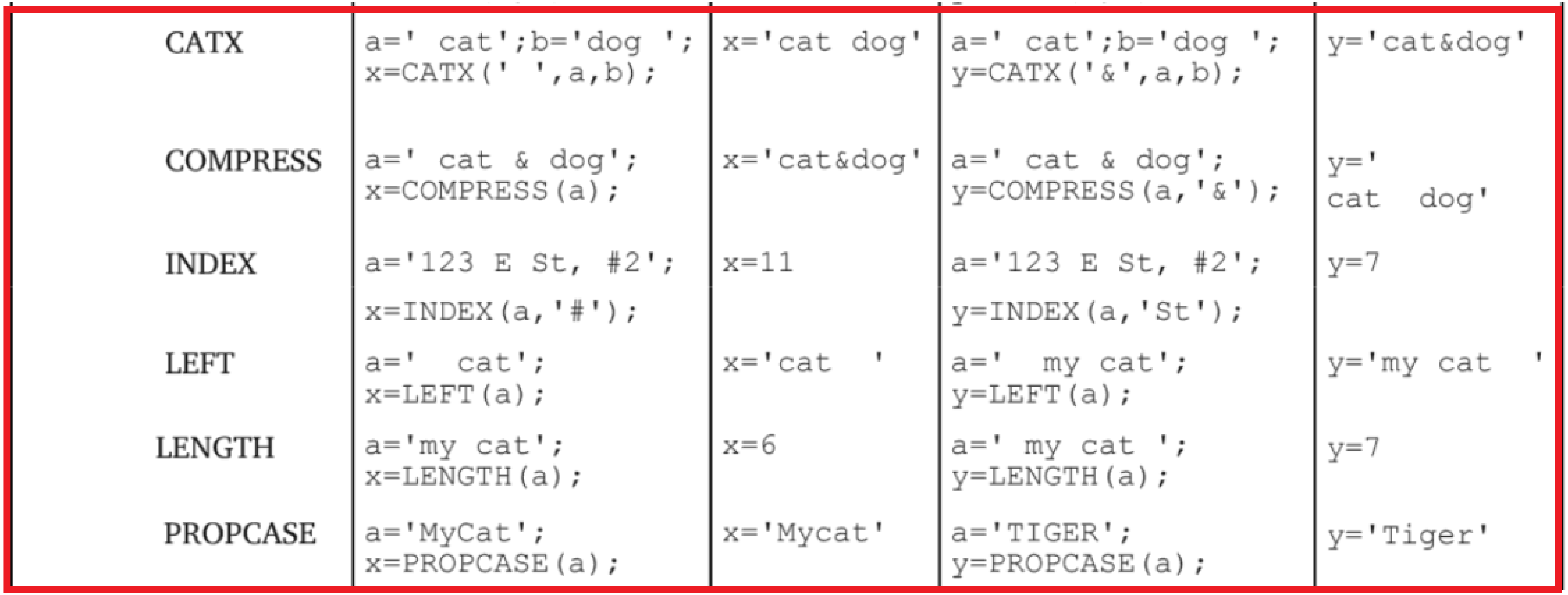
ANYSPACE	`a='123 E St, #2 ';`
	`x=ANYSPACE(a);`
		`x=4`	`a='123 E St, #2 ';`
			`y=ANYSPACE(a,10);`
				`y=10`

CAT	`a=' cat';b='dog ';`
	`x=CAT(a,b);`
		`x=' catdog '`	`a='cat ';b=' dog';`
			`y=CAT(a,b);`
				`y='cat  dog'`

CATS	`a=' cat';b='dog ';`
	`x=CATS(a,b);`
		`x='catdog'`	`a='cat ';b=' dog';`
			`y=CATS(a,b);`
				`y='catdog'`

CATX	`a=' cat';b='dog ';`
	`x=CATX(' ',a,b);`
		`x='cat dog'`	`a=' cat';b='dog ';`
			`y=CATX('&',a,b);`
				`y='cat&dog'`

String Functions: Overview



CATX	`a=' cat';b='dog ';`
	`x=CATX(' ',a,b);`
		`x='cat dog'`	`a=' cat';b='dog ';`
			`y=CATX('&',a,b);`
				`y='cat&dog'`

COMPRESS	`a=' cat & dog';`
	`x=COMPRESS(a);`
		`x='cat&dog'`	`a=' cat & dog';`
			`y=COMPRESS(a,'&');`
				`y='`
				`cat   dog'`

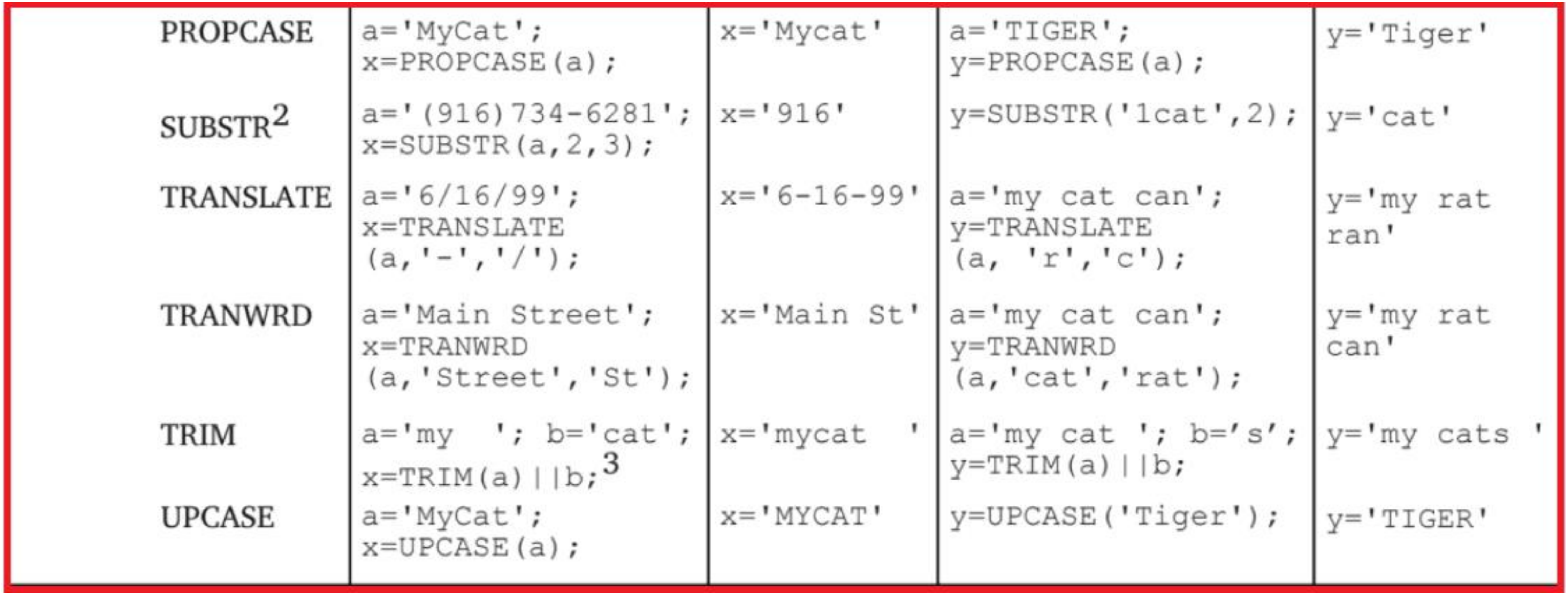
INDEX	`a='123 E St, #2';`
	`x=INDEX(a,'#');`
		`x=11`	`a='123 E St, #2';`
			`y=INDEX(a,'St');`
				`y=7`

LEFT	`a='  cat';`
	`x=LEFT(a);`
		`x='cat  '`	`a='  my cat';`
			`y=LEFT(a);`
				`y='my cat  '`

LENGTH	`a='my cat';`
	`x=LENGTH(a);`
		`x=6`	`a=' my cat ';`
			`y=LENGTH(a);`
				`y=7`

PROPCASE	`a='MyCat';`
	`x=PROPCASE(a);`
		`x='Mycat'`	`a='TIGER';`
			`y=PROPCASE(a);`
				`y='Tiger'`

String Functions: Overview



PROPCASE	a='MyCat';
	x=PROPCASE(a);
		x='Mycat'	a='TIGER';
			y=PROPCASE(a);
				y='Tiger'

SUBSTR[2]	a='(916)734-6281';
	x=SUBSTR(a,2,3);
		x='916'	y=SUBSTR('1cat',2);	y='cat'

TRANSLATE	a='6/16/99';
	x=TRANSLATE
	(a,'-','/');
		x='6-16-99'	a='my cat can';
			y=TRANSLATE
			(a, 'r','c');
				y='my rat
				ran'

TRANWRD	a='Main Street';
	x=TRANWRD
	(a,'Street','St');
		x='Main St'	a='my cat can';
			y=TRANWRD
			(a,'cat','rat');
				y='my rat
				can'

TRIM	a='my '; b='cat';
	x=TRIM(a)||b;[3]
		x='mycat '	a='my cat '; b=’s’;
			y=TRIM(a)||b;
				y='my cats '

UPCASE	a='MyCat';
	x=UPCASE(a);
		x='MYCAT'	y=UPCASE('Tiger');	y='TIGER'

Table 1 Examples - TEXT=the abc 123/%!( )?+ qWerty § Ref 19.3-20,9 __

#	SAS code	Will make TEXT2 result in[*]	Explanation
1	Text2=compress(Text);	theabc123/%!(▨)?+qWerty§Ref19.3-20,9__	Removes the blanks but not the tabulation (the old functionality)
2	Text2=compress(Text,,'kw') ;	the abc 123/%!()?+ qWerty § Ref 19.3-20,9 __	To remove tabulations, we can use a combination of two modifiers k (keep) and w (writable)
3	Text2=compress(Text,,'w');	▨	Using only the w as a modifier will keep tabulations, character returns etc. that we cannot see.
4	Text2=compress(Text,,'ka');	theabcqWertyRef	ka as a modifier makes us keep all alphabetic characters
5	Text2=compress(Text,' ','ka');	the abc qWerty Ref	To keep alphabetic characters and spaces add the ' ' to the second argument
6	Text2=compress(Text,,'a');	123/%!(▨)?+ § 19.3-20,9 __	a as a modifier is used to remove all alphabetic characters
7	Text2=compress(Text,,'ad');	/%!(▨)?+ § .-, __	ad as a modifier is used to remove all alphabetic characters together with digits
8	Text2=compress(Text,,'kpd');	123/%!()?+§19.3-20,9__	kpd as a modifier is used to keep all punctuations and digits
9	Text2=compress(Text,',./- ','kd');	123/ 19.3-20,9	With kd as a modifier is used to keep all digits
10	Text2=compress(Text,'0123456789,./ ','k');	123/ 19.3-20,9	With k as a modifier you can specify the exact characters/digit to keep. Adding k as a modifier can be seen as the opposite of the old functionality
11	Text2=compress(Text,,'u');	the abc 123/%!(▨)?+ qerty § ef 19.3-20,9 __	With u as the modifier, all uppercase letters are removed
12	Text2=compress(Text,'','ku');	W R	With ku as the modifier and '' specified as second argument, uppercase letters and spaces are kept
13	Text2=compress(Text,,'ku');	WR	With ku as the modifier and no second argument, ONLY uppercase letters are kept

SAS Loop: Overview

Designate a group of statements to be executed as a unit using a DO block. The following are general syntaxes

```
DO;
    SAS Statements;
END;
```

```
DO var=1 TO x;
    SAS Statements;
END;
```

```
DO UNITL ( condition);
    SAS Statements;
END;
```

```
DO WHILE ( condition);
    SAS Statements;
END;
```

The difference between the two DO loops is that DO UNTIL statement tests At the bottom of the of the loop and DO WHILE statements tests at the top.

SAS Loop: Iterative Do Loops

```
DO index-variable=start TO stop BY increment;
      SAS statements
END
```

```
DATA DOLoop;        /* SAS dataset  */
    DO i = 1 to 5;  /* I will be a variable in the SAS data set*/
        Y = i**2;   /* values are 1, 4, 9, 16, 25 , Y will be
                       another variable in the data set.*/
       OUTPUT;
    END;
 RUN;
```

SAS Loop: Iterative Do Loops

By default, each iteration of a DO statement increments the value of the counter by 1, but you can use the BY option to increment the counter by other amounts, including non-integer amounts. For example, each iteration of the following DATA step increments the value i by 0.5:

```
DATA DOLoopBy;     /* SAS dataset  */
    DO i = 1 to 5 by 0.5;    /* I will be a variable in the SAS data set*/
        Y = i**2;                /* values are 1, 1.5^2, 2^2 2.5^2,... Y will be
                                    another variable in the data set.    */
       OUTPUT;
    END;
 RUN;
```

SAS Loop: DO-WHILE Loop

By default, each iteration of a DO statement increments the value of the counter by 1, but you can use the BY option to increment the counter by other amounts, including non-integer amounts. For example, each iteration of the following DATA step increments the value i by 0.5:

```
DATA DOLoopBy;      /* SAS dataset  */
    DO i = 1 to 5 WHILE (y < 20);    /*  I will be a variable in the SAS data set*/
        Y = i**2;                       /*  values are 1, 4, 9, 16, 25 , Y will be
                                            another variable in the data set.        */
       OUTPUT;
    END;
 RUN;
```

SAS Loop: DO-UNTIL Loop

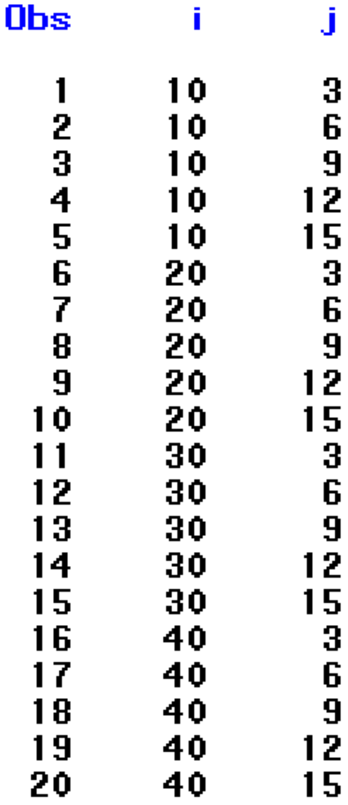
Using a DO UNTIL loop, SAS executes the DO loop **until** the expression you've specified is true

```
DATA investment;
    DO UNTIL (value >= 50000);
        value + 1200;
        value + value * 0.05;
        year + 1;
        OUTPUT;
    END;
RUN;
```

SAS Loop: Double Loop

Nested Loop – 3-by-5 factorial design

```
DATA design;
 DO i = 10 to 40 by 10;        /*     I = 10, 20, 30, 40      */
     DO j = 3 to 15 BY 3;      /*     j = 3, 6, 9, 12,15      */
         OUTPUT;               /*  output the value from
     END;                          each iteration             */
 END;
RUN;
```



```
Obs     i      j

  1    10      3
  2    10      6
  3    10      9
  4    10     12
  5    10     15
  6    20      3
  7    20      6
  8    20      9
  9    20     12
 10    20     15
 11    30      3
 12    30      6
 13    30      9
 14    30     12
 15    30     15
 16    40      3
 17    40      6
 18    40      9
 19    40     12
 20    40     15
```

SAS RETAIN Statement

RETAIN in SAS

- ❑ The **RETAIN** statement simply copies retaining values by telling the SAS not to reset the variables to **missing** at the beginning of each iteration of the DATA step.
- ❑ If the **RETAIN** statement is NOT used, SAS will return a missing value at the beginning of each iteration.

RETAIN + DO-Loop

Operations Within Groups in Longitudinal Data

```
DATA base;
INPUT id $
      sales
      vis_date mmddyy10.;
DATALINES;
a 235 07/11/1997
a 324 11/12/1997
b 321 06/15/1998
b 319 09/21/1998
b 357 11/11/1998
c 279 07/21/1997
c 302 10/20/1997
c 314 11/19/1997
c 298 12/27/1997
;
RUN;
```

Typical Longitudinal Data Set

3 salespersons in this data

Tasks

- ❑ **Count the number of sales per person**
- ❑ **Calculate the total sales per person**
- ❑ **Calculate average sales per person**

```
DATA new;
SET base;
BY id;
RETAIN count total;
IF FIRST.id THEN DO;
    count = 0;
    total = 0;
END;
count = count + 1;
total = total + sales;
IF LAST.id THEN DO;
    mean = total / count;
    OUTPUT;
END;

PROC PRINT DATA=new; RUN;
```