```
* SAS Strings and Arrays;
/*
Strings in SAS are the values which are enclosed with in a pair of single quotes.
Also the string variables are declared by adding a space and $ sign at
the end of the variable declaration.
*/
* Declaring String Variables;
/*
In the code below we declare two character variables of lengths 6 and 5.
The LENGTH keyword is used for declaring variables
without creating multiple observations.
*/
data string_examples;
   LENGTH string1 $ 6 String2 $ 5;
   /*String variables of length 6 and 5 */
   String1 = 'Hello';
   String2 = 'World';
   Joined_strings =  String1 ||String2 ;
run;
proc print data=string_examples noobs;
run;
*noobs is used to hide the index in the result table;

*String Functions;
/*
SUBSTRN : This function extracts a substring using the start and end positions.
Syntax:
SUBSTRN('stringval',p1,p2)
p1 is the start position of extraction.
p2 is the final position of extraction.
*/

data string_example;
    length string1 $ 10;
    string1 = 'Kirankumar';
    sub_string1 = SUBSTRN(string1,1,5);
run;
proc print data=string_example noobs;
run;

*TRIMN : This function removes the trailing space form a string.;
/*
Syntax
TRIMN('stringval')
*/
data trim_example;
    length string1 $ 10 ;
    String1 = 'Kiran ';
    length_1 = lengthc(String1);
    length_2 = lengthc(TRIMN(String1));
run;
proc print data = trim_example noobs;
run;

*Arrays;
/*
Arrays in SAS are used to store and retrieve a series of values using
an index value. The index represents the location in a reserved memory area.

Syntax:
ARRAY ARRAY-NAME(SUBSCRIPT) ($) VARIABLE-LIST ARRAY-VALUES



    ARRAY is the SAS keyword to declare an array.

    ARRAY-NAME is the name of the array which follows the same rule as variable names.

    SUBSCRIPT is the number of values the array is going to store.

    ($) is an optional parameter to be used only if the array is going to store character values.

    VARIABLE-LIST is the optional list of variables which are the place holders for array values.

    ARRAY-VALUES are the actual values that are stored in the array. They can be declared here or
    can be read from a file or dataline.
```

```
# Declare an array of length 5 named AGE with values.
ARRAY AGE[5] (12 18 5 62 44)

# Declare an array of length 5 named COUNTRIES with values starting at index 0.
ARRAY COUNTRIES(0:8) A B C D E F G H I

# Declare an array of length 5 named QUESTS which contain character values.
ARRAY QUESTS(1:5) $ Q1-Q5

# Declare an array of required length as per the number of values supplied.
ARRAY ANSWER(*) A1-A100;
*/
```

```
data array_example;
    input x1 x2 x3 x4;
    array x[4] x1-x4;
    sum = x1 + x2 + x3 +x4;
    mean = sum/4;
    datalines;
    1 2 3 4
    23 23 23 23
    ;
run;
```

```
data array2;
    input a0-a5;
    array country(*) a0-a5;
    datalines;
    1 2 3 4 5 4
    34 43 23 12 43 32
    ;
run;

*Using the OF operator;
/*
The OF operator is used when analysing the data from an Array to perform
calculations on the entire row of an array.
In the below example we apply the Sum and Mean of values in each row.
*/
```

```
data array_example;
    input x1 x2 x3 x4;
    array x[4] x1-x4;
    sum_array = SUM(OF x(*)) ;
    mean = MEAN(OF x(*));
    datalines;
    1 2 3 4
    23 23 23 23
    ;
run;

*Using the IN operator;
/*
The value in an array can also be accessed using the IN operator
which checks for the presence of a value in the row of the array.
*/
```

```
data array_example;
    input x1 x2 x3 x4;
    array x[4] x1-x4;
    sum = x1 + x2 + x3 +x4;
    mean = sum/4;
    if 23 IN x then age = 'Adult';else age = 'Child';
    datalines;
    1 2 3 4
    23 23 23 23
    ;
run;

*Last Line
```