```
/******************************************************
                       Week 02 Sample Code
    Author: C. Peng
    Date: 02/01/2021
    Updates: Fall 2020, Spring 2021
    Topics: Examples Illustrating the Following Concepts
      1. Program header;
      2. Display control;
      3. SAS display management: DM-statement
      4. SAS library - file/folder structure;
      5. Data Step and Procedure Step;
      6. Commenting and documentation;
      7. Conventions and naming rules;
      8. Debugging code - understand SAS log;
      9. Find SAS dataset structure: descriptor and data block
**********************************************/


*OOOOOOOOOOOOOOOOO;
**    Topic 1;
*OOOOOOOOOOOOOOOOO;
/*  Display management: clear log and output
    The following DM statement will automatically
    delete the content in the designated window!
*/
DM "CLEAR LOG";
DM "CLEAR OUT";


*OOOOOOOOOOOOOOOOOOO;
**      Topic 2;
*OOOOOOOOOOOOOOOOOOO;
*************************************************
  Set up a SAS library and save the created SAS
  dataset to the local computer - SAS permanent data
**************************************************;
```

```
LIBNAME sta311 "I:\Desktop\cpeng\WCU-Teaching\2020Fall\STA311\SAS";
   *NOTE: if you have SAS installed on your own computer,
          the path in the libname statement should be:
          LIBNAME sta311 "C:\STA311\Fall2020";


*OOOOOOOOOOOOOOOOOO;
**   Topic 3;
*OOOOOOOOOOOOOOOOOO;
/****************************************************************
  Display control: SAS system OPTIONS
  There many options avaialble in SAS that can be found by typing
  PROC OPTIONS;
  RUN;
  Most commonly used options are:
  pagesize, linesize, date, nodate, number, nonumber, etc. See SAS
  document for explanantions of these commonly used options:

https://documentation.sas.com/?docsetId=basess&docsetTarget=n1km315k2himgpn1j6d82y0yju4v.htm&docsetVersion=
9.4&locale=en
****************************************************************/
OPTIONS PS = 100  /* page size of log and output (listing) */
        LS = 90   /* line size of log and output (listing) */
            NODATE    /* turn off the date option in listing window */
            NONUMBER; /* SAS will not print page numbers in the output window */


*OOOOOOOOOOOOOOOOO;
**    Topic 4;
*OOOOOOOOOOOOOOOOO;
/**********************************************
 DATA step statements
 One way to create SAS dataset is to use DATA step
 statement. There are several different ways to read
 data in different formats. We will introduce the three
 most commonly used methods: column, list, and formatted
 methods. The following DATA step uses the column input.
 Components of a DATA STEP:

 DATA statement;
 INPUT statement;
 DATALINES (CARDS);
      data block
            ;
 RUN statement;

 The different input styles determine the types of input.
 The detailed information can be found at

https://documentation.sas.com/?docsetId=lestmtsref&docsetTarget=n0oaql83drile0n141pdacojq97s.htm&docsetVers
ion=9.4&locale=en#p0ert9pwdtsq4en1czu9tcrncgm8

 Column input illustration:
----+----1----+----2----+----3----
Florida    130 90


***********************************************/
/* This data step uses COLUMN INPUT style, More examples
      and explanations will be given in the next few weeks.
*/

DATA Orange;  /* send the SAS data set to the temporary library */
INPUT state $ 1-10 early 12-14 late 16-18;
     /*
     SAS has two basic types of data: numeric and categorical. $ is used
     to specify categorical data. As an example, we look at variable state:
     $ indicates that state is a categorical variable.
     1-10: all characters including white space in the first column define
           a string. That string is the value of the categorical variable STATE.
     */

/* DATALINES - tells SAS the data block begins.
   we can also use CARDS instead. CARDS was used
   in the old version of SAS.                 */
```

```
DATALINES;
Florida    130  90
California  37  26
Texas      1.3 .15
Arizona    .65 .85
;     /* this semi-colon is required! SAS will wait for
         the next line of data until "seen" this semi-colon.*/
RUN;  /* This is optional, but it is always a good idea to
         close this DATA STEP with this RUN; statement.     */

/*  Follow the steps to see where this data SAS data resides:
    go to the Explore window -> Libraries -> Work -> Orange
*/


*OOOOOOOOOOOOOOOOOO;
**     Topic 5;
*OOOOOOOOOOOOOOOOOO;
/*******************************************************
A simplest SAS Procedure syntax:

PROC keyword DATA = datasetname;
 the specific task to perform on a SAS data set;
RUN;

Next, we print out the SAS data set created from the above
DATA STEP
*******************************************************/

PROC PRINT DATA = Orange; /* PRINT <- keyword, DATA <- dataset in the trmporary library */
TITLE1 "This is a toy data set";
TITLE2 "Orange";
RUN;

/* After you run this block of code, you will see the data set
   in a browser (html) and the output window (if it is selected).  */

** Topic 6. SAS data structure: descriptor;
```

```
/* We can find descriptor of as SAS data set by using PROC CONTENTS */

PROC CONTENTS DATA = ORANGE; /* display the contents in the output windows */
RUN;

/* Remarks: The default length for a numeric variable is 8 bytes.
   The length of a character variable is set at the first occurrence of the variable.

   The following data step is called LIST INPUT. More examples and details
   will be given in the next few weeks.
*/

DATA Orange_02;  /* send the SAS data set to the temporary library */
INPUT state $  early   late  ;
DATALINES;
California  37  26
Florida    130  90
Texas      1.3 .15
Arizona    .65 .85
;
RUN;

PROC PRINT DATA = ORANGE_02;
TITLE1 "List input style";
TITLE2 "Orange 02";
RUN;

TITLE "";
PROC CONTENTS DATA = ORANGE_02; /* display the contents in the output windows */
RUN;


/* You can check them out from the Results windows: */

/*OOOOOOOOOOOOOOOOOOOOOOOOO
         Summary
OOOOOOOOOOOOOOOOOOOOOOOOOOO
```

```
1. A good program should have an informative program header.
2. Define a library to store your SAS data sets.
3. Set options to produce nice-looking outputs.
4. Commenting on your program to make it clear and understandable.
5. DATA step creates and/or modifies SAS data sets.
6. PROC step performs specific tasks on SAS data.

Next time, we still use column input to define the data set and add
a few more ingredients in both the data step and procedure step to make
better SAS programs.

*******************************/
```