```
/*
PHS451:  Introduction to SAS;

Session #1
Overview:
1.  Read in a Permanent SAS Data Set
2.    Creation of Temporary and Permanent Data Sets
3.    Contents, Print and Sort Procedures

The lines in capital letters are to be typed exactly as shown into the
Enhanced Editor window of the SAS program.  The sentences in lowercase
letters (most often green) are explanations and instructions.

The Data step: The DATA step begins the process of building a SAS data
set.  A LIBNAME statement is required to create and/or read in a
permanent SAS dataset, one that is stored and that you can access on
future dates. The LIBNAME statement assigns a name and location to a
SAS data library in which the data file will be stored
*/

* First, let's learn how to read in a permanent SAS data set into SAS
;

* You may need to change the directory in the LIBNAME statement before
  you run this code ;

* We will run the following code in the Enhanced Editor ;

LIBNAME S1 'E:\PHS451\Session 1';
DATA ONE;
  SET S1.TOPNAMES2014;
  RUN;

* You have just created your first datastep in SAS.  You created a
  temporary dataset called ONE by reading in a permanent dataset
called
  TOPNAMES2014 ;

* Next, we will look at the various windows in the SAS software ;
* But first, let's run a SAS Procedure.  This will give us a bit of
  output to look at ;

PROC PRINT;
  RUN;

** Look at Enhanced Editor, Log, Explorer, Results, Output
   and Results Viewer ;
** ALWAYS LOOK AT THE LOG!! ;


************************************************************ ;
```

```
* Sample program that creates a dataset using LIST INPUT ;

DATA S1.SESSION1A;
           INPUT STUDYID GENDER $ HEIGHT WEIGHT HAIR $ EYE $;
                DATALINES;
                     01 FEMALE 64 120 BROWN BROWN
                     02 MALE    70 180 BLONDE BLUE
                     03 MALE 72 200 BROWN BLUE
                     04 FEMALE 68 140 BROWN BROWN
                     05 MALE 71 170 BLOND BLUE
                ;
     RUN;


* Let's look at each line in the above program by itself;

* A DATA statement will name the data set being created and tells
  what library (location) to store it in (same as defined in the
  LIBNAME statement);

DATA S1.SESSION1A;

/*In this example we are going to assume that we have some data we
want to make into a SAS data set so we can analyze it.  There are
several styles of data input.  The following is an example of list
input.  This is the easiest type of input but it has some
restrictions.
            1. The fields must be separated by at least one blank.
                2. Each field must be specified in order.
                3. The default length of character variables is 8
bytes.
                4. Data must be standard character or numeric format.

We now type the INPUT statement which defines the variables and
indicates how the variables are to be read.  Variables which are
to be entered as characters (words) as opposed to
numeric are followed by a $.*/

INPUT STUDYID GENDER $ HEIGHT WEIGHT HAIR $ EYE $;

/*
The variables STUDYID HEIGHT AND WEIGHT are numeric variables and will
contain numeric data whereas the variables GENDER, HAIR, and EYE are
character variables that will contain character (text) data.


Now the actual data for each variable specified is entered using a
CARDS or DATALINES statement.  The data for each observation/record
(each person/event) is entered on one line followed by an "enter".
After data for the last observation is entered, hit the "enter" key
and type a semicolon.
This signified to SAS that data entry is complete.
```

```
All DATA and PROCEDURE steps in SAS should have a "RUN;" line
of code at the end of the step.
*/

DATALINES;
      01 FEMALE 64 120 BROWN BROWN
      02 MALE    70 180 BLONDE BLUE
      03 MALE 72 200 BROWN BLUE
      04 FEMALE 68 140 BROWN BROWN
      05 MALE 71 170 BLOND BLUE
      ;
RUN;


*SAME PROGRAM WITHOUT COMMENTS;
DATA S1.SESSION1A;
INPUT STUDYID GENDER $  HEIGHT WEIGHT HAIR $ EYE $;
DATALINES;
      01 FEMALE 64 120 BROWN BROWN
      02 MALE    70 180 BLONDE BLUE
      03 MALE 72 200 BROWN BLUE
      04 FEMALE 68 140 BROWN BROWN
      05 MALE 71 170 BLOND BLUE
      ;
RUN;

/*If the Syntax is correct, SAS will execute the DATA set.  Your
permanent SAS data file, called SESSION1A, is created and stored
in the location specified by your LIBNAME statement.
You can now reference this dataset to execute procedures.

Procedures are specified with the word PROC.  It is often helpful
when creating a new dataset to look at the contents of the dataset
and to see a tabular listing of all the variables.

PROC CONTENTS gives you a table of contents.
The contents are listed in alphabetical order by default.
If you would like to see the contents in the order in which they
are entered into the dataset and the alphabetical order, you can
type the option POSITION.  If you only want to see the contents
in the order in which they are entered, you can use the option VARNUM
*/

PROC CONTENTS DATA = S1.SESSION1A;
RUN;

PROC CONTENTS POSITION DATA = S1.SESSION1A;
RUN;

PROC CONTENTS VARNUM DATA = S1.SESSION1A;
RUN;
```

```
/*PROC PRINT will print your data in tabular form.
If you only want to print the data contained in certain variables,
you can specify which ones by using a VAR statement and listing
the data for variables you want to see.  For each procedure, specify
with a DATA option (DATA=) the data set to which you are referring*/

PROC PRINT DATA = S1.SESSION1A;
RUN;

PROC PRINT DATA = S1.SESSION1A ;
     VAR STUDYID HEIGHT WEIGHT;
RUN;

/* Sometimes you will be given raw data which is already in an
electronic file other than a SAS data set.  If the data are arranged
with one space between each variable and has no embedded blank spaces
you can use a simple list input statement.  When you want refer to an
external file of raw data you use an INFILE statement.  You then use
an INPUT statement with the appropriate input style to name the
variables.  No CARDS or DATALINES statement is necessary in the case
of external raw data. Chapter 2 in the Little SAS Book will be very
useful when you receive data in nonstandard formats.

Suppose you have an external data file named RAW_SESSION1B.TXT which
is stored in the S1 library (because you made a copy
from Learn@UW and placed it there).

--- CONTENTS OF RAW_SESSION1B.TXT___
01 case 28 exposed
02 control 29 exposed
03 case 32 unexposed
04 control 26 exposed
05 control 25 unexposed
06 case 30 exposed

You can refer to this file, name the variables, and create a
SAS data set using the INFILE AND INPUT statements */


DATA WORK.SESSION1B;
INFILE 'E:\PH451\Session 1\RAW_SESSION1B.TXT';
INPUT STUDYID CACO  $ AGE EXPOSURE $;
RUN;

PROC PRINT DATA = SESSION1B;
RUN;
```

```
*/ Note that because you did not use a LIBNAME statement and did not
specify a two level name for the file, a temporary SAS data set was
created from the raw data file.  A temporary SAS data set will be
available only during the current SAS session.  Once you quit the
program, the file will be deleted and you will have create it again
to be able to use it.  If you are creating a SAS data set that you
intend to use multiple times, it often makes sense to create
a permanent SAS dataset.

Formatted input provides the ability to read non-standard numeric or
character values.  Dates are a good example of when formatted
input is needed.

Suppose you have an external data file called RAW_SESSION1C
stored in the S1 library (again because you made a copy
from Learn@UW and placed it there).


--- CONTENTS OF RAW_SESSION1C.TXT___
01 case 01/16/1995 28 exposed
02 control 01/25/1992 29 exposed
03 case 10/14/1996 32 unexposed
04 control 08/25/1992 26 exposed
05 control 03/18/1997 25 unexposed
06 case 06/24/1993 30 exposed


Notice that the third item of data in the file is a date.
You can refer to this file, name the variables, and create a
SAS data set using the INFILE and INPUT statements*/;

DATA SESSION1C;
     INFILE ' E:\PH451\Session 1\RAW_SESSION1C.TXT';
     INPUT STUDYID CACO $ BIRTHDATE MMDDYY10. AGE EXPOSURE $;
RUN;

*The MMDDYY10. is called an informat and it tells SAS to read the
input as a date which is in the form of month/day/year and is 10 bytes
long.  Let's print the data set to see what it looks like;

PROC PRINT DATA = SESSION1C;
RUN;

*Notice that the BIRTHDATE field has some numbers that don't look
like dates at all.  This is because of the way that SAS stores dates.
It calculates the number of days that the date you wish to store is
from a set date in its memory and stores that value.  If you want to
view dates in a dataset in the usual way, you must add a FORMAT
statement to your PRINT statement;
```

```
PROC PRINT DATA = SESSION1C;
     FORMAT BIRTHDATE MMDDYY10.;
RUN;


/* Now the dates are in a format that is understandable to you.
The SAS book gives more detail about styles of input.  It would
be particularly useful for you to learn about the various types
of formatted input SAS allows.  For instance you can use an informat
to specify to SAS that a variable in the raw data is a number
with a comma (comma5.) These informats, like the date informat,
tell SAS how to read the data.  Note that informats in an
INPUT statement end with a period (i.e. MMDDYY10.)

We have created a few temporary datasets including SESSION1C
(Look in the Work folder under libraries).  If we leave the program
now the dataset will be deleted.  We can make complete copies of SAS
datasets using the SET command as follows: */


DATA S1.SESSION1C; *Creates new dataset SESSION1C placing it into the
S1 folder;
SET WORK.SESSION1C; *Starting with a copy of the dataset Session1C in
the WORK folder;
RUN;



/* The code above creates a copy of the dataset WORK.SESSION1C and
calls the same name SESSION1C and places it in the location S1
which has been previously referenced using a LIBNAME statement.

When you wish to read in saved SAS datasets into the temporary
WORK folder you will do this in reverse.  In this case you have
a generic dataset named "dataset" stored in the S1 library
that you want to pull into the temporary workspace giving it
the same name "dataset"*/


DATA WORK.DATASET;  *Creates a new dataset called "dataset" in the
Work folder;
SET S1.DATASET; *Starts with copying dataset "dataset" in the S1
folder;
RUN; *Runs the program;
```

```
/*   Let's create a new dataset called NEW and place it
    into the Work library.
     Let's have 3 variables Firstname, Nickname, AgeYrs
     Let's make the first two variables character
    and the last one numeric
     Create one entry for everyone in your row and print
    the contents to the screen;*/




/*
The DATALINES statement tells SAS that the following lines of code
contain the data for each observation.  Again, note that data for each
observation goes on one line. Be sure to insert a blank space between
each data item.  When finished entering  all of the data for all
observations
hit ENTER and type a semicolon.  Hit ENTER again then type RUN;
This tells SAS that you want it to execute your program.

Note how missing data is entered into the row for each observation.
Missing data is denoted with a period (.) for numeric variables should
someone not report an age.

You should always check your log file.  It should tell you that your
dataset was created, how many observations it contained,
and how many variables are in the dataset

You should also check that the dataset was stored in the location
desired.
Click the tab marked "Explorer" located at the bottom of the third
window
on the left side of the SAS screen.  This activates the window.
The explorer is similar to the Windows Explorer that lets you view
the files on your computer.  The SAS explorer is limited only to areas
of
your computer's disk drives that you tell SAS about using the LIBNAME
statement.  Double-click the file cabinet icon labeled "Libraries".
*/


/*
Session #2 Preview:
1. Use of Import Wizard to Upload Standard File Types
2. Syntax of Sorting Procedure
3. Creating Copies of Datasets (Review)
4. Data Stacking and Merging
*/
```