Conditional Statements
Subsetting
Loops
Start by saving the data sets student.txt and student2.txt from the course web site to the folder from which you usually import data into SAS, in my case U:\data.
## Comparison Operators

<table>
<thead>
<tr>
<th>Logical Comparison</th>
<th>Mnemonic</th>
<th>Symbol</th>
</tr>
</thead>
<tbody>
<tr>
<td>Equal to</td>
<td>EQ</td>
<td>=</td>
</tr>
<tr>
<td>Not equal to</td>
<td>NE</td>
<td>^= or ~=</td>
</tr>
<tr>
<td>Less than</td>
<td>LT</td>
<td>&lt;</td>
</tr>
<tr>
<td>Less than or equal to</td>
<td>LE</td>
<td>&lt;=</td>
</tr>
<tr>
<td>Greater than</td>
<td>GT</td>
<td>&gt;</td>
</tr>
<tr>
<td>Greater than or equal</td>
<td>GE</td>
<td>&gt;=</td>
</tr>
<tr>
<td>Equal to one in a list</td>
<td>IN</td>
<td></td>
</tr>
</tbody>
</table>
Using `student.txt` generate new variable `AgeGroup`.

```sas
data student;
  infile "U:\data\student.txt";
  input Age Major $ GPA;
  if Age le 22 then AgeGroup = 1;
  if Age gt 22 then AgeGroup = 2;
  * try it with <= and > in place of le and gt;
run; * Also try Agegroup instead of AgeGroup;

title "Student Data with Age Group";
proc print data = student noobs;
  * noobs removes the observations column;
run;
```
Here I used the Snipping Tool in Accessories (not on server) to capture the output image and saved it as student.jpg in the appropriate folder for use in \LaTeX{} on my machine. Not crisp.
Be careful with missing values in if/then statements.
SAS treats missing values logically as the most negative number on your computer.
If you use < or <= statements, this will include missing values.

data student2;
  infile "U:\data\student2.txt";
  input Age Major $ GPA;
  if Age le 22 then AgeGroup = 1;
  if Age gt 22 then AgeGroup = 2;
run;

title "Student Data with Age Group and Missing Values";
proc print data = student2 noobs;
run;
Here missing values are classified as $\leq 22$, which is inappropriate.
Correct treatment of missing values

data student2;
infile "U:\data\student2.txt";
input Age Major $ GPA;
case = _N_;  
if Age le 22 and Age ne . then AgeGroup = 1;
if Age gt 22 then AgeGroup = 2;
run;

Missing values are now reflected as . in the AgeGroup column.
Here we printed the output to Adobe PDF, for better graphics.
Showed how to get a Case variable using _N_.
Showed how to control the variable order in proc print.
### Student Data with Age Group and Missing Values

<table>
<thead>
<tr>
<th>case</th>
<th>Age</th>
<th>Major</th>
<th>GPA</th>
<th>AgeGroup</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>24</td>
<td>Math</td>
<td>2.3</td>
<td>2</td>
</tr>
<tr>
<td>2</td>
<td>21</td>
<td>Stat</td>
<td>2.6</td>
<td>1</td>
</tr>
<tr>
<td>3</td>
<td>20</td>
<td>Math</td>
<td>3.8</td>
<td>1</td>
</tr>
<tr>
<td>4</td>
<td>27</td>
<td>Bio</td>
<td>3.8</td>
<td>1</td>
</tr>
<tr>
<td>5</td>
<td>18</td>
<td>Bio</td>
<td>3.9</td>
<td>1</td>
</tr>
<tr>
<td>6</td>
<td>18</td>
<td>Nursing</td>
<td>2.4</td>
<td>1</td>
</tr>
<tr>
<td>7</td>
<td>19</td>
<td>Stat</td>
<td>2.6</td>
<td>1</td>
</tr>
<tr>
<td>8</td>
<td>35</td>
<td>Nursing</td>
<td>3.8</td>
<td>.</td>
</tr>
<tr>
<td>9</td>
<td>25</td>
<td>Bio</td>
<td>3.8</td>
<td>2</td>
</tr>
<tr>
<td>10</td>
<td>18</td>
<td>Math</td>
<td>3.9</td>
<td>1</td>
</tr>
<tr>
<td>11</td>
<td>19</td>
<td>Bio</td>
<td>2.4</td>
<td>1</td>
</tr>
<tr>
<td>12</td>
<td>27</td>
<td>Nursing</td>
<td>2.5</td>
<td>2</td>
</tr>
<tr>
<td>13</td>
<td>20</td>
<td>Nursing</td>
<td>2.7</td>
<td>1</td>
</tr>
<tr>
<td>14</td>
<td>20</td>
<td>Stat</td>
<td>2.4</td>
<td>1</td>
</tr>
<tr>
<td>15</td>
<td>22</td>
<td>Math</td>
<td>2.8</td>
<td>1</td>
</tr>
<tr>
<td>16</td>
<td>25</td>
<td>Stat</td>
<td>3.3</td>
<td>1</td>
</tr>
<tr>
<td>17</td>
<td>23</td>
<td>Nursing</td>
<td>3.8</td>
<td>2</td>
</tr>
<tr>
<td>18</td>
<td>19</td>
<td>Bio</td>
<td>3.9</td>
<td>1</td>
</tr>
<tr>
<td>19</td>
<td>20</td>
<td>Bio</td>
<td>3.4</td>
<td>1</td>
</tr>
</tbody>
</table>
data stat_student;
    infile "U:\data\student.txt";
    input Age Major $ GPA;
    if Major eq "Stat";
* here Stat is case sensitive, STAT and stat will produce an empty set, no output.
    Inside the file student.txt Stat is used;
run;

title "Stat Student Data";
proc print data = stat_student;
run;
# Stat Student Data

<table>
<thead>
<tr>
<th>Obs</th>
<th>Age</th>
<th>Major</th>
<th>GPA</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>21</td>
<td>Stat</td>
<td>2.6</td>
</tr>
<tr>
<td>2</td>
<td>19</td>
<td>Stat</td>
<td>2.6</td>
</tr>
<tr>
<td>3</td>
<td>20</td>
<td>Stat</td>
<td>2.4</td>
</tr>
<tr>
<td>4</td>
<td>23</td>
<td>Stat</td>
<td>3.3</td>
</tr>
</tbody>
</table>
The IN operator can check for multiple conditions, can be used in place of multiple OR statements

if Quiz="A+" or Quiz="A" or Quiz="B+"
then QuizRange=1;

We can use the IN operator instead as follows

if Quiz in ("A+" "A" "B+)
then QuizRange=1;

The list values in (...) can be separated by spaces or commas.

You can also use IN with numeric values

where Subject in (10,22:25,30);
We can use both WHERE and IF to subset data.

There are more operators that can be used with WHERE.

WHERE can be used inside SAS procs to subset data.

The IF cannot be used inside procs.
data student;
   infile "U:\data\student.txt";
   input Age Major $ GPA;
   case = _N_; * _N_ is a system variable;
run;

title "Student Data";
proc print data = student;
   where case in (1:4 9:10 16);
run;
Student Data

<table>
<thead>
<tr>
<th>Obs</th>
<th>Age</th>
<th>Major</th>
<th>GPA</th>
<th>case</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>24</td>
<td>Math</td>
<td>2.3</td>
<td>1</td>
</tr>
<tr>
<td>2</td>
<td>21</td>
<td>Stat</td>
<td>2.6</td>
<td>2</td>
</tr>
<tr>
<td>3</td>
<td>20</td>
<td>Math</td>
<td>3.8</td>
<td>3</td>
</tr>
<tr>
<td>4</td>
<td>19</td>
<td>Bio</td>
<td>3.8</td>
<td>4</td>
</tr>
<tr>
<td>9</td>
<td>25</td>
<td>Bio</td>
<td>3.8</td>
<td>9</td>
</tr>
<tr>
<td>10</td>
<td>18</td>
<td>Math</td>
<td>3.9</td>
<td>10</td>
</tr>
<tr>
<td>16</td>
<td>23</td>
<td>Stat</td>
<td>3.3</td>
<td>16</td>
</tr>
</tbody>
</table>
data student;
  infile "U:\data\student.txt";
  input Age Major $ GPA;
  case = _N_;  
  if case in (1:4 9:10 16);
  * could use WHERE in place of IF;
run;

title "Student Data";
proc print data = student;
run;
The Output: Subsetting Data Set by Case

Student Data

<table>
<thead>
<tr>
<th>Obs</th>
<th>Age</th>
<th>Major</th>
<th>GPA</th>
<th>case</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>24</td>
<td>Math</td>
<td>2.3</td>
<td>1</td>
</tr>
<tr>
<td>2</td>
<td>21</td>
<td>Stat</td>
<td>2.6</td>
<td>2</td>
</tr>
<tr>
<td>3</td>
<td>20</td>
<td>Math</td>
<td>3.8</td>
<td>3</td>
</tr>
<tr>
<td>4</td>
<td>19</td>
<td>Bio</td>
<td>3.8</td>
<td>4</td>
</tr>
<tr>
<td>5</td>
<td>25</td>
<td>Bio</td>
<td>3.8</td>
<td>9</td>
</tr>
<tr>
<td>6</td>
<td>18</td>
<td>Math</td>
<td>3.9</td>
<td>10</td>
</tr>
<tr>
<td>7</td>
<td>23</td>
<td>Stat</td>
<td>3.3</td>
<td>16</td>
</tr>
</tbody>
</table>
Using the WHERE operator

<table>
<thead>
<tr>
<th>Operator</th>
<th>Description</th>
<th>Example</th>
</tr>
</thead>
<tbody>
<tr>
<td>IS MISSING</td>
<td>is missing the stated value</td>
<td>where Age is missing</td>
</tr>
<tr>
<td>IS NULL</td>
<td>Equivalent to IS MISSING</td>
<td>where Age is null</td>
</tr>
<tr>
<td>BETWEEN_AND_</td>
<td>An inclusive range</td>
<td>where age is between 20 and 25</td>
</tr>
<tr>
<td>CONTAINS</td>
<td>Matches a substring</td>
<td>where name contains Mac</td>
</tr>
<tr>
<td>LIKE</td>
<td>Matching with wildcards</td>
<td>where name like R_n%</td>
</tr>
</tbody>
</table>

- The LIKE expression contains 2 wildcard operators.
- The underscore _ is a place holder for any character (use as many as you like)
- The % matches nothing or a string of any length.
- In the above example R_n% matches Ron, Ronald, Running, Run, etc.
libname mydata "U:\data";
data nursing_student;
set mydata.student;
where Major eq "Nursing";
run;
title "Nursing Student Data";
proc print data = nursing_student;
run;

- This assumes the presence of a permanent SAS data set student.sas7bdat in location U:\data.
- This code creates a new, temporary SAS data set, nursing_student, consisting of just the nursing students.
- Using mydata.nursing_student in place of nursing_student throughout creates a permanent SAS data set nursing_student.sas7bdat in U:\data.
Nursing Student Data

<table>
<thead>
<tr>
<th>Obs</th>
<th>ID</th>
<th>Major</th>
<th>Grade</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>18</td>
<td>Nursing</td>
<td>2.4</td>
</tr>
<tr>
<td>2</td>
<td>23</td>
<td>Nursing</td>
<td>3.8</td>
</tr>
<tr>
<td>3</td>
<td>27</td>
<td>Nursing</td>
<td>2.5</td>
</tr>
<tr>
<td>4</td>
<td>20</td>
<td>Nursing</td>
<td>2.7</td>
</tr>
<tr>
<td>5</td>
<td>23</td>
<td>Nursing</td>
<td>3.8</td>
</tr>
</tbody>
</table>
libname mydata "U:\data";
data nursing_student;
set mydata.student (keep=Major Grade);
where Major eq "Nursing";
run;
title "Nursing Student Data, Major and Grade only";
proc print data = nursing_student;
run;

- This assumes the presence of a permanent SAS data set student in location U:\data.
- This code creates a new, temporary SAS data set, nursing_student, consisting of just the nursing students.
<table>
<thead>
<tr>
<th>Obs</th>
<th>Major</th>
<th>Grade</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Nursing</td>
<td>2.4</td>
</tr>
<tr>
<td>2</td>
<td>Nursing</td>
<td>3.8</td>
</tr>
<tr>
<td>3</td>
<td>Nursing</td>
<td>2.5</td>
</tr>
<tr>
<td>4</td>
<td>Nursing</td>
<td>2.7</td>
</tr>
<tr>
<td>5</td>
<td>Nursing</td>
<td>3.8</td>
</tr>
</tbody>
</table>
libname learn "U:\learn";

data grades; * This will be in WORK;
set learn.grades; * This comes from learn;
if missing(Age) then delete;
if Age le 39 then AgeGrp = "Younger Group";
if Age le 39 then Grade = .4*Midterm+.6*FinalExam;
if Age gt 39 then AgeGrp = "Older Group";
if Age gt 39 then Grade = (Midterm+FinalExam)/2;
run;

title "Listing of Grades";
proc print data=grades;
run;
libname learn "U:\learn";

data grades; * This will be in WORK;
set learn.grades; * This comes from learn;
if missing(Age) then delete;
if Age le 39 then do;
   AgeGrp = "Younger Group";
   Grade = .4*Midterm+.6*FinalExam;
end;
if Age gt 39 then do;
   AgeGrp = "Older Group";
   Grade = (Midterm+FinalExam)/2;
end;
run;
title "Listing of Grades Using Do Group";
proc print data=grades;
run;
### Listing of Grades Using Do Group

<table>
<thead>
<tr>
<th>Obs</th>
<th>Age</th>
<th>Gender</th>
<th>Midterm</th>
<th>Quiz</th>
<th>FinalExam</th>
<th>AgeGrp</th>
<th>Grade</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>21</td>
<td>M</td>
<td>80</td>
<td>B-</td>
<td>82</td>
<td>Younger Group</td>
<td>81.2</td>
</tr>
<tr>
<td>2</td>
<td>35</td>
<td>M</td>
<td>87</td>
<td>B+</td>
<td>85</td>
<td>Younger Group</td>
<td>85.8</td>
</tr>
<tr>
<td>3</td>
<td>48</td>
<td>F</td>
<td>.</td>
<td>.</td>
<td>76</td>
<td>Older Group</td>
<td>.</td>
</tr>
<tr>
<td>4</td>
<td>59</td>
<td>F</td>
<td>95</td>
<td>A+</td>
<td>97</td>
<td>Older Group</td>
<td>96.0</td>
</tr>
<tr>
<td>5</td>
<td>15</td>
<td>M</td>
<td>88</td>
<td>.</td>
<td>93</td>
<td>Younger Group</td>
<td>91.0</td>
</tr>
<tr>
<td>6</td>
<td>67</td>
<td>F</td>
<td>97</td>
<td>A</td>
<td>91</td>
<td>Older Group</td>
<td>94.0</td>
</tr>
<tr>
<td>7</td>
<td>35</td>
<td>F</td>
<td>77</td>
<td>C-</td>
<td>77</td>
<td>Younger Group</td>
<td>77.0</td>
</tr>
<tr>
<td>8</td>
<td>49</td>
<td>M</td>
<td>59</td>
<td>C</td>
<td>81</td>
<td>Older Group</td>
<td>70.0</td>
</tr>
</tbody>
</table>
if(condition1){
    ...
    ...
}
if(condition2){
    ...
    ...
}

- condition1 and condition2 are two logic variables, with values TRUE or FALSE.
The Sum Statement (First Attempt)

data revenue;
input Day : $3.
    Revenue : dollar6. ;
    Total = Total + Revenue; * this won't work;
format Revenue Total dollar8. ;
datalines;
Mon $1,000
Tue $1,500
Wed .
Thu $2,000
Fri $3,000
; run;
title "Listing of Revenue";
proc print data=revenue; run;

○ This does not work since Total is initialized as missing value
   and Total = Total + Revenue ⇒ Total = .
<table>
<thead>
<tr>
<th>Obs</th>
<th>Day</th>
<th>Revenue</th>
<th>Total</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Mon</td>
<td>$1,000</td>
<td>.</td>
</tr>
<tr>
<td>2</td>
<td>Tue</td>
<td>$1,500</td>
<td>.</td>
</tr>
<tr>
<td>3</td>
<td>Wed</td>
<td>.</td>
<td>.</td>
</tr>
<tr>
<td>4</td>
<td>Thu</td>
<td>$2,000</td>
<td>.</td>
</tr>
<tr>
<td>5</td>
<td>Fri</td>
<td>$3,000</td>
<td>.</td>
</tr>
</tbody>
</table>
data revenue;
retain Total 0; * Initializes Total;
input Day : $3.
    Revenue : dollar6.;
    Total = Total + Revenue;
    * this won't work either;
format Revenue Total dollar8. ;
datalines;
Mon $1,000
Tue $1,500
Wed .
Thu $2,000
Fri $3,000
; run;
title "Listing of Revenue";
proc print data=revenue; run;
<table>
<thead>
<tr>
<th>Obs</th>
<th>Total</th>
<th>Day</th>
<th>Revenue</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>$1,000</td>
<td>Mon</td>
<td>$1,000</td>
</tr>
<tr>
<td>2</td>
<td>$2,500</td>
<td>Tue</td>
<td>$1,500</td>
</tr>
<tr>
<td>3</td>
<td>.</td>
<td>Wed</td>
<td>.</td>
</tr>
<tr>
<td>4</td>
<td>.</td>
<td>Thu</td>
<td>$2,000</td>
</tr>
<tr>
<td>5</td>
<td>.</td>
<td>Fri</td>
<td>$3,000</td>
</tr>
</tbody>
</table>
data revenue;
retain Total 0;
input Day : $3.
    Revenue : dollar6.;
    if not missing(Revenue) then
        Total = Total + Revenue;
format Revenue Total dollar8.2 ;
datalines;
Mon $1,000
Tue $1,500
Wed .
Thu $2,000
Fri $3,000
; run;
title "Listing of Revenue";
proc print data=revenue;
var Day Revenue Total; run;
<table>
<thead>
<tr>
<th>Obs</th>
<th>Day</th>
<th>Revenue</th>
<th>Total</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Mon</td>
<td>$1000.00</td>
<td>$1000.00</td>
</tr>
<tr>
<td>2</td>
<td>Tue</td>
<td>$1500.00</td>
<td>$2500.00</td>
</tr>
<tr>
<td>3</td>
<td>Wed</td>
<td></td>
<td>$2500.00</td>
</tr>
<tr>
<td>4</td>
<td>Thu</td>
<td>$2000.00</td>
<td>$4500.00</td>
</tr>
<tr>
<td>5</td>
<td>Fri</td>
<td>$3000.00</td>
<td>$7500.00</td>
</tr>
</tbody>
</table>
data revenue;
input Day : $3.
    Revenue : dollar6.;
    Total + Revenue;
format Revenue Total dollar8.2 ;
datalines;
Mon $1,000
Tue $1,500
Wed .
Thu $2,000
Fri $3,000
; run;
title "Listing of Revenue";
proc print data=revenue;

• Same output as on previous slide.
data compound;
Interest = .0125;
Total = 100;
Year+1; * A SUM statement;
Total + Interest*Total; * ditto;
output; * writes observation to the output;
Year+1;
Total + Interest*Total;
output; * same here;
format Total dollar10.2;
run;

title "Listing of Compound";
proc print data=compound noobs;
run;

- The compounding statements can be repeated, or use a loop.
<table>
<thead>
<tr>
<th>Interest</th>
<th>Total</th>
<th>Year</th>
</tr>
</thead>
<tbody>
<tr>
<td>0.0125</td>
<td>$101.25</td>
<td>1</td>
</tr>
<tr>
<td>0.0125</td>
<td>$102.52</td>
<td>2</td>
</tr>
</tbody>
</table>
data compound;
    Interest = .0125;
    Total = 100;
    do Year = 1 to 5;
        Total + Total * Interest;
        output;
    end;
    format Total dollar10.2;
run;

title "Listing of Compound";
proc print data=compound noobs;
run;

• A lot more compact.
<table>
<thead>
<tr>
<th>Interest</th>
<th>Total</th>
<th>Year</th>
</tr>
</thead>
<tbody>
<tr>
<td>0.0125</td>
<td>$101.25</td>
<td>1</td>
</tr>
<tr>
<td>0.0125</td>
<td>$102.52</td>
<td>2</td>
</tr>
<tr>
<td>0.0125</td>
<td>$103.80</td>
<td>3</td>
</tr>
<tr>
<td>0.0125</td>
<td>$105.09</td>
<td>4</td>
</tr>
<tr>
<td>0.0125</td>
<td>$106.41</td>
<td>5</td>
</tr>
</tbody>
</table>
data equation;
  do X = -10 to 10 by 1;
    Y = 2*x**3-5*x**2+15*x-8;
  output;
  end;
run;
symbol value=dot interpol=sm;
title "Plot of Y against X";
proc gplot data = equation;
  plot Y * X;
run;
Plot of Y against X
do x = 1, 2, 5, 10;

(values of x are: 1, 2, 5, 10)

do month = 'Jan', 'Feb', 'Mar';

(values of month are: 'Jan', 'Feb', 'Mar')

do n = 1, 3, 5 to 9 by 2, 100 to 200 by 50

(values of n are: 1, 3, 5, 7, 9, 100, 150, 200)
data treat;
    do Group = 'Placebo', 'Active';
        do Subj = 1 to 5;
            input Score @;
            * @ ==> keep reading from same line until done;
            output;
        end;
    end;
end;
datalines;
250 222 230 210
199 166 183 123 129 234
;
run;
title "Score by Treatment";
    proc print data = treat;
run;
Score by Treatment

<table>
<thead>
<tr>
<th>Obs</th>
<th>Group</th>
<th>Subj</th>
<th>Score</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Placebo</td>
<td>1</td>
<td>250</td>
</tr>
<tr>
<td>2</td>
<td>Placebo</td>
<td>2</td>
<td>222</td>
</tr>
<tr>
<td>3</td>
<td>Placebo</td>
<td>3</td>
<td>230</td>
</tr>
<tr>
<td>4</td>
<td>Placebo</td>
<td>4</td>
<td>210</td>
</tr>
<tr>
<td>5</td>
<td>Placebo</td>
<td>5</td>
<td>199</td>
</tr>
<tr>
<td>6</td>
<td>Active</td>
<td>1</td>
<td>166</td>
</tr>
<tr>
<td>7</td>
<td>Active</td>
<td>2</td>
<td>183</td>
</tr>
<tr>
<td>8</td>
<td>Active</td>
<td>3</td>
<td>123</td>
</tr>
<tr>
<td>9</td>
<td>Active</td>
<td>4</td>
<td>129</td>
</tr>
<tr>
<td>10</td>
<td>Active</td>
<td>5</td>
<td>234</td>
</tr>
</tbody>
</table>
data double;
    Interest = .0225;
    Total = 100;
    do until (Total ge 200);
        year+1;
        Total = Total+Interest*Total;
        output;
    end;
    format Total Dollar10.2;
run;

title "Doubling Capital";
proc print data = double;
where Total ge 180;
run;

- **do until** always executes at least once.
Doubling Capital

<table>
<thead>
<tr>
<th>Obs</th>
<th>Interest</th>
<th>Total</th>
<th>year</th>
</tr>
</thead>
<tbody>
<tr>
<td>27</td>
<td>0.0225</td>
<td>$182.35</td>
<td>27</td>
</tr>
<tr>
<td>28</td>
<td>0.0225</td>
<td>$186.45</td>
<td>28</td>
</tr>
<tr>
<td>29</td>
<td>0.0225</td>
<td>$190.65</td>
<td>29</td>
</tr>
<tr>
<td>30</td>
<td>0.0225</td>
<td>$194.94</td>
<td>30</td>
</tr>
<tr>
<td>31</td>
<td>0.0225</td>
<td>$199.33</td>
<td>31</td>
</tr>
<tr>
<td>32</td>
<td>0.0225</td>
<td>$203.81</td>
<td>32</td>
</tr>
</tbody>
</table>
data double;
    Interest = .0225;
    Total = 100;
    do while (Total le 200);
        year+1;
        Total = Total+Interest*Total;
    output;
    end;
    format Total Dollar10.2;
run;
title "Doubling Capital";
proc print data = double;
where Total ge 180;
run;

- **while** condition is tested at the start of loop (same result).
- May not execute at all.
Avoid Infinite Loops

- You can stop a SAS program by clicking the exclamation point on the task bar.
- Often the program stops itself because some number overflows.
- Combine do until with upper limit do loop.

```sas
data double;
    Interest = .0225;
    Total = 100;
    do Year = 1 to 100 until (Total gt 200);
        Total = Total+Interest*Total;
        output;
    end;
    format Total Dollar10.2; run;
    title "Doubling Capital";
    proc print data = double;
    where Total ge 180; run;
```
Break out of a loop by the **LEAVE** or **CONTINUE** statements.

data double;
   Interest = .0225;
   Total = 100;
   do Year = 1 to 100;
      Total = Total + Interest * Total;
      output;
      if Total ge 200 then leave;
   end;
   format Total Dollar10.2;
run;
title "Doubling Capital";
proc print data = double;
where Total ge 180;
run;
data double;
    Interest = .0225;
    Total = 100;
    do Year = 1 to 100 until (Total ge 200);
        Total = Total+Interest*Total;
        if Total le 190 then continue;
        output;
    end;
    format Total Dollar10.2;
run;

title "Doubling Capital";
proc print data = double;
where Total ge 180;
run;
### Doubling Capital

<table>
<thead>
<tr>
<th>Obs</th>
<th>Interest</th>
<th>Total</th>
<th>Year</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>0.0225</td>
<td>$190.65</td>
<td>29</td>
</tr>
<tr>
<td>2</td>
<td>0.0225</td>
<td>$194.94</td>
<td>30</td>
</tr>
<tr>
<td>3</td>
<td>0.0225</td>
<td>$199.33</td>
<td>31</td>
</tr>
<tr>
<td>4</td>
<td>0.0225</td>
<td>$203.81</td>
<td>32</td>
</tr>
</tbody>
</table>