LESSON 13: Program control

FOCUS QUESTION: How can I execute different code depending on the data?

This lesson demonstrates how to execute different code depending on the situation.

In this lesson you will:

- Learn how program control is useful for solving problems.
- Use `if-else` to express alternative paths in code.
- Use a `for` loop to repeat a block of code with different variables.

Contents

DATA FOR THIS LESSON

File	Description
%	▪ The data set contains contains sleep diary data for a cohort in MATLAB variables.

File	Description

`diaries.mat` (found on Learn)	▪ The arrays have a column for each person.
	▪ The vectors have an element for each person.
	▪ The values in column *n* correspond to the same person as the value in position n of each vector.
	▪ The file contains the following variables:
	  ▪ `bedTimes` - array of bed times in decimal-date format.
	  ▪ `dayCaffeine` - array of daytime caffeine indicators.
	  ▪ `gender` - vector of male/female gender designators.
	  ▪ `nightCaffeine` - array of evening caffeine indicators.
	  ▪ `section` - vector of section indicators. The possible section numbers are 0, 1, 2, and 3. Section 0 contains only a single instructor. The remaining values correspond to course section numbers.
	  ▪ `toSleepMinutes` - an array of number of minutes to fall asleep.
	  ▪ `useAlarm` - array of alarm use indicators.
	  ▪ `wakeTimes` - array of wakeup times in decimal-date format.
	▪ The data was originally gathered by students taking CS 1173 in the fall 2009 semester and anonymized and randomized to be unidentifiable.
	▪ The first column of each array represents the instructor's values, the rest of the columns represent individual students.
	▪ Diaries were recorded for 21 days (from September 23, 2009 to October 13, 2009).

SETUP FOR LESSON 13

- Create a `ProgramControl` directory on your `V:` drive and make it your current directory.
- Download the `diaries.mat` data file from Blackboard and save it to your `ProgramControl` directory.
- Create a `ProgramControlLesson.m` script file in your `ProgramControl` directory. Enter each of the examples in a new cell in this script.

EXAMPLE 1: Simulate tossing a coin (selection using `if-else`)

Create a new cell in which you type and execute:

```
toss = rand(1, 1);            % Pick a value at random between 0 and 1

if toss <= 0.5                % Test against the value 0.5

    fprintf('Tossed heads\n'); % Say its heads if toss is less 0.5
```

```
    else

        fprintf('Tossed tails\n'); % Say its tails if toss is less 0.5

    end
```

You should see a one variable in your Workspace Browser:

- `toss` - a value between 0 and 1 picked at "random" by the `rand` function

You should also see a message of the following form in the Command Window. If you execute this cell multiple times you will get different messages.

```
Tossed heads
```

EXERCISE 1: Compare two random numbers
Write code to pick two random numbers. Output a message only if the first value is greater than the second value.

EXAMPLE 2: Output the square roots of first 3 integers (simple `for` loop)

Create a new cell in which you type and execute:

```
    for k = 1:3              % The loop index k takes values 1, 2, 3

      fprintf('sqrt(%g) = %g\n', k, sqrt(k));

    end;
```

You should see the following variable in the Workspace Browser:

- k - the "loop counter"

You should also see the following output in the Command Window:

```
sqrt(1) = 1

sqrt(2) = 1.41421

sqrt(3) = 1.73205
```

EXERCISE 2: Output the squares of the integers from 1 to 20 on separate lines.

EXAMPLE 3: Sum the square roots of the first 10 integers (accumulation using a `for` loop)

Create a new cell in which you type and execute:

```
sumSqrts = 0;               % Need a variable to accumulate sum

for k = 1:10                % Loop over the values k = 1, 2, ... 10

  sumSqrts = sumSqrts + sqrt(k);  % Add the next sqrt root to total

end;

fprintf('Sum of square roots from 1 to %g is %g\n', k, sumSqrts);
```

You should see the following 2 variables in your Workspace Browser:

- k - acts as the loop counter
- `sumSqrts` - variable holding the total of the square roots of the numbers from 1 to 10

You should also see the following output in the Command Window:

```
Sum of square roots from 1 to 10 is 22.4683
```

EXERCISE 3: Output the sum of the squares of the integers from 1 to 10.

EXAMPLE 4: Simulate tossing coin 50 times (`for` loop with selection and accumulation)

Create a new cell in which you type and execute:

```
numTosses = 50;           % Number of times to toss the coin

numHeads = 0;             % Need a variable to accumulate total heads

for k = 1:numTosses       % Loop over the values k = 1, 2, ... numTosses

   if rand(1, 1) <= 0.5   % Add to head count if 'tossed a head'

       numHeads = numHeads + 1;

    end;

end;
```

```
    fprintf('%g heads in %g tosses\n', numHeads, numTosses);
```

You should see the following 3 variables in your Workspace Browser:

- k - acts as the loop counter
- `numHeads` - the number of heads that result from the tosses
- `numTosses` - variable holding number of tosses to simulate

You should also see output of the following form in the Command Window:

```
18 heads in 50 tosses
```

EXAMPLE 5: Alternative implementation of coin toss simulation (vector indexing)

Create a new cell in which you type and execute:

```
    timesToTosses = 50;                 % Number of times to toss the coin

    randTosses = rand(timesToTosses,1);  % Create vector of "tosses"

    numHeads = sum(randTosses <= 0.5);   % How many were heads?

    fprintf('%g heads in %g tosses\n', numHeads, numTosses);
```

You should see the following 3 variables in your Workspace Browser:

- `numHeads` - number of values that are less than or equal to 0.5
- `randTosses` - vector of random numbers simulating the tosses
- `timesToTosses` - variable holding the number of times to "toss"

You should also see output of the following form in the Command Window:

```
23 heads in 50 tosses
```

EXAMPLE 6: Load the sleep diary data

Create a new cell in which you type and execute:

```
    load diaries.mat;                       % Load the sleep diaries
```

You should see the following 8 variables in the Workspace Browser:

- `bedTimes` - an array with the bedtimes of individual students in the columns
- `dayCaffeine` - a logical array with columns indicating daytime caffeine use for individual students
- `gender` - a vector of strings containing 'male' or 'female' designations for each student
- `nightCaffeine` - a logical array with columns indicating caffeine use after 6 pm for individual students
- `section` - vector containing sections numbers of the individual studens
- `toSleepMinutes` - an array with the number of minutes to fall asleep each night for the individual students
- `useAlarm` - a logical array with indications of alarm use for individual students in the columns.
- `wakeTimes` - an array with the wake times of individual students in the columns.

EXAMPLE 7: Output a message if any subjects awoke after 3:30 pm

Create a new cell in which you type and execute:

```
wakeHours = (wakeTimes - floor(wakeTimes))*24; % Calculate the wake-up hours

lateWakeup = sum(sum(wakeHours > 15.5));       % How many late wake-ups?

if  lateWakeup > 0                             % See if any late wake-ups

    fprintf('%g wake-ups after 3:30 pm\n', lateWakeup);

end;
```

You should see the following 2 variables in your Workspace Browser:

- `lateWakeup` - number of wake-up times that were after 3:30 pm
- `wakeHours` - array with the wake-up time of day for the diary data set

You should also see the following output in the Command Window.

```
30 wake-ups after 3:30 pm
```

EXERCISE 4: Wake-ups before 5 am

Write MATLAB code to print the number of wake-ups before 5 am.

EXAMPLE 8: Output subject number and gender for subjects with at least 1 wake-up after 3:30 pm

Create a new cell in which you type and execute:

```
    timesLate = sum(wakeHours > 15.5);  % Times each subject woke up late

    fprintf('Subjects who had a least one wake-up after 3:30 pm:\n');

    for k = 1:length(timesLate)

        if timesLate(k) > 0

          fprintf('Subject %g: a %s with %g late wake-ups\n', ...

            k, gender{k}, timesLate(k));

        end;

    end;
```

You should see the following two variables in your Workspace Browser:

- k - acts as the loop counter
- `timesLate` - number of times each subject awoke after 3:30 pm

You should also see the following output in the Command Window.

```
Subjects who had a least one wake-up after 3:30 pm:

Subject 2: a female with 1 late wake-ups

Subject 7: a male with 1 late wake-ups

Subject 8: a female with 1 late wake-ups

Subject 40: a female with 1 late wake-ups

Subject 46: a male with 3 late wake-ups

Subject 66: a male with 1 late wake-ups

Subject 70: a female with 1 late wake-ups

Subject 71: a female with 1 late wake-ups

Subject 73: a female with 1 late wake-ups
```

```
Subject 86: a female with 5 late wake-ups

Subject 101: a male with 4 late wake-ups

Subject 118: a female with 3 late wake-ups

Subject 125: a male with 2 late wake-ups

Subject 134: a female with 3 late wake-ups

Subject 142: a female with 2 late wake-ups
```

EXAMPLE 9: Output the subject number and gender of the first student in section 3 (break)

Create a new cell in which you type and execute:

```
    sect3 = (section == 3);          % True (1) for subjects in section 3

    for k = 1:length(sect3)          % Here k = 1, 2, ... subject number

       if sect3(k)                   % If subject is in section 3

            fprintf('First in section 3 is a %s with subject number %g\n', ...

                gender{k}, k);

            break;                   % Get out of the loop, we done

       end;

    end;
```

You should see the following two variables in your Workspace Browser:

- k - acts as the loop counter
- `sect3` - logical vector with 1's corresponding to students in section

You should also see the following output in the Command Window:

```
First in section 3 is a female with subject number 2
```

EXAMPLE 10: Output a table of early wake-ups using a loop

Create a new cell in which you type and execute:

```
    averWake = mean(wakeHours);        % Compute the average wake up time for all
subjects

    earlyWake = 6;

    fprintf('\n\n\tEarly wake-ups\n');

    fprintf('Subj\tSect\tGender\tAver Wakeup\n');    % Print out a title

    for k = 1:length(averWake)           % Here k = 1, 2, ... subject number

       if averWake(k) >= earlyWake       % Skip subjects who awoke later

           continue;

       end;

       fprintf(' %g\t  %g\t%s\t   %5.2f\n', k, section(k), gender{k}, averWake(k));

    end;
```

You should see the following 2 variables in your Workspace Browser:

- `earlyWake`- threshold for an "early" wake-up
- k - acts as the loop counter

You should also see the following output in the Command Window:

```
         Early wake-ups

Subj     Sect     Gender   Aver Wakeup

 32        1      male         5.87

 91        2      female       5.50

 140       3      female       5.68
```

SUMMARY OF SYNTAX

MATLAB syntax	Description
break	Exits the innermost enclosing loop.
continue	Goes to the next iteration of the innermost enclosing loop, skipping the remaining statements in this iteration.
The `for` loop: **for k = initval:endval** **statements to execute each time** **end;**	Execute `statements` for each value of the loop variable `k` from `initval` to `endval`. Note: the loop variable `k` takes on a different value each time through the loop. You should not modify the loop variable inside the loop.
One alternative selection: **if expression** **statements** **end;**	Execute the `statements` only if the `expression` has the value `true` (non-zero).
Two alternative selection: **if expression** **statements1** **else**	Execute `statements1` when `expression` is `true` (non-zero). Otherwise, execute `statements2`.

MATLAB syntax	Description

`statements2`
 	
`end;`	

`rand(k, j)`	Create an an array with k rows and j columns containing values that are uniformly distributed in (0, 1). The values appear to be statistically "random".
`sqrt(X)`	Returns an array whose elements are the square roots of the corresponding elements of X.

This lesson was written by Kay A. Robbins of the University of Texas at San Antonio and last modified by Dawn Roberson on 1 March 2014. Please contact kay.robbins@utsa.edu with comments or suggestions. The image is a photo of a silver Dekadrachm (Greek) from about 400 B.C. taken by Carl Malamud on 12/14/05 and available at http://commons.wikimedia.org/wiki/File:Ancient_Greek_Silver_Coin_%28Dekadrachm%29,_rev,_about_400_B.C.E..jpg.

Published with MATLAB® 7.14