LESSON 13: Program control questions

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

Contents

- EXAMPLE 1: Simulate tossing a coin (selection using if-else)
- EXAMPLE 2: Output the square roots of first 3 integers (simple for loop)
- EXAMPLE 3: Sum the square roots of the first 10 integers (accumulation using a for loop)
- EXAMPLE 4: Simulate tossing coin 50 times (for loop with selection and accumulation)
- EXAMPLE 5: Alternative implementation of coin toss simulation (vector indexing)
- EXAMPLE 6: Load the sleep diary data
- EXAMPLE 7: Output a message if any subjects awoke after 3:30 pm
- EXAMPLE 8: Output subject number and gender for subjects with at least 1 wake-up after 3:30 pm
- EXAMPLE 9: Output the subject number and gender of the first student in section 3 (break)
- EXAMPLE 10: Output a table of early wake-ups using a loop

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

```matlab
toss = rand(1, 1);
if toss <= 0.5
    fprintf('Tossed heads
');
else
    fprintf('Tossed tails
');
end;
```

<table>
<thead>
<tr>
<th>Questions</th>
<th>Answers</th>
</tr>
</thead>
<tbody>
<tr>
<td>Why does the output change when I execute this cell multiple times?</td>
<td>The if statement gives alternative paths of execution, and only one of the two fprintf statements executes each time. The branch is determined by the &quot;random&quot; value generated by rand.</td>
</tr>
<tr>
<td>Does <code>rand</code> produce truly &quot;random&quot; values?</td>
<td>The <code>rand</code> function generates &quot;pseudorandom&quot; values that are distributed uniformly in (0, 1). The values are not correlated and so they can simulate picking numbers at random or tossing a coin. However, they appear in a well-defined, repeatable sequence so they are not truly random.</td>
</tr>
</tbody>
</table>
### Questions | Answers
--- | ---
The fact that the sequence is repeatable is very useful for repeating experiments. | Does `rand` ever generate a value that is exactly 0 or exactly 1? | No, the output of `rand` is strictly between 0 and 1, exclusive. | What do the parameters of the `rand` function do? | The `rand` function generates an array of random numbers and the parameters specify the size of the array. In this example, `rand` generates a single value (i.e., a 1 x 1 array). The call `rand(2, 3)` generates an array with 2 rows and 3 columns. | Can each branch of the `if` contain more than one statement? | Yes, you can put any number of statements in each section. | What happens after the selected branch of the `if` executes? | MATLAB continues execution with the statement after the `end`. | Tossed tails

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

```matlab
for k = 1:3
    fprintf('sqrt(%g) = %g\n', k, sqrt(k));
end;
```

### Questions | Answers
--- | ---
How many times is this loop executed? | MATLAB executes statements between the `for` and the `end` 3 times. Each time, the loop variable `k` takes on one of the values in the list: 1, 2, 3. | Does a loop body always contain just one statement? | No, you can put as many statements as you want between the `for` and the `end`. | Do I always have to use `k` for the loop variable? | No, you can use any loop variable name that you want. | Does the loop variable have to take on consecutive integral values? | No, you can specify any list of values (e.g., `for x = [1.3, 2.2, 4.2, 5.1]`). | Can I modify the loop variable inside the loop? | Technically you can. However, it is highly recommended that you do not. | What is the value of the loop variable after the loop completes? | Assuming that you following good programming practice and do not modify the loop variable inside the loop, the loop variable will have the last value in the list. For this example, the loop variable `k` will have the value 3.
sqrt(1) = 1
sqrt(2) = 1.41421
sqrt(3) = 1.73205

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

```
sumSqrts = 0;
for k = 1:10
    sumSqrts = sumSqrts + sqrt(k);
end;
fprintf('Sum of square roots is %g\n', sumSqrts);
```

Sum of square roots is 22.4683

**EXAMPLE 4: Simulate tossing coin 50 times (for loop with selection and accumulation)**
numTosses = 50;

numHeads = 0;

for k = 1:numTosses
    if rand(1, 1) <= 0.5
        numHeads = numHeads + 1;
    end;
end;

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

Questions

How many times is this loop executed?
MATLAB executes this loop 50 times, one for each value of k in the list 1, 2, ..., 50.

How many times is the statement numHeads = numHeads + 1 executed?
MATLAB only executes this statement when the result of rand is less than or equal to 0.5. On average, this happens half the time. However, results will vary each time you execute the cell, just as the results of tossing a coin 50 times will vary.

Which end statement closes the loop?
The last end statement closes the for loop. The other end statement finishes the if.

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

timesToTosses = 50;

randTosses = rand(timesToTosses,1);

numHeads = sum(randTosses <= 0.5);

fprintf('%g heads in %g tosses
', numHeads, numTosses);
How big is `randTosses`?
The `randTosses` variable has 50 rows and 1 column.

How big is `randTosses <= 0.5`?
The result is a logical array with 50 rows and 1 column. The entries are 1's when the corresponding entries of `randTosses` are less than or equal to 0.5.

Why does `sum(randTosses <= 0.5)` give the number of heads?
The result counts the number of values in `randTosses` that are less than or equal to 0.5. We have assigned the values of `rand` that are less than or equal to 0.5 to mean heads.

Could I choose heads to be `sum(randTosses > 0.5)` to mean heads?
Yes, the assignment is arbitrary as long as you designate half of the values in (0, 1).

EXAMPLE 6: Load the sleep diary data

```matlab
load diaries.mat;
```

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

```matlab
wakeHours = (wakeTimes - floor(wakeTimes))*24;
lateWakeup = sum(sum(wakeHours > 15.5));
if lateWakeup > 0
    fprintf('%g wake-ups after 3:30 pm\n', lateWakeup);
end;
```

Questions

<table>
<thead>
<tr>
<th><strong>Questions</strong></th>
<th><strong>Answers</strong></th>
</tr>
</thead>
<tbody>
<tr>
<td>How big is <code>randTosses</code>?</td>
<td>The <code>randTosses</code> variable has 50 rows and 1 column.</td>
</tr>
<tr>
<td>How big is <code>randTosses &lt;= 0.5</code>?</td>
<td>The result is a logical array with 50 rows and 1 column. The entries are 1’s when the corresponding entries of <code>randTosses</code> are less than or equal to 0.5.</td>
</tr>
<tr>
<td>Why does <code>sum(randTosses &lt;= 0.5)</code> give the number of heads?</td>
<td>The result counts the number of values in <code>randTosses</code> that are less than or equal to 0.5. We have assigned the values of <code>rand</code> that are less than or equal to 0.5 to mean heads.</td>
</tr>
<tr>
<td>Could I choose heads to be <code>sum(randTosses &gt; 0.5)</code> to mean heads?</td>
<td>Yes, the assignment is arbitrary as long as you designate half of the values in (0, 1).</td>
</tr>
</tbody>
</table>

Questions

<table>
<thead>
<tr>
<th><strong>Questions</strong></th>
<th><strong>Answers</strong></th>
</tr>
</thead>
<tbody>
<tr>
<td>What is the size of <code>wakeHours</code>?</td>
<td>The <code>wakeHours</code> variable has 21 rows and 144 columns.</td>
</tr>
<tr>
<td>What is the size of <code>wakeHours &gt; 15.5</code>?</td>
<td>The variable holds an array that is the same size as <code>wakeHours</code>. It has 1’s where the corresponding entries of <code>wakeHours</code> are greater than 15.5 and 0’s elsewhere.</td>
</tr>
<tr>
<td>Why do I need two <code>sum</code> functions to find the total?</td>
<td>Since <code>wakeHours &gt; 15.5</code> is a 21 x 144 array, <code>sum(wakeHours &gt; 15.5)</code> is a row vector size 1 x 144 containing the column sums. To find the overall total, you need to add up these column sums. Hence, the nested <code>sum</code> functions.</td>
</tr>
</tbody>
</table>

30 wake-ups after 3:30 pm
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:

```matlab
    timesLate = sum(wakeHours > 15.5);

    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;
```

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)

```matlab
sect3 = section == 3;

for k = 1:length(sect3)
    if sect3(k)
        fprintf('First in section 3 is a %s with subject number %g\n', ...
            gender{k}, k);
        break;
    end;
end;
```

<table>
<thead>
<tr>
<th>Questions</th>
<th>Answers</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>What does the break do?</strong></td>
<td>The <em>break</em> provides a controlled way to get out of a loop without going through all of the iterations. The <em>break</em> causes the statement after the innermost enclosing loop to be the next one executed.</td>
</tr>
<tr>
<td><strong>Does break transfer control out of all loops?</strong></td>
<td>No, <em>break</em> only &quot;breaks&quot; out of the innermost enclosing loop.</td>
</tr>
<tr>
<td><strong>Does it matter that the break appears in an if-else?</strong></td>
<td>No, the <em>break</em> is not affected by <em>if-else</em>. However, you will often see <em>break</em> as part of an <em>if-else</em> because you will usually only want to &quot;break&quot; out of the loop under certain circumstances.</td>
</tr>
</tbody>
</table>

First in section 3 is a female with subject number 2
EXAMPLE 10: Output a table of early wake-ups using a loop

```matlab
averWake = mean(wakeHours);
earlyWake = 6;

fprintf('

	Early wake-ups
');
fprintf('Subj	Sect	Gender	Aver Wakeup
');

for k = 1:length(averWake)
    if averWake(k) >= earlyWake
        continue;
    end;

    fprintf(' %g	  %g	%s	  %5.2f
', k, section(k), gender{k}, averWake(k));
end;
```

<table>
<thead>
<tr>
<th>Subj</th>
<th>Sect</th>
<th>Gender</th>
<th>Aver Wakeup</th>
</tr>
</thead>
<tbody>
<tr>
<td>32</td>
<td>1</td>
<td>male</td>
<td>5.87</td>
</tr>
<tr>
<td>91</td>
<td>2</td>
<td>female</td>
<td>5.50</td>
</tr>
<tr>
<td>140</td>
<td>3</td>
<td>female</td>
<td>5.68</td>
</tr>
</tbody>
</table>

This lesson was written by Kay A. Robbins of the University of Texas at San Antonio and last modified on 05-Nov-2012. Please contact krobbins@cs.utsa.edu with comments or suggestions.