Learning Objectives

1. Use more complex ways of setting the loop index
2. Construct nested loops in the following situations:
   a. For use with two dimensional arrays
   b. For repeating a calculation numerous times
3. Use loops to repeat a code with conditional statements

Topics

Students will read Chapter 6.4.1 and 6.4.6 of the MATLAB book before coming to class. This preparation material is provided to supplement this reading.

Students will learn a more advanced understanding of while loops by expanding on the information presented in the Basic For Loops class. This knowledge will be used understand creating and executing nested for loops, as well as additional commands. This material contains the following:

1. Key Definitions
2. Review of for-end loops
3. Nested for loops
4. The ‘break’ Command
5. The ‘continue’ Command

1. Key Definitions

**Loop** – a command, or group of commands, can be executed multiple times, consecutively.

**Relational Operator** – compares two numbers by determining whether a comparison statement is true or false. Operators include: ‘less than <’, ‘greater than >’, ‘less than or equal to <=’, ‘greater than or equal to >=’, ‘equal to ==’, ‘not equal to ~='. 

**Logical Operator** – examines true/false statements and produces a result that is true (1) or false (0) according to the specific operator. Operators include ‘AND’ (&), ‘OR’ (|), ‘NOT’ (~).
Order of Precedence – computers and calculators follow a hierarchy for the order in which operations are executed. Higher-precedence operations are executed before lower-precedence operations; when they are equal, they are executed from left to right.

Conditional Statements – a command that allows a program to make a decision of whether to execute a set of commands or skip those commands.

2. Review of for-end loops

for–end loops are used when the number of passes is known (or can be calculated) in advance. A variable is used to control the looping process. The general structure of a for–end loop is:

```
for k = f : s : t
    ........
    ........
    ........
end
```

- In the first pass, \( k = f \), and MATLAB executes the commands between the ‘for’ and the ‘end’ commands.
- The computer goes back to the ‘for’ command for the second pass. \( k \) obtains a new value equal to \( k = f+s \), and the commands between the ‘for’ and the ‘end’ are executed with the new value of \( k \).
- The process repeats itself until the last pass where \( k = t \).
- **Example:** \( k = 1: 2: 9 \)
  - There are five passes through the loop. The values of \( k \) are: 1 3 5 7 9
3. Nested for-end loops

A **for-end** loop can be nested within another **for-end** loop.

```matlab
for k = 1 : 3
    for n = 1 : 5
        ... commands
    end
end
```

Every time **k** is increased by 1 the nested loop loops five times with the value of **n** ranging from 1 through 5. Overall the commands will be executed 15 times with the values of:
<table>
<thead>
<tr>
<th>k</th>
<th>n</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>1</td>
<td>2</td>
</tr>
<tr>
<td>1</td>
<td>3</td>
</tr>
<tr>
<td>1</td>
<td>4</td>
</tr>
<tr>
<td>1</td>
<td>5</td>
</tr>
<tr>
<td>2</td>
<td>1</td>
</tr>
<tr>
<td>2</td>
<td>2</td>
</tr>
<tr>
<td>2</td>
<td>3</td>
</tr>
<tr>
<td>2</td>
<td>4</td>
</tr>
<tr>
<td>2</td>
<td>5</td>
</tr>
<tr>
<td>3</td>
<td>1</td>
</tr>
<tr>
<td>3</td>
<td>2</td>
</tr>
<tr>
<td>3</td>
<td>3</td>
</tr>
<tr>
<td>3</td>
<td>4</td>
</tr>
<tr>
<td>3</td>
<td>5</td>
</tr>
</tbody>
</table>
for $i = 1: m$
    for $j = 1: n$
        if $i == j$
            matrix($i, j$) = 1;
        else
            matrix($i, j$) = 7;
        end
    end
end
Now that the first pass of \( i \) is complete (because \( j \) has completed the designed \( m=3 \) passes), the loop returns back to the outside loop begins the second pass of \( i \). Within the \( i=2 \) pass, the nested loop will complete 3 passes to meet the required \( j=3 \) passes. These 3 passes produce a row 7 1 7.
Remember that within this nested loop, the program will complete the full cycle of passes for the outside loop, work inwards as required to the inside loop, complete the full cycle of passes for the inside loop, work back out, and then repeat the outside loop as many times as necessary.

```
for i = 1: m
    for j = 1: n
        if i == j
            matrix(i, j) = 1;
        else
            matrix(i, j) = 7;
        end
    end
end
```

The above nested loop, for m = 3 and n = 3, will create a 3x3 matrix:

```
    1   7   7
    7   1   7
    7   7   1
```

If the input was changed to m = 3 and n = 5, then the following 3 x 5 matrix would be created:

```
    1   7   7   7   7
    7   1   7   7   7
    7   7   1   7   7
```
4. The ‘break’ Command
   - When MATLAB encounters a ‘break’ command within a loop, MATLAB jumps to the end command of the loop
   - MATLAB then continues to execute the commands that follow
   - The ‘break’ command is typically used within a conditional ‘if’ statement to terminate the execution of a loop if some condition is satisfied.

Example of nested loop with ‘break’ command

```matlab
a = 0;
for x=1:10
    a = a + 1;
    if (a == 5)
        break
    end
    disp (a)
end
```

![Image of nested loop with 'break' command]

5. The ‘continue’ Command
   - The ‘continue’ command resembles the ‘break’ command but it only stops the remainder of the current iteration, not all iterations
   - MATLAB jumps to the end statement and continues execution with the NEXT iteration of the loop
   - The ‘continue’ command typically occurs within a conditional statement (if – end)

Example of nested loop with ‘continue’ command

```matlab
a = 0;
for x=1:10
    a = a + 1;
    if (a == 5)
        continue
    end
    disp (a)
end
```

![Image of nested loop with 'continue' command]