Topics: iteration using for, random number, vectorized code, intro to user defined function

Reading (ML): Sec 4.2–4.5 (exclude 4.2.2), revisit Sec 2.8.2 (exclude matrix operations), Sec 5.0.

General form of the for Loop

```
for index = expression
    Statements to execute
    Also called loop body
end
```

Expression usually takes the form of a vector. E.g., 1:n.

Two patterns for doing something $n$ times

```
for i = 1:n
    % do something
    % ...
end
```

```
while i<=n
    % do something
    % ...
    i = i + 1;
end
```

Example 1: Average

Write a program that prompts the user for 10 numbers and then print the average. Use a for loop and store all user input in a vector data.

```matlab
n = 10;    % no. of data points
for
    data = input('Enter a number: ');
end
ave = mean(data);
fprintf('Average of the %d numbers is %.2f\n', n, ave);
```

Vectorized Code

MATLAB can perform operations on more than one value (variable) at a time. Program segments that show this feature are said to be vectorized.

E.g., let $a$ and $b$ be vectors of equal length. One can add these two vectors such that $c(i) = a(i) + b(i)$ for all indices $i$. In most programming languages (including Java), one must perform this add operation on each element of the vector individually. In MATLAB, the add operation can be performed on the entire vector at the same time.

<table>
<thead>
<tr>
<th>Code with a loop</th>
<th>Vectorized code</th>
</tr>
</thead>
<tbody>
<tr>
<td>$n = length(a);$</td>
<td></td>
</tr>
<tr>
<td>$c = zeros(1,n);$</td>
<td></td>
</tr>
<tr>
<td>for $i = 1:n$</td>
<td>$c = a + b;$</td>
</tr>
<tr>
<td>$c(i) = a(i) + b(i);$</td>
<td></td>
</tr>
<tr>
<td>end</td>
<td></td>
</tr>
</tbody>
</table>

Mathematical and logical operations that can be used in vectorized code include

```
+ - .* ./ .^ == > < ~= & | ~
```
Example 2: Random walk

Write a program that performs a “random walk.” In a random walk, possible moves are left, right, up, or down (in a Cartesian plane). Prompt the user for the number of steps and the starting point.

```matlab
% Perform n steps of random walk starting from (x0,y0)

disp('Do a random walk!')
n = input('How many steps? ');
x0 = input('From what x-coordinate? ');
y0 = input('From what y-coordinate? ');

% possible movements: ( xdir(i), ydir(i) )
    xdir = [ 1 -1 0 0];
    ydir = [ 0 0 1 -1];

    x = [x0 zeros(1,n)];  % trajectory in x direction
    y = [y0 zeros(1,n)];  % trajectory in y direction

% Perform walk, each step is based on a random integer

    for k =
        % Generate an integer in the range of 1..4

            % Calculate new location

    end

% Show the walk, starting point, ending point
    plot(x,y,x(1),y(1),'*',x(end),y(end),'o')
    title([num2str(n) ' steps of random walk from * to o'])
```

User-Defined Function

- Can easily “reuse” code
- Functions can be independently tested
- Upon invocation, each function has its own memory space inaccessible by other functions or the command window space—variables in a function can be “seen” only inside the function
- Values stored in variables are not preserved between function calls.
- Arguments are “passed by value”