Topics: 1-dimensional array, vectorized code, iteration using `for`

Reading (ML): Sec 4.2–4.4, revisit Sec 2.1-2.3, 2.7, 2.9 for discussion on array

1-Dimensional Array: Vector

An array is a *named* collection of data values organized into rows and/or columns. A 1-d array is a row or a column, also known as a *vector*. An *index* is a positive integer that identifies the position of a value in the vector.

Suppose vector **v** is a collection of 4 values, i.e., vector **v** has 4 cells.

The *i*th value can be accessed as **v(i)**.

Assign a value of 9 to into the 4th cell of vector **v**: **v(4) = 9**.

Copy the value in the 4th cell to the 2nd cell of vector **v**: **v(2) = v(4)**.

Copy the value in the current cell to the next cell of vector **v**: **v(i+1) = v(i)**.

Array Initialization

MATLAB function **zeros**: **vecA = zeros(1,5)**
MATLAB function **ones**: **vecB = ones(5,1)**
MATLAB short-cut expression for consecutive numbers: **1:6** or **1:1:6**
"Manual": **vecC(5) = 10**

Can you write a program for calculating the average of 10 numbers (Example 1 from 9/13 lecture) that stores all the data entered by the user? Below is the original program that doesn't store all user input.

```
% Average 10 numbers from user input




n = 10;     % number of data values
total = 0;  % current sum (initialized to zero)
i = 1;      % initialize counter
while (i<=n)
  % read and process input value
    num   = input('Enter a number: ');
    total = total + num;
  % update
    i = i + 1;
end
ave = total/n  % average of n numbers
```

What are some useful MATLAB built-in functions for the above problem?

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
```

```
i = 1;
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.

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.

Code with a loop	Vectorized code

`n = length(a);`
`c = zeros(1,n);`
`for i = 1:n`
`  c(i) = a(i) + b(i);`
`end`
	`c = a + b;`

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

```
+  -  .*  ./  .^  ==  >  <  ~=  &  |
```

Example 2

Write a program segment that determines whether a given integer **n** is prime. Assume n>2. (Hint: MATLAB function **mod(x,y)** returns the value of the remainder of x divided by y assuming integer values of x, y.)

Example 3

Given an integer **n**, write a program segment to list all the prime numbers in the range of [2,**n**].