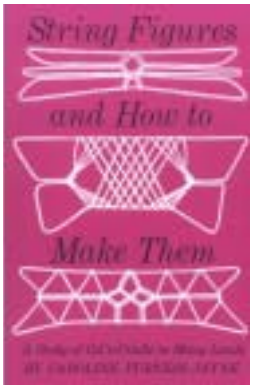



# Characters & Strings

Lecture 10 (Feb 27)
CS100M – Spring 2007

## Announcements

- Prelim One
  - Returned in Section
  - Unclaimed in section or no indication of section
    - Available for pick up at the ACCEL Green Room during consulting hours starting Thursday
- Section
  - In lab this week
  - Online evaluations of TAs start this week (see website)
- Project 3
  - Due Thursday, March 8
  - Posted last Saturday; typo in *integral.m* was corrected Monday

## Topics

- Reading: CFile 5, Section 5.2
- Recall
  - Matlab vectors (1D arrays)
  - Vector indices ("subscripts")
  - Creating vectors
    - [ ], ":" notation, special functions, appending, combining
- Plans for today
  - Characters & strings
  - More examples using Matlab vectors
  - Use of plot( )

## Matlab Strings

- We have already made use of strings
  - n = input('Next number: ');
  - fprintf('The answer is %d.', answer);
- 'Next number: ' and 'The answer is %d.' are both *strings*
- A string is made up of individual characters
  - The string 'CS100M rules' consists of 12 characters (8 letters, 3 digits, and 1 space)
- In Matlab, a string is a *vector* of characters
  - Since a string is a vector, it uses the same indexing scheme as any other vector

## Single Quotes

- Anything enclosed in single quotes is a string (*even if it looks like something else*)
  - '100' is a string (i.e., a character vector) of length 3
  - 100 is a numeric value
  - 'pi' is a string of length 2
  - pi is predefined constant (= 3.14159…)
  - 'x' is a character (also a string of length 1)
  - x is a variable name

## Strings as Vectors

### Vectors

- Indexing

```
v = [ 7 0 5 ];
x = v(3);      % x is 5
v(1) = 1;      % v is [1 0 5]
```

- ":" notation

```
v = 2:5;       % v is [2 3 4 5]
```

- Appending

```
v = [7 0 5];
v(4) = 2;      % v is [7 0 5 2]
```

- Concatenation

```
v = [v [4 6]]
    % v is [7 0 5 2 4 6]
```

### Strings

- Indexing

```
s = 'hello';
c = s(2);      % c is 'e'
s(1) = 'J';    % s is 'Jello'
```

- ":" notation

```
s = 'a' : 'g';   % s is 'abcdefg'
```

- Appending

```
s = 'duck';
s(5) = 's';    % s is 'ducks'
```

- Concatenation

```
s = [s ' quack']
    % s is 'ducks quack'
```

## Some Useful String Functions

```
str = 'CS100M rules';

isletter(str)      % [ 1 1 0 0 0 1 0 1 1 1 1 1 ]
isspace(str)       % [ 0 0 0 0 0 0 0 1 0 0 0 0 0 ]

s = lower(str);    % s is 'cs100m rules'
s = upper(str);    % s is 'CS100M RULES'

ischar(str);       % Is str a char array?  1 (= true)
```

## Example: Capitalize First Letters

- Goal:
  - Write a function to capitalize just the first letter of each word in a string
  - Assume the string consists entirely of letters and spaces
- Function header

```
function result = capitalize(str)
% Post: Convert string so each word has just first letter capitalized
% Pre: Input string consists entirely of letters & spaces
```

Post = What is supposed to have happened when function is done (i.e., what the function does)

Pre = What assumptions are being made when function starts

## ASCII

(American Standard Code for Information Interchange)

| ASCII Code | Character | ASCII Code | Character |
|---|---|---|---|
| 48 | '0' | 97 | 'a' |
| 49 | '1' | 98 | 'b' |
| 50 | '2' | 99 | 'c' |
| 51 | '3' | ... | ... |
| ... | ... | 122 | 'z' |
| 65 | 'A' | ... | ... |
| 66 | 'B' | 127 | DEL |
| 67 | 'C' | | |
| ... | ... | | |
| 90 | 'Z' | | |
| ... | ... | | |

## Characters ↔ ASCII Code

```
str = 'CS100M';          % Vector (1D array) of characters

code = double(str);      % Converts each character to a number;
                         % code is a standard Matlab vector

s = char(code);          % Converts a vector of numbers into
                         % a string (i.e., a vector of characters)
```

## Character Arithmetic

- You can do "math" with characters

```
'd' – 'a'          % Produces 3
'9' – '8'          % Produces 1
'a' < 'd'          % Produces 1 (= true)
'd' < 'b'          % Produces 0 (= false)
'Z' < 'b'          % Produces 1 (= true)
                   % Because 90, the ASCII code for 'Z',
                   % is less than 98, the ASCII code for 'b'

'a' + 2            % Produces 99
char('a'+2)        % Produces 'c'
```

## Example: toUpper

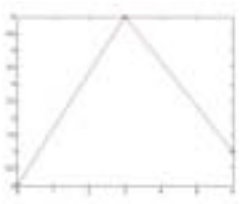
- Goal: Write toUpper( ), our own version of Matlab's upper( ), a function to convert a string to all uppercase
  - We want to do this without using Matlab's function upper( )
- Function header

```
function str = toUpper(str)
% Post: Convert string so all letters are upper case
% Pre: Input is a string
```

- Idea: Note that 'a' – 'A' has the same value as 'b' – 'B' which has the same value as 'c' – 'C', etc.
  - All we have to do is subtract the right number from a lowercase letter and we'll have the equivalent uppercase letter

## Drawing in Matlab

```
x = [0 3 6];
y = [0 5 1];
plot(x, y, '-or');
```



- This code will plot the points (0,0), (3,5), and (6,1)
- '-or' indicates
  - '-': Connect with lines
  - 'o': Mark points with circles
  - 'r': Use red
  - Use **`help plot`** to see other options

## Multiple Graphs

- Plot will take any number of arguments grouped in threes (x-values, y-values, format-info)
  - You can actually leave out the format-info (default formatting is used)
- Example: Suppose a, b, c, and d are vectors;

```
plot(a, b, '-', c, d, '*g')
```

will draw 2 graphs
  - a vs. b as a line
  - c vs. d as individual points marked by green stars
  - length(a) must equal length(b)
  - length(c) must equal length(d)

## Even Better Drawings

- You can add titles and labels to your drawings

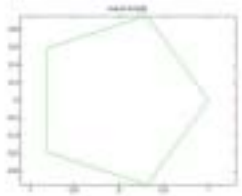
```
title('Your Fabulous Title')

xlabel('Name of x-axis')

ylabel('Name of y-axis')
```

- If you type **`help plot`** in the Command Window, there are links to these and other useful drawing-related functions

## Drawing a Circle

- Note that cosine (or sine) of a vector produces a vector

```
function makeCircle(n)
% Make circle using n segments
angles = 0 : 2*pi/n : 2*pi;
x = cos(angles);
y = sin(angles);
plot(x, y, '-g');
axis equal
```



## Example: Random Walk

- Write a function randomWalk(n) to perform n steps of a random walk in the plane starting from (0,0)
  - Function header: function randomWalk(n)
- At each step, possible moves are up, down, left, or right
- Display the walk
  - This part turns out to be easy
  - plot(x, y, '-') where x and y are vectors draws connecting lines from (x(0), y(0)) to (x(1), y(1)) to (x(2), y(2)) to…

## Random Walk Algorithm

- To do the drawing, we need all the steps stored in two vectors: x and y
- For n steps we need vectors of length n+1
- E.g., if we use vectors of length 2, we can hold
  - The starting position (0,0)
  - And one step to either (1,0), (0,1), (-1,0), or (0,-1)
- Pseudocode

```
Load x and y with n+1 zeros
for each step k
    Choose a random direction
    Update x(k+1) and y(k+1)
Draw the result
```