Previous Lecture:
- Working with images

Today’s Lecture:
- Characters and strings
- Very brief introduction to recursion—more later

Announcements:
- Section will be in the classrooms this week
- Project 4 posted. Due Thurs 10/30 at 6pm.

---

Characters & strings

- We have used strings already:
  - `n= input('Next number: ')`
  - `sprintf('Answer is %d', ans)`
- A string is made up of individual characters, so a string is a 1-d array of characters
- ’CS1112 rocks!’ is a character array of length 13; it has 7 letters, 4 digits, 1 space, and 1 symbol.

---

Single quotes enclose strings in Matlab

- Anything enclosed in single quotes is a string (even if it looks like something else)
  - ‘100’ is a character array (string) of length 3
  - 100 is a numeric value
  - ’pi’ is a character array of length 2
  - pi is the built-in constant 3.1416…
  - ’x’ is a character (vector of length 1)
  - x may be a variable name in your program

---

Comparison of genomic sequences is another example of string computation

- E.g., looking for a pattern:
  - Given the sequence ATTCTGACCTCGATC...
  - Look for the pattern ACCT
  - E.g., quantifying the difference between sequences:
    - ATTCTGACCTCGATC
    - ATTCGTGACCTCGAT
    - What if this nucleotide is removed?

---

Strings as vectors

**Vectors**
- Assignment: `v= [7 0 5];`
- Indexing: `v(1) = v(2); % x is 7, v(1) = 7`
- Appending: `v= [v, 0]; % v is [7 0 5]`
- Concatenation: `v= [v, 0]; % v is [7 0 5]`

**Strings**
- Assignment: `s= 'hello';`
- Indexing: `c= s(1); % c is 'h'`
- Appending: `s= 'duck'; % s is 'ducks'`
- Concatenation: `s= [s 'quack']; % s is 'ducks quack'`
Some useful string functions
str = 'Cs 1112';

length(str) % 7
isletter(str) % [1 1 0 0 0 0 0]
isspace(str) % [0 0 1 0 0 0 0]
lower(str) % 'cs 1112'
upper(str) % 'CS 1112'
ischar(str)
% Is str a char array? True (1)
strcmp(str(1:2),'cs')
% Compare strings str(1:2) & 'cs'. False (0)
strcmp(str(1:3),'Cs')
% False (0)

Example: capitalize 1st letter
Write a function to capitalize the first letter of each word in a string. Assume that the string has lower case letters and blanks only.

function [str, nCaps] = caps(str)
% Post: Capitalize first letter of each word.
% str = partially capitalized string
% nCaps = no. of capital letters
% Pre: str = string with lower case letters & blanks only

look for the spaces
Look For The Spaces

ASCII characters
(American Standard Code for Information Interchange)

<table>
<thead>
<tr>
<th>ascii code</th>
<th>Character</th>
<th>ascii code</th>
<th>Character</th>
</tr>
</thead>
<tbody>
<tr>
<td>:</td>
<td>:</td>
<td>:</td>
<td>:</td>
</tr>
<tr>
<td>65</td>
<td>'A'</td>
<td>48</td>
<td>'0'</td>
</tr>
<tr>
<td>66</td>
<td>'B'</td>
<td>49</td>
<td>'1'</td>
</tr>
<tr>
<td>67</td>
<td>'C'</td>
<td>50</td>
<td>'2'</td>
</tr>
<tr>
<td>:</td>
<td>:</td>
<td>:</td>
<td>:</td>
</tr>
<tr>
<td>90</td>
<td>'Z'</td>
<td>57</td>
<td>'9'</td>
</tr>
<tr>
<td>:</td>
<td>:</td>
<td>:</td>
<td>:</td>
</tr>
</tbody>
</table>

Character vs ASCII code
str = 'Age 19'
% a 1-d array of characters
code = double(str)
% convert chars to ascii values
str1 = char(code)
% convert ascii values to chars

Arithmetic and relational ops on characters
- 'c'-'a' gives 2
- '6'-'5' gives 1
- letter1='e'; letter2='f';
- letter1-letter2 gives -1
- 'c'>='a' gives true
- letter1>=letter2 gives false
- 'A' + 2 gives 67
- char('A'+2) gives 'C'

What is in variable g (if it gets created)?
d1 = 'Oct 3'; d2 = 'Oct 9';
x1 = d1(5); x2 = d2(5);
g = x2-x1;
A: the character '6'
B: the number 6
C: Error in the subtraction operation
D: Error in assigning variables x1, x2
E: Some other value or error
What is in variable g (if it gets created)?

\[ d_1 = \text{'Oct 13'}; \quad d_2 = \text{'Oct 29'}; \]
\[ x_1 = d_1(5:6); \quad x_2 = d_2(5:6); \]
\[ g = x_2 - x_1; \]

A: the string '16'
B: the number 16
C: Error in the subtraction operation
D: Error in assigning variables x1, x2
E: Some other value or error

Example: toUpper

Write a function toUpper(cha) to convert character cha to upper case if cha is a lower case letter. Return the converted letter. If cha is not a lower case letter, simply return the character cha.

Hint: Think about the distance between a letter and the base letter 'a' (or 'A'). E.g.,

\[ \begin{array}{cccccccc}
    a & b & c & d & e & f & g & h \\
    A & B & C & D & E & F & G & H \\
\end{array} \]
\[ \text{distance} = \text{'g'-'a'} = 6 = \text{'G'-'A'} \]

Of course, do not use Matlab's function upper!

Example: removing all occurrences of a character

- From a genome bank we get a sequence
  \[ \text{ATTG CCG TA GCTA CGTACGC AACTGG AAATGGC CGTAT...} \]
- First step is to “clean it up” by removing all the blanks. Write this function:

```matlab
function s = removeChar(c, s)
% Return string s with all occurrences
% of character c removed
```

Example: removing all occurrences of a character

- Can solve this problem using iteration—check one character (one component of the vector) at a time
- New strategy: recursion
  - Possible when result can be accumulated iteratively
  - E.g., remove all the blanks in string s
    Same as remove blank in s(1)
    and remove blanks in s(2:length(s))
  - E.g., capitalize first letter of all words in a sentence
    Same as capitalize 1st letter of first word
    and capitalize 1st letter of the rest of the words

There is recursion in grammar!

- A noun: paper
- A noun phrase: blue paper
- A noun phrase is a series of adjectives (possibly empty) followed by a noun. So thin blue paper is also a noun phrase

Recursive definition: A noun phrase is

- a noun
- or

an adjective followed by a noun phrase.