Previous Lecture:
- Working with images

Today’s Lecture:
- Characters and strings

Announcements:
- Discussion this week in classrooms as listed on roster
- Project 4 due Thurs 3/28 at 11pm

Characters & strings
- We have used strings already:
  - \[ \text{n= input('Next number: ')} \]
  - \[ \text{sprintf('Answer is \%d', ans)} \]
- 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.
  - Can have 2-d array of characters as well
    - \[
      \begin{bmatrix}
        \text{C} & \text{S} & 1 & 1 & 1 & 2 \\
        \text{r} & \text{o} & \text{c} & \text{k} & \text{s} & !
      \end{bmatrix}
    \]

Matlab types: char, double, uint8, logical
- \[
  \begin{bmatrix}
    \text{C} & \text{S} & 1 \\
    3 & 9
  \end{bmatrix}
\]
  - \(a\) is a 1-d array with type \text{char} components. We call \(a\) a “string” or “char array”
  - \(b\) is a 1-d array with type \text{double} components. \text{double} is the default type for numbers in Matlab. We call \(b\) a “numeric array”
  - \(c\) is a 1-d array with type \text{uint8} components. We call \(c\) a “uint8 array”
  - \(d\) is a scalar of the type \text{logical}. We call \(d\) a “boolean value”

Strings are important in computation
Numerical data is often encoded in strings. E.g., a file containing Ithaca weather data begins with the string ‘W07629N4226’
  - \text{meaning}
    - Longitude: 76° 29’ West
    - Latitude: 42° 26’ North
We may need to grab hold of the substring ‘W07629’, convert 076 and 29 to the numeric values 76 and 29, and do some computation

Comparison of genomic sequences is another example of string computation
- E.g., looking for a pattern:
  - Given the sequence \(\text{ATTCTGACCTCGATC…}\)
  - Look for the pattern \(\text{ACCT}\)
- E.g., quantifying the difference between sequences:
  - \(\text{ATTCTGACCTCGATC}\)
  - \(\text{ATTCTGACCTCGATC}\)
    - What if this nucleotide is removed?

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
Lecture slides

**Strings are vectors**

<table>
<thead>
<tr>
<th>Vectors</th>
<th>Strings</th>
</tr>
</thead>
<tbody>
<tr>
<td>Assignment</td>
<td>Assignment</td>
</tr>
<tr>
<td>( v = \begin{bmatrix} 7 &amp; 0 &amp; 5 \end{bmatrix} )</td>
<td>( s = 'hello' )</td>
</tr>
<tr>
<td>Indexing</td>
<td>Indexing</td>
</tr>
<tr>
<td>( v(i) = 1 ); ( v ) is ( \begin{bmatrix} 1 &amp; 0 &amp; 5 \end{bmatrix} )</td>
<td>( c(1) = 7 ); ( s ) is 'hello'</td>
</tr>
<tr>
<td>( w = v {2:3}; ) ( w ) is ( \begin{bmatrix} 0 &amp; 5 \end{bmatrix} )</td>
<td>( t = s {2:4}; ) ( t ) is 'ell'</td>
</tr>
<tr>
<td>( \cdot ) notation</td>
<td>( \cdot ) notation</td>
</tr>
<tr>
<td>( v = \begin{bmatrix} 2 &amp; 3 &amp; 4 \end{bmatrix} )</td>
<td>( s = 'abcde' )</td>
</tr>
<tr>
<td>Appending</td>
<td>Appending</td>
</tr>
<tr>
<td>( v = \begin{bmatrix} 7 &amp; 0 &amp; 5 \end{bmatrix} ); ( v(4) = 2 ); ( v ) is ( \begin{bmatrix} 7 &amp; 0 &amp; 5 &amp; 2 \end{bmatrix} )</td>
<td>( s = 'ducks' )</td>
</tr>
<tr>
<td>Concatenation</td>
<td>Concatenation</td>
</tr>
<tr>
<td>( v = \begin{bmatrix} 4 &amp; 6 \end{bmatrix} ); ( \cdot ) notation</td>
<td>( s = ['quack'] ); ( \cdot ) notation</td>
</tr>
<tr>
<td>( v ) is ( \begin{bmatrix} 7 &amp; 0 &amp; 5 &amp; 2 &amp; 4 &amp; 6 \end{bmatrix} )</td>
<td>( s ) is 'ducks quack'</td>
</tr>
</tbody>
</table>

**Some useful string functions**

\[
\begin{align*}
\text{str} &= 'Cs 1112'; \\
\text{length(str)} &= 7 \\
\text{isletter(str)} &= \begin{bmatrix} 1 & 1 & 0 & 0 & 0 & 0 & 0 \end{bmatrix} \\
\text{isspace(str)} &= \begin{bmatrix} 0 & 0 & 1 & 0 & 0 & 0 & 0 \end{bmatrix} \\
\text{lower(str)} &= 'cs 1112' \\
\text{upper(str)} &= 'CS 1112' \\
\text{ischar(str)} &= \text{Is str a char array? True (1)} \\
\text{strcmp(str(1:2), 'cs')} &= \text{False (0)} \\
\text{strcmp(str(1:3), 'CS')} &= \text{False (0)}
\end{align*}
\]

**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. (OK to use built-in function `upper`)

```matlab
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>
</tr>
</thead>
<tbody>
<tr>
<td>:</td>
<td>:</td>
</tr>
<tr>
<td>:</td>
<td>:</td>
</tr>
<tr>
<td>65</td>
<td>'A'</td>
</tr>
<tr>
<td>66</td>
<td>'B'</td>
</tr>
<tr>
<td>67</td>
<td>'C'</td>
</tr>
<tr>
<td>:</td>
<td>:</td>
</tr>
<tr>
<td>90</td>
<td>'Z'</td>
</tr>
<tr>
<td>:</td>
<td>:</td>
</tr>
</tbody>
</table>

**Character vs ASCII code**

\[
\text{str} = 'Age 19'
\]

\[
\begin{align*}
\text{% a 1-d array of characters} \\
\text{code} &= \text{double(str)} \\
\text{% convert chars to ascii values} \\
\text{str1} &= \text{char(code)} \\
\text{% convert ascii values to chars}
\end{align*}
\]

**Arithmetic and relational ops on characters**

- 'c' - 'a' gives 2
- '6' - '5' gives 1
- \( \text{letter1} = 'e'; \text{letter2} = 'f'; \)
- \( \text{letter1} - \text{letter2} \) gives -1
- 'c' > 'a' gives true
- \( \text{letter1} == \text{letter2} \) gives false
- 'A' + 2 gives 67
- \( \text{char('A'+2)} \) gives 'C'
What is in variable \( g \) (if it gets created)?

\[
d1 = 'Mar 3'; \quad d2 = 'Mar 9'; \\
x1 = d1(5); \quad x2 = d2(5); \\
g = x2 - x1; \\
\]

A: the character ‘6’

B: the numeric value 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)?

\[
d1 = 'Mar 13'; \quad d2 = 'Mar 29'; \\
x1 = d1(5:6); \quad x2 = d2(5:6); \\
g = x2 - x1; \\
\]

A: the string ‘16’

B: the numeric value 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 \( \text{toUpper}(\text{cha}) \) to convert character \( \text{cha} \) to upper case if \( \text{cha} \) is a lower case letter. Return the converted letter. If \( \text{cha} \) is not a lower case letter, simply return the character \( \text{cha} \).

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

\[
\begin{align*}
\text{a} & \quad \text{b} & \quad \text{c} & \quad \text{d} & \quad \text{e} & \quad \text{f} & \quad \text{g} & \quad \text{h} \\
\cdots & \cdots & \cdots & \cdots & \cdots & \cdots & \cdots & \cdots \\
\text{A} & \quad \text{B} & \quad \text{C} & \quad \text{D} & \quad \text{E} & \quad \text{F} & \quad \text{G} & \quad \text{H} \\
\end{align*}
\]

Distance = ‘g’-‘a’ = 6 = ‘G’-‘A’

Of course, do not use Matlab function \( \text{upper}! \)

Example: censoring words

\[
\begin{align*}
\text{function } D &= \text{censor}(\text{str}, \text{A}) \\
\text{function } \up &= \text{toUpper}(\text{cha}) \\
\end{align*}
\]

\[
\begin{align*}
\text{function } \up &= \text{toUpper}(\text{cha}) \\
\% \up \text{ is the upper case of character } \text{cha}. \\
\% \text{ if } \text{cha} \text{ is not a letter then } \up \text{ is just } \text{cha}. \\
\end{align*}
\]

\[
\begin{align*}
\text{function } D &= \text{censor}(\text{str}, \text{A}) \\
\% \text{ Replace all occurrences of string } \text{str} \text{ in character matrix } \text{A}, \\
\% \text{ regardless of case, with } X's, \text{ regardless of } \\
\% \text{ case.} \\
\% \text{ Assume str is never split across two lines.} \\
\% \text{ D is } \text{A} \text{ with } X's \text{ replacing str.} \\
\end{align*}
\]

\[
\begin{align*}
\text{distance} = \text{‘g’-‘a’} = 6 = \text{‘G’-‘A’} \\
\text{Of course, do not use Matlab function upper!} \\
\end{align*}
\]
function D = censor(str, A)
% Replace all occurrences of string str in character matrix A, regardless of case, with X's.
% A is a matrix of characters.
% str is a string. Assume that str is never split across two lines.
% D is A with X's replacing the censored string str.
D = A;
B = lower(A);
s = lower(str);
ns = length(str);
[nr,nc] = size(A);
% Build a string of X's of the right length
Xs = char(zeros(1,ns));
for k = 1:ns
    Xs(k) = 'X';
end
% Traverse the matrix to censor string str

Example: removing all occurrences of a character

From a genome bank we get a sequence
ATTG CCG TA GCTA CGTACGC AACTGG AAATGGC CGTAT...

First step is to “clean it up” by removing all the blanks. Write this function:

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

function s = removeChar_loop(c, s)
% Return string s with all occurrences of character c removed.
t = '';
for k = 1:length(s)
    if s(k) == c
        t = t + ' '; % Here is an array of type double
    end
    s = t;
end

Example: removing all occurrences of a character

Can solve this problem using iteration—check one character (one component of the vector) at a time

function s = removeChar_loop(c, s)
% Return string s with all occurrences of character c removed.
t = '';
for k = 1:length(s)
    if s(k) == c
        t = t + ' '; % Here is an array of type double
    end
    s = t;
end