Previous Lecture:
- Working with images
- Working with type uint8

Today’s Lecture:
- Characters and strings

Announcement:
- Project 4 due Mon 10/24 at 11pm
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.

```
'C'S'1'1'1'2' 'r'o'c'k's'!
```
- Can have 2-d array of characters as well

```
C'S'1'1'1'2'
'r'o'c'k's'!
```

2×6 matrix
Matlab types: char, double, uint8, logical

There is not a type “string”! What we call a “string” is a 1-d array of chars

\[ a = \text{'C'S'1'} \]

\( a \) is a 1-d array with type char components. We call \( a \) a “string” or “char array”

\[ b = [3 \ 9] \]

\( b \) is a 1-d array with type double components. double is the default type for numbers in Matlab. We call \( b \) a “numeric array”

\[ c = \text{uint8}(b) \]

\( c \) is a 1-d array with type uint8 components. We call \( c \) a “uint8 array”

\[ d = \text{rand} > .5 \]

\( d \) is a scalar of the type 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 \texttt{W07629N4226}

meaning

\begin{itemize}
  \item Longitude: \textit{76}\degree\ 29’ West
  \item Latitude: \textit{42}\degree\ 26’ North
\end{itemize}

We may need to grab hold of the substring \texttt{W07629}, convert \texttt{076} and \texttt{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 \textbf{ATTCTGACCTCGATC...}
  Look for the pattern \textbf{ACCT}

- E.g., quantifying the difference between sequences:
  \textbf{ATTCTGACCTCGATC}
  \textbf{ATTCGTGACCTCGAT}

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
Strings are vectors

Vectors

- **Assignment**
  
  \[ v = [7 0 5]; \]

- **Indexing**
  
  \[ x = v(3); \quad \% x \text{ is } 5 \]
  \[ v(1)= 1; \quad \% v \text{ is } [1 0 5] \]
  \[ w= v(2:3); \quad \% w \text{ is } [0 5] \]

- **: notation**
  
  \[ v= 2:5; \quad \% v \text{ is } [2 3 4 5] \]

- **Appending**
  
  \[ v= [7 0 5]; \]
  \[ v(4)= 2; \quad \% v \text{ is } [7 0 5 2] \]

- **Concatenation**
  
  \[ v= [v [4 6]]; \]
  \[ \% v \text{ is } [7 0 5 2 4 6] \]

Strings

- **Assignment**
  
  \[ s= \text{ ‘hello’}; \]

- **Indexing**
  
  \[ c= s(2); \quad \% c \text{ is } \text{ ‘e’} \]
  \[ s(1)= \text{ ‘j’}; \quad \% s \text{ is } \text{ ‘jello’} \]
  \[ t= s(2:4); \quad \% t \text{ is } \text{ ‘ell’} \]

- **: notation**
  
  \[ s= \text{ ‘a’}:'g'; \quad \% s \text{ is } \text{ ‘abcdefg’} \]

- **Appending**
  
  \[ s= \text{ ‘duck’}; \]
  \[ s(5)= ‘s’; \quad \% s \text{ is } \text{ ‘ducks’} \]

- **Concatenation**
  
  \[ s= [s ‘ \text{ quack’} ]; \]
  \[ \% s \text{ is } \text{ ‘ducks quack’} \]
Some useful string functions

\texttt{str= 'Cs 1112';}

\texttt{length(str) \hspace{1em} % 7}
\texttt{isletter(str) \hspace{1em} % [1 1 0 0 0 0 0 0]}
\texttt{isspace(str) \hspace{1em} % [0 0 1 0 0 0 0 0]}
\texttt{lower(str) \hspace{1em} % 'cs 1112'}
\texttt{upper(str) \hspace{1em} % 'CS 1112'}

\texttt{ischar(str)}
\hspace{1em} % \texttt{Is str a char array? True (1)}
\texttt{strcmp(str(1:2), 'cs')} \hspace{1em} % \texttt{Compare strings str(1:2) & 'cs'. False (0)}
\texttt{strcmp(str(1:3), 'CS')} \hspace{1em} % \texttt{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. (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
```

See `caps.m`
## The ASCII Table

<table>
<thead>
<tr>
<th>Char</th>
<th>Dec</th>
<th>Oct</th>
<th>Hex</th>
<th>Char</th>
<th>Dec</th>
<th>Oct</th>
<th>Hex</th>
<th>Char</th>
<th>Dec</th>
<th>Oct</th>
<th>Hex</th>
</tr>
</thead>
<tbody>
<tr>
<td>(null)</td>
<td>0000</td>
<td>0x00</td>
<td>0x00</td>
<td>(sp)</td>
<td>0040</td>
<td>0x20</td>
<td>0x40</td>
<td>(@)</td>
<td>0100</td>
<td>0x40</td>
<td>0x60</td>
</tr>
<tr>
<td>(soh)</td>
<td>0001</td>
<td>0x01</td>
<td>0x01</td>
<td>!</td>
<td>0041</td>
<td>0x21</td>
<td>0x41</td>
<td>a</td>
<td>0141</td>
<td>0x41</td>
<td>0x61</td>
</tr>
<tr>
<td>(stx)</td>
<td>0002</td>
<td>0x02</td>
<td>0x02</td>
<td>&quot;</td>
<td>0042</td>
<td>0x22</td>
<td>0x42</td>
<td>b</td>
<td>0142</td>
<td>0x42</td>
<td>0x62</td>
</tr>
<tr>
<td>(etx)</td>
<td>0003</td>
<td>0x03</td>
<td>0x03</td>
<td>#</td>
<td>0043</td>
<td>0x23</td>
<td>0x43</td>
<td>c</td>
<td>0143</td>
<td>0x43</td>
<td>0x63</td>
</tr>
<tr>
<td>(eot)</td>
<td>0004</td>
<td>0x04</td>
<td>0x04</td>
<td>$</td>
<td>0044</td>
<td>0x24</td>
<td>0x44</td>
<td>d</td>
<td>0144</td>
<td>0x44</td>
<td>0x64</td>
</tr>
<tr>
<td>(eng)</td>
<td>0005</td>
<td>0x05</td>
<td>0x05</td>
<td>%</td>
<td>0045</td>
<td>0x25</td>
<td>0x45</td>
<td>e</td>
<td>0145</td>
<td>0x45</td>
<td>0x65</td>
</tr>
<tr>
<td>(ack)</td>
<td>0006</td>
<td>0x06</td>
<td>0x06</td>
<td>&amp;</td>
<td>0046</td>
<td>0x26</td>
<td>0x46</td>
<td>f</td>
<td>0146</td>
<td>0x46</td>
<td>0x66</td>
</tr>
<tr>
<td>(bel)</td>
<td>0007</td>
<td>0x07</td>
<td>0x07</td>
<td>'</td>
<td>0047</td>
<td>0x27</td>
<td>0x47</td>
<td>g</td>
<td>0147</td>
<td>0x47</td>
<td>0x67</td>
</tr>
<tr>
<td>(bs)</td>
<td>0010</td>
<td>0x08</td>
<td>0x08</td>
<td>(</td>
<td>0040</td>
<td>0x28</td>
<td>0x48</td>
<td>h</td>
<td>0148</td>
<td>0x48</td>
<td>0x68</td>
</tr>
<tr>
<td>(ht)</td>
<td>0011</td>
<td>0x09</td>
<td>0x09</td>
<td>)</td>
<td>0041</td>
<td>0x29</td>
<td>0x49</td>
<td>i</td>
<td>0149</td>
<td>0x49</td>
<td>0x69</td>
</tr>
<tr>
<td>(nl)</td>
<td>0012</td>
<td>0x0a</td>
<td>0x0a</td>
<td>*</td>
<td>0042</td>
<td>0x2a</td>
<td>0x4a</td>
<td>j</td>
<td>014a</td>
<td>0x4a</td>
<td>0x6a</td>
</tr>
<tr>
<td>(vt)</td>
<td>0013</td>
<td>0x0b</td>
<td>0x0b</td>
<td>+</td>
<td>0043</td>
<td>0x2b</td>
<td>0x4b</td>
<td>k</td>
<td>014b</td>
<td>0x4b</td>
<td>0x6b</td>
</tr>
<tr>
<td>(np)</td>
<td>0014</td>
<td>0x0c</td>
<td>0x0c</td>
<td>,</td>
<td>0044</td>
<td>0x2c</td>
<td>0x4c</td>
<td>l</td>
<td>014c</td>
<td>0x4c</td>
<td>0x6c</td>
</tr>
<tr>
<td>(cr)</td>
<td>0015</td>
<td>0x0d</td>
<td>0x0d</td>
<td>-</td>
<td>0045</td>
<td>0x2d</td>
<td>0x4d</td>
<td>m</td>
<td>014d</td>
<td>0x4d</td>
<td>0x6d</td>
</tr>
<tr>
<td>(so)</td>
<td>0016</td>
<td>0x0e</td>
<td>0x0e</td>
<td>.</td>
<td>0046</td>
<td>0x2e</td>
<td>0x4e</td>
<td>n</td>
<td>014e</td>
<td>0x4e</td>
<td>0x6e</td>
</tr>
<tr>
<td>(si)</td>
<td>0017</td>
<td>0x0f</td>
<td>0x0f</td>
<td>/</td>
<td>0047</td>
<td>0x2f</td>
<td>0x4f</td>
<td>o</td>
<td>014f</td>
<td>0x4f</td>
<td>0x6f</td>
</tr>
<tr>
<td>(dle)</td>
<td>0020</td>
<td>0x10</td>
<td>0x10</td>
<td>0</td>
<td>0048</td>
<td>0x30</td>
<td>0x30</td>
<td>p</td>
<td>0150</td>
<td>0x50</td>
<td>0x70</td>
</tr>
<tr>
<td>(dc1)</td>
<td>0021</td>
<td>0x11</td>
<td>0x11</td>
<td>1</td>
<td>0049</td>
<td>0x31</td>
<td>0x31</td>
<td>q</td>
<td>0151</td>
<td>0x51</td>
<td>0x71</td>
</tr>
<tr>
<td>(dc2)</td>
<td>0022</td>
<td>0x12</td>
<td>0x12</td>
<td>2</td>
<td>004a</td>
<td>0x32</td>
<td>0x32</td>
<td>r</td>
<td>0152</td>
<td>0x52</td>
<td>0x72</td>
</tr>
<tr>
<td>(dc3)</td>
<td>0023</td>
<td>0x13</td>
<td>0x13</td>
<td>3</td>
<td>004b</td>
<td>0x33</td>
<td>0x33</td>
<td>s</td>
<td>0153</td>
<td>0x53</td>
<td>0x73</td>
</tr>
<tr>
<td>(dc4)</td>
<td>0024</td>
<td>0x14</td>
<td>0x14</td>
<td>4</td>
<td>004c</td>
<td>0x34</td>
<td>0x34</td>
<td>t</td>
<td>0154</td>
<td>0x54</td>
<td>0x74</td>
</tr>
<tr>
<td>(nak)</td>
<td>0025</td>
<td>0x15</td>
<td>0x15</td>
<td>5</td>
<td>004d</td>
<td>0x35</td>
<td>0x35</td>
<td>u</td>
<td>0155</td>
<td>0x55</td>
<td>0x75</td>
</tr>
<tr>
<td>(syn)</td>
<td>0026</td>
<td>0x16</td>
<td>0x16</td>
<td>6</td>
<td>004e</td>
<td>0x36</td>
<td>0x36</td>
<td>v</td>
<td>0156</td>
<td>0x56</td>
<td>0x76</td>
</tr>
<tr>
<td>(etb)</td>
<td>0027</td>
<td>0x17</td>
<td>0x17</td>
<td>7</td>
<td>004f</td>
<td>0x37</td>
<td>0x37</td>
<td>w</td>
<td>0157</td>
<td>0x57</td>
<td>0x77</td>
</tr>
<tr>
<td>(can)</td>
<td>0030</td>
<td>0x18</td>
<td>0x18</td>
<td>8</td>
<td>0050</td>
<td>0x38</td>
<td>0x38</td>
<td>x</td>
<td>0158</td>
<td>0x58</td>
<td>0x78</td>
</tr>
<tr>
<td>(em)</td>
<td>0031</td>
<td>0x19</td>
<td>0x19</td>
<td>9</td>
<td>0051</td>
<td>0x39</td>
<td>0x39</td>
<td>y</td>
<td>0159</td>
<td>0x59</td>
<td>0x79</td>
</tr>
<tr>
<td>(sub)</td>
<td>0032</td>
<td>0x1a</td>
<td>0x1a</td>
<td>:</td>
<td>0052</td>
<td>0x3a</td>
<td>0x3a</td>
<td>z</td>
<td>015a</td>
<td>0x5a</td>
<td>0x7a</td>
</tr>
<tr>
<td>(esc)</td>
<td>0033</td>
<td>0x1b</td>
<td>0x1b</td>
<td>;</td>
<td>0053</td>
<td>0x3b</td>
<td>0x3b</td>
<td></td>
<td></td>
<td>015b</td>
<td>0x5b</td>
</tr>
<tr>
<td>(fs)</td>
<td>0034</td>
<td>0x1c</td>
<td>0x1c</td>
<td>&lt;</td>
<td>0054</td>
<td>0x3c</td>
<td>0x3c</td>
<td>\</td>
<td>015c</td>
<td>0x5c</td>
<td>0x7c</td>
</tr>
<tr>
<td>(gs)</td>
<td>0035</td>
<td>0x1d</td>
<td>0x1d</td>
<td>=</td>
<td>0055</td>
<td>0x3d</td>
<td>0x3d</td>
<td></td>
<td></td>
<td>015d</td>
<td>0x5d</td>
</tr>
<tr>
<td>(rs)</td>
<td>0036</td>
<td>0x1e</td>
<td>0x1e</td>
<td>&gt;</td>
<td>0056</td>
<td>0x3e</td>
<td>0x3e</td>
<td></td>
<td></td>
<td>015e</td>
<td>0x5e</td>
</tr>
<tr>
<td>(us)</td>
<td>0037</td>
<td>0x1f</td>
<td>0x1f</td>
<td>?</td>
<td>0057</td>
<td>0x3f</td>
<td>0x3f</td>
<td></td>
<td></td>
<td>015f</td>
<td>0x5f</td>
</tr>
<tr>
<td>(del)</td>
<td>127</td>
<td>0x7f</td>
<td>0x7f</td>
<td></td>
<td></td>
<td></td>
<td></td>
<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)?

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

A: the character ‘6’
B: the numeric value 6
C: Error in assigning variables \( x1, x2 \)
D: Error in the subtraction operation
E: Some other value or error
What is in variable \( g \) (if it gets created)?

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

A: the string '16'
B: the numeric value 16
C: Error in assigning variables \( x1, x2 \)
D: Error in the subtraction operation
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.,

\[
\begin{array}{cccccccccc}
\text{a} & \text{b} & \text{c} & \text{d} & \text{e} & \text{f} & \text{g} & \text{h} & \ldots \\
| & | & | & | & | & | & | & |
\end{array}
\]

\[
\begin{array}{cccccccccc}
\text{A} & \text{B} & \text{C} & \text{D} & \text{E} & \text{F} & \text{G} & \text{H} & \ldots \\
| & | & | & | & | & | & | & |
\end{array}
\]

\[
distance = ‘g’-‘a’ = 6 = ‘G’-‘A’
\]

Of course, do not use Matlab function `upper`!
function up = toUpper(cha)
% up is the upper case of character cha.
% If cha is not a letter then up is just cha.

up= cha;

cha is lower case if it is between 'a' and 'z'
function up = toUpper(cha)
% up is the upper case of character cha.
% If cha is not a letter then up is just cha.

up = cha;

if ( cha >= 'a' && cha <= 'z' )

    % Find distance of cha from ‘a’
end
function up = toUpper(cha)
% up is the upper case of character cha.
% If cha is not a letter then up is just cha.

up= cha;

if ( cha >= 'a' && cha <= 'z' )

    % Find distance of cha from 'a'
    offset= cha - 'a';

    % Go same distance from 'A'
    up= char('A' + offset);

end
Example: removing all occurrences of a character

- From a genome bank we get a sequence
  \begin{verbatim}
  ATTG CCG TA GCTA CGTACGC AACTGG AAATGGGC CGTAT...
  \end{verbatim}

- 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

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

    t = ''; % Initialize t as an empty string
    for k = 1:length(s) % Loop through each character in s
        end % End the loop
    s = t; % Return the modified string s
```
Example: removing all occurrences of a character

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

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

    t = ''; % Initialize t as an empty string
    for k = 1:length(s) % Loop through each character in s
        if s(k) ~= c % Check if the current character is not equal to c
            t = [t s(k)]; % Append the character to t if not equal to c
        end
    end
    s = t; % Assign the modified t back to s
```

Lecture 17