Today’s Lecture:
- 1-d and 2-d arrays of type `char`
- Computing with characters

Announcements:
- A1 resubmissions currently being graded
- Assignment 2 first submission due **Monday 9/27**
- Mon lecture: Review session for Test 1
- Test on **Wed 9/29** in Thurston 205, 2:40-3:30pm
Character array (an array of type `char`)

- We have used strings of characters in programs already:
  - `n = input('Next number: ')`
  - `sprintf('Answer is %f', ans)`

- A string is made up of individual characters, so a string is a 1-d array of characters

- `'CS1132 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

```plaintext
'C'S'1'1'3'2'  'r'o'c''k''s'!'

'C''S''1''1''3''2'
'r''o''c''k''s''!'  2x6 matrix
```
Recap: Single quotes enclose char arrays 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 3.14159...
- 'x' is a character (vector of length 1)
- x may be a variable name in your program
Types so far: char, double, logical

\[ a = 'CS1' \]
\[ a = ['C', 'S', '1'] \]
\[ b = [3, 9] \]
\[ d = \text{rand()} > .5 \]

\textbf{a} is a 1-d array with type \texttt{char} components. Often called a \textit{string}; NOT the same as a \textit{new} type in Matlab 2017+ called \texttt{string}.

\[ a \quad \boxed{\text{'C'S1'}} \]

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

\[ b \]

\textbf{d} is a scalar of the type \texttt{logical}. We call \textbf{d} a “Boolean value”

\[ d \]
Basic (simple) types in MATLAB

• E.g., char, double, logical
• Each uses a set amount of memory
  • Each double value uses 64 bits (=8 bytes)
  • Each char value uses 16 bits (=2 bytes)
  • Use function whos to see memory usage by variables in workspace
• Can easily determine amount of memory used by a simple array
  (array of a basic type, where each component stores one simple
   value)
• Later: Special arrays where each component is a container for a
  collection of values
Text—sequences of characters often called strings—are important in computation.

Numerical data is often encoded in strings. E.g., a file containing Ithaca weather data begins with the string

\[\text{W07629N4226}\]

meaning

\[
\begin{align*}
\text{Longitude:} & \quad 76^\circ 29' \text{ West} \\
\text{Latitude:} & \quad 42^\circ 26' \text{ North}
\end{align*}
\]

We may need to grab hold of the substring \text{W07629}, convert 076 and 29 to the numeric values 76 and 29, and do some computation.
A text sequence is a vector (of characters)

### Vectors

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

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

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

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

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

### Strings

- **Assignment**
  
  \[
  s = [\text{'h'}, \text{'e'}, \text{'l'}, \text{'l'}, \text{'o'}];
  \quad % \text{formal}
  
  s = \text{'hello'}; \quad % \text{shortcut}
  \]

- **Indexing**
  
  \[
  c = s(2); \quad % c is \text{'e'}
  
  s(1) = \text{'J'}; \quad % s is \text{'Jello'}
  
  t = s(2:4); \quad % t is \text{'ell'}
  \]

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

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

- **Concatenation**
  
  \[
  s = [s \ ' quack']; \quad % s is \text{'ducks quack'}
  \]
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 char array s with all occurrences of
% char scalar 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

• Challenge: Can you solve it using logical indexing?

```matlab
function t = removeChar_loop(c, s)
% Return char array s with all % occurrences of char scalar c % removed.
    t= ''; % Return char array s with all % occurrences of char scalar c % removed.
    t= ''; % Return char array s with all % occurrences of char scalar c % removed.
    for k = 1:length(s)
        if s(k) ~= c
            t= [t s(k)];
        end
    end
end
```
Some useful char array functions

s = 'Matlab 1132';

length(s)    % 11
isletter(s)  % [1 1 1 1 1 1 0 0 0 0 0]
ispace(s)    % [0 0 0 0 0 0 1 0 0 0 0]
lower(s)     % 'matlab 1132'
upper(s)     % 'MATLAB 1112'

ischar(s)
    % Is s a char array? True (1)
strcmpr(s(1:3), 'mat')
    % Compare strings str(1:3) & 'mat'. False (0)
strcmpr(s(1:3), 'Ma')
    % False (0)
<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>
<th>Char</th>
<th>Dec</th>
<th>Oct</th>
<th>Hex</th>
</tr>
</thead>
<tbody>
<tr>
<td>(nul)</td>
<td>0</td>
<td>0000</td>
<td>0x00</td>
<td>(sp)</td>
<td>32</td>
<td>0040</td>
<td>0x20</td>
<td>@</td>
<td>64</td>
<td>0100</td>
<td>0x40</td>
<td>~</td>
<td>96</td>
<td>0140</td>
<td>0x60</td>
</tr>
<tr>
<td>(soh)</td>
<td>1</td>
<td>0001</td>
<td>0x01</td>
<td>!</td>
<td>33</td>
<td>0041</td>
<td>0x21</td>
<td>A</td>
<td>65</td>
<td>0101</td>
<td>0x41</td>
<td>a</td>
<td>97</td>
<td>0141</td>
<td>0x61</td>
</tr>
<tr>
<td>(stx)</td>
<td>2</td>
<td>0002</td>
<td>0x02</td>
<td>&quot;</td>
<td>34</td>
<td>0042</td>
<td>0x22</td>
<td>B</td>
<td>66</td>
<td>0102</td>
<td>0x42</td>
<td>b</td>
<td>98</td>
<td>0142</td>
<td>0x62</td>
</tr>
<tr>
<td>(etx)</td>
<td>3</td>
<td>0003</td>
<td>0x03</td>
<td>#</td>
<td>35</td>
<td>0043</td>
<td>0x23</td>
<td>C</td>
<td>67</td>
<td>0103</td>
<td>0x43</td>
<td>c</td>
<td>99</td>
<td>0143</td>
<td>0x63</td>
</tr>
<tr>
<td>(eot)</td>
<td>4</td>
<td>0004</td>
<td>0x04</td>
<td>$</td>
<td>36</td>
<td>0044</td>
<td>0x24</td>
<td>D</td>
<td>68</td>
<td>0104</td>
<td>0x44</td>
<td>d</td>
<td>100</td>
<td>0144</td>
<td>0x64</td>
</tr>
<tr>
<td>(eng)</td>
<td>5</td>
<td>0005</td>
<td>0x05</td>
<td>%</td>
<td>37</td>
<td>0045</td>
<td>0x25</td>
<td>E</td>
<td>69</td>
<td>0105</td>
<td>0x45</td>
<td>e</td>
<td>101</td>
<td>0145</td>
<td>0x65</td>
</tr>
<tr>
<td>(ack)</td>
<td>6</td>
<td>0006</td>
<td>0x06</td>
<td>&amp;</td>
<td>38</td>
<td>0046</td>
<td>0x26</td>
<td>F</td>
<td>70</td>
<td>0106</td>
<td>0x46</td>
<td>f</td>
<td>102</td>
<td>0146</td>
<td>0x66</td>
</tr>
<tr>
<td>(bel)</td>
<td>7</td>
<td>0007</td>
<td>0x07</td>
<td>'</td>
<td>39</td>
<td>0047</td>
<td>0x27</td>
<td>G</td>
<td>71</td>
<td>0107</td>
<td>0x47</td>
<td>g</td>
<td>103</td>
<td>0147</td>
<td>0x67</td>
</tr>
<tr>
<td>(bs)</td>
<td>8</td>
<td>0010</td>
<td>0x08</td>
<td>(</td>
<td>40</td>
<td>0050</td>
<td>0x28</td>
<td>H</td>
<td>72</td>
<td>0110</td>
<td>0x48</td>
<td>h</td>
<td>104</td>
<td>0150</td>
<td>0x68</td>
</tr>
<tr>
<td>(ht)</td>
<td>9</td>
<td>0011</td>
<td>0x09</td>
<td>)</td>
<td>41</td>
<td>0051</td>
<td>0x29</td>
<td>I</td>
<td>73</td>
<td>0111</td>
<td>0x49</td>
<td>i</td>
<td>105</td>
<td>0151</td>
<td>0x69</td>
</tr>
<tr>
<td>(nl)</td>
<td>10</td>
<td>0012</td>
<td>0x0a</td>
<td>*</td>
<td>42</td>
<td>0052</td>
<td>0x2a</td>
<td>J</td>
<td>74</td>
<td>0112</td>
<td>0x4a</td>
<td>j</td>
<td>106</td>
<td>0152</td>
<td>0x6a</td>
</tr>
<tr>
<td>(vt)</td>
<td>11</td>
<td>0013</td>
<td>0x0b</td>
<td>+</td>
<td>43</td>
<td>0053</td>
<td>0x2b</td>
<td>K</td>
<td>75</td>
<td>0113</td>
<td>0x4b</td>
<td>k</td>
<td>107</td>
<td>0153</td>
<td>0x6b</td>
</tr>
<tr>
<td>(np)</td>
<td>12</td>
<td>0014</td>
<td>0x0c</td>
<td>,</td>
<td>44</td>
<td>0054</td>
<td>0x2c</td>
<td>L</td>
<td>76</td>
<td>0114</td>
<td>0x4c</td>
<td>l</td>
<td>108</td>
<td>0154</td>
<td>0x6c</td>
</tr>
<tr>
<td>(cr)</td>
<td>13</td>
<td>0015</td>
<td>0x0d</td>
<td>-</td>
<td>45</td>
<td>0055</td>
<td>0x2d</td>
<td>M</td>
<td>77</td>
<td>0115</td>
<td>0x4d</td>
<td>m</td>
<td>109</td>
<td>0155</td>
<td>0x6d</td>
</tr>
<tr>
<td>(so)</td>
<td>14</td>
<td>0016</td>
<td>0x0e</td>
<td>.</td>
<td>46</td>
<td>0056</td>
<td>0x2e</td>
<td>N</td>
<td>78</td>
<td>0116</td>
<td>0x4e</td>
<td>n</td>
<td>110</td>
<td>0156</td>
<td>0x6e</td>
</tr>
<tr>
<td>(si)</td>
<td>15</td>
<td>0017</td>
<td>0x0f</td>
<td>/</td>
<td>47</td>
<td>0057</td>
<td>0x2f</td>
<td>O</td>
<td>79</td>
<td>0117</td>
<td>0x4f</td>
<td>o</td>
<td>111</td>
<td>0157</td>
<td>0x6f</td>
</tr>
<tr>
<td>(dle)</td>
<td>16</td>
<td>0020</td>
<td>0x10</td>
<td>0</td>
<td>48</td>
<td>0060</td>
<td>0x30</td>
<td>P</td>
<td>80</td>
<td>0120</td>
<td>0x50</td>
<td>p</td>
<td>112</td>
<td>0160</td>
<td>0x70</td>
</tr>
<tr>
<td>(dc1)</td>
<td>17</td>
<td>0021</td>
<td>0x11</td>
<td>1</td>
<td>49</td>
<td>0061</td>
<td>0x31</td>
<td>Q</td>
<td>81</td>
<td>0121</td>
<td>0x51</td>
<td>q</td>
<td>113</td>
<td>0161</td>
<td>0x71</td>
</tr>
<tr>
<td>(dc2)</td>
<td>18</td>
<td>0022</td>
<td>0x12</td>
<td>2</td>
<td>50</td>
<td>0062</td>
<td>0x32</td>
<td>R</td>
<td>82</td>
<td>0122</td>
<td>0x52</td>
<td>r</td>
<td>114</td>
<td>0162</td>
<td>0x72</td>
</tr>
<tr>
<td>(dc3)</td>
<td>19</td>
<td>0023</td>
<td>0x13</td>
<td>3</td>
<td>51</td>
<td>0063</td>
<td>0x33</td>
<td>S</td>
<td>83</td>
<td>0123</td>
<td>0x53</td>
<td>s</td>
<td>115</td>
<td>0163</td>
<td>0x73</td>
</tr>
<tr>
<td>(dc4)</td>
<td>20</td>
<td>0024</td>
<td>0x14</td>
<td>4</td>
<td>52</td>
<td>0064</td>
<td>0x34</td>
<td>T</td>
<td>84</td>
<td>0124</td>
<td>0x54</td>
<td>t</td>
<td>116</td>
<td>0164</td>
<td>0x74</td>
</tr>
<tr>
<td>(nak)</td>
<td>21</td>
<td>0025</td>
<td>0x15</td>
<td>5</td>
<td>53</td>
<td>0065</td>
<td>0x35</td>
<td>U</td>
<td>85</td>
<td>0125</td>
<td>0x55</td>
<td>u</td>
<td>117</td>
<td>0165</td>
<td>0x75</td>
</tr>
<tr>
<td>(syn)</td>
<td>22</td>
<td>0026</td>
<td>0x16</td>
<td>6</td>
<td>54</td>
<td>0066</td>
<td>0x36</td>
<td>V</td>
<td>86</td>
<td>0126</td>
<td>0x56</td>
<td>v</td>
<td>118</td>
<td>0166</td>
<td>0x76</td>
</tr>
<tr>
<td>(etb)</td>
<td>23</td>
<td>0027</td>
<td>0x17</td>
<td>7</td>
<td>55</td>
<td>0067</td>
<td>0x37</td>
<td>W</td>
<td>87</td>
<td>0127</td>
<td>0x57</td>
<td>w</td>
<td>119</td>
<td>0167</td>
<td>0x77</td>
</tr>
<tr>
<td>(can)</td>
<td>24</td>
<td>0030</td>
<td>0x18</td>
<td>8</td>
<td>56</td>
<td>0070</td>
<td>0x38</td>
<td>X</td>
<td>88</td>
<td>0130</td>
<td>0x58</td>
<td>x</td>
<td>120</td>
<td>0170</td>
<td>0x78</td>
</tr>
<tr>
<td>(em)</td>
<td>25</td>
<td>0031</td>
<td>0x19</td>
<td>9</td>
<td>57</td>
<td>0071</td>
<td>0x39</td>
<td>Y</td>
<td>89</td>
<td>0131</td>
<td>0x59</td>
<td>y</td>
<td>121</td>
<td>0171</td>
<td>0x79</td>
</tr>
<tr>
<td>(sub)</td>
<td>26</td>
<td>0032</td>
<td>0x1a</td>
<td>:</td>
<td>58</td>
<td>0072</td>
<td>0x3a</td>
<td>Z</td>
<td>90</td>
<td>0132</td>
<td>0x5a</td>
<td>z</td>
<td>122</td>
<td>0172</td>
<td>0x7a</td>
</tr>
<tr>
<td>(esc)</td>
<td>27</td>
<td>0033</td>
<td>0x1b</td>
<td>;</td>
<td>59</td>
<td>0073</td>
<td>0x3b</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>(fs)</td>
<td>28</td>
<td>0034</td>
<td>0x1c</td>
<td>&lt;</td>
<td>60</td>
<td>0074</td>
<td>0x3c</td>
<td>\</td>
<td>92</td>
<td>0134</td>
<td>0x5c</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>(gs)</td>
<td>29</td>
<td>0035</td>
<td>0x1d</td>
<td>=</td>
<td>61</td>
<td>0075</td>
<td>0x3d</td>
<td>]</td>
<td>93</td>
<td>0135</td>
<td>0x5d</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>(rs)</td>
<td>30</td>
<td>0036</td>
<td>0x1e</td>
<td>&gt;</td>
<td>62</td>
<td>0076</td>
<td>0x3e</td>
<td>^</td>
<td>94</td>
<td>0136</td>
<td>0x5e</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>(us)</td>
<td>31</td>
<td>0037</td>
<td>0x1f</td>
<td>?</td>
<td>63</td>
<td>0077</td>
<td>0x3f</td>
<td>_</td>
<td>95</td>
<td>0137</td>
<td>0x5f</td>
<td>(del)</td>
<td>127</td>
<td>0177</td>
<td>0x7f</td>
</tr>
</tbody>
</table>
# 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>:</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 Unicode code points

str = 'Age 19'

% a 1-d array of characters
code = double(str)

% convert chars to Unicode values
str1 = char(code)

% convert Unicode 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'; d2= 'Mar 9';
x1= d1(5); x2= d2(5);
g= x2-x1;

Alfa: the character ‘6’
Bravo: the numeric value 6
Charlie: Error in assigning variables $x1$, $x2$
Delta: Error in the subtraction operation
Echo: 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;

Alfa: the string ‘16’
Bravo: the numeric value 16
Charlie: Error in assigning variables $x1$, $x2$
Delta: Error in the subtraction operation
Echo: 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.,

```
  a b c d e f g h ...
      |   |   |   |
    g   a

  A B C D E F G H ...
```

\[ \text{distance} = 'g'-'a' = 6 = 'G'-'A' \]

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.
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'
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: censoring words

```
function D = censor(str, A)
% Replace all occurrences of string str in
% character matrix A with X's, regardless of
% case.
% Assume str is never split across two lines.
% D is A with X’s replacing str.
```

Function `strcmpi` does case-insensitive string comparison.
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;
ns= length(str);
[nr,nc]= size(A);

% Build a string of X's of the right length

% Traverse the matrix to censor string str
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;
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
for r= 1:nr
    for c= 1:nc-ns+1
        if strcmpi( str, A(r, c:c+ns-1) )
            D(r, c:c+ns-1)= Xs;
        end
    end
end

Returns an array of type double
Changes the type to char

Case insensitive comparison of strings