Project 11  Strings and Character Processing

Objectives: Learn to input and manipulate character data.
Outcome:

- Write functions that process character data including converting character data to numerical codes.

1. Back to MATLAB

1. Start up MATLAB. Remember to set your directory to My Documents
2. For help with today’s topic use help with double char upper num2str int2str str2num

2. Character Data

1. Character data is usually in the form of a string which is a set of characters enclosed by single quotes, like 'Hi World!' or 'This is a string'. This is a value so it can be assigned to a variable like st = 'my string'.
2. A string is really just an array of characters so if S = 'ABCDE' then S(3) is 'C', S(5) is 'E', etc.
3. Besides setting a variable equal to a string, you can use the input command to have the user enter a string, but you must let MATLAB know you are expecting a string as input, rather than a number. To do so we use this form of the input statement:

   S = input('Enter a string:','s');

   Note that even if you enter numerical data, MATLAB will treat it as a string.

3. Character Conversion

1. Strings can be created by converting numerical data to strings using various commands. It is important to remember that there is a difference between what the string says and the code behind the string. For example the string '1' represents the value 1 but is also represented by the code 49 and we use different commands to access the different parts.
2. To convert a string that represents a number into that number, use str2num. To convert a number into a string you can use int2str (for integers) or num2str (for other numbers). There is also a variation on fprintf called sprintf which uses format strings to produce strings from various data.
3. To get the data from a string use double. This will produce one (ascii) code per character in the string. To convert ascii into characters use char.

4. Try these examples: (type or copy and paste)

   s = '12.3';
   str2num(s)
   double(s)
   x = 12.3;
   int2str(x)
   num2str(x)
   sprintf('%5.2f',x)
   char([77 97 116 104])

4. To Do and Turn In

1. The result will be a set of functions and a script that calls them that form the beginning of a program to do some cryptography (code - spy stuff).
2. Write a function that takes a string as input and produces an array containing a code for each letter in the string with A=0, B=1, ..., Z=25 and all other characters getting the value 26. For example if the input is 'Math!' the output should be [12 0 19 7 26]. We’ll call this the code array.

   I’m not going to give you the specific code to use, but I will give you the flow and the commands. If you want to see examples of their use, use the help command.

   (a) Convert the string to all upper case (upper).
   (b) Convert the string to an array of ascii codes (double).
   (c) Loop through the array and change the codes to the right values (for, length). Check the value of double('A') and double('Z'). All other characters will be outside this range.

3. Write a function that takes a code array and returns the corresponding string (using a space for any of the extra characters). So if the input was [12 0 19 7 25] the output would be 'MATH'.
4. Write a function that takes a code array and counts the number of As, Bs, ..., Zs in the array (the frequency) and displays the results in a nicely formatted table.

(a) You’ll need an array that has 27 slots to accumulate your count. It should start containing all zeros. You can use a loop to do this or use the command `zeros(1,27)`

(b) Loop through the code array and count the occurrences of each letter. The easiest way to do this is to use each code array value to get the index for the accumulation array. For example if you are putting your counts in an array `counts` and you have a code array `carray` then `j = carray(i)+1` would give you 1 for an A, 2 for a B, etc. and then you could use `counts(j) = counts(j) + 1` to keep track.

(c) Put up a nice header and then loop through the array of counts and display the results (labeled, please). If you want to be fancy, only print the ones that have occur in the code array. Also, you could print both the count and the percentage.

There are two ways to create the corresponding characters. One is to create a string `Alpha = 'AB....Z';` and then `Alpha(i)` would produce the corresponding letter. The other is to use the `char` command with `char(64+i)` giving the appropriate letter. Play around and see how they work.

5. Write a script that prompts the user to enter a string. Then takes that string and converts it into a code array. Next, convert the code array back into a string and display it. Finally, from the code array, determine and display the frequency of the letter distributions.

6. Run your script and save the results in a diary file. For the input use:
   Now is the time for all good people to come to the aid of their country.

7. Submit your script, 3 functions and diary file.

8. As always, when you are done, please fill out the survey.