Matlab lecture 4: Loops (for, while) and Logical Control Flow (if, else)

Logical Control Flow

* If, else
  + If <logical statement>; commands; end
  + If <logical statement>; commands; else; commands; end
  + If <logical statement>; else if <logical statement>; end
  + Class exercise: Number guessing game. Write a program that picks a random integer, and gives you one guess, giving right/wrong feedback based on the guess.
    - **randi()**, **input()**,
    - Build on it: instead of right/wrong only feedback, have it also say if you were too high or low.
* While
  + While <logical statement>; commands; end
  + This loops continuously until the logical statement is false. Then, it continues.
  + Class exercise: have the number guessing game keep giving you guesses until you get it right.
  + Now: have it end when you are right, OR when 3 guesses have been made. If you lose, let it say Game over!
* For
  + for el = array; commands; end
  + For el = matrix; commands; end
  + This statement loops a set number of times, then continues with the rest of the script/function.
    - How many times? The number of columns in the array/matrix.
    - Inside the for loop, el is a single column of that matrix.
      * Each loop, el is the next column of that matrix.
      * The = sign then, has a slightly different meaning in a for statement, since it doesn't define the whole matrix, but rather a variable that *iterates* over that matrix.
  + Class exercise: do the same 3-guess game as before, but this time use a for loop instead of a while loop.
    - This time, the code will look much easier to read.

Efficient Matlab: “Vectorizing” your code

* + Class Exercise: make a new classical mean() function using a for loop over the elements. Compare to a vectorized version of the code.

New Data Structures: **Curly-Bracketed Cell Arrays** { }and **Structures**

* **Cell Arrays**
  + You make cell arrays by using the curly brackets, instead of the square brackets used for making matrix arrays. Cell arrays are the same as matrix arrays in Matlab, except that:
    - Each element of the cell array is completely independent of the other elements, so you can store different data types in each element.
      * ex) myCell = {‘a’, 1, ‘b’, 2}
    - Each element can contain any size data. i.e. You can store an entire word or matrix inside a single cell.
  + Indexing is the same as matrix arrays, except that you must choose either to reference using curly brackets or parentheses:
    - Curly Brackets: “give me what is inside this cell. Extract out its contents and give it to me as whatever data type is inside.”
    - Parentheses: “give me this cell, and keep it as a cell.”
* **Structures**
  + Structures are used to group different variables together and arrange them hierarchically (in a tree-like structure)
  + To place a variable in a structure, you simply add a period (.) after the name of the structure you want to make.
    - ex) data.spikes = mySpikeData
  + Structures can also be indexed into structure arrays. This is useful, for example, if you have many datasets that share the same data format.
    - ex) data(1).spikes for first day’s recording, and data(2).spikes for second day’s recording.

Navigating Filesystems:

* + **cd()**: changes current directory to the one specified. Can write the full path, or a relative path for the current one. ".." Means go up one level.
  + **mkdir()** makes a new folder.
  + **filelist= dir()** outputs a list of all the files, folders, and cd() options in the current directory in a structure. To get the name of each file, call filelist(ii).name, in which ii is index the order of the list of files (usually alphabetical).
    - This index starts from 3, so the first file in the list will be filelist(3).name
    - The fourth file, then, will be filelist(7).name.

Homework: Batch Analysis

**Warning: make a backup copy of the data folder for this homework before running your script, as there is possibility of accidentally overwriting the data while working on the homework.**

Data: The data directory contains 200 files, each containing 15 random tab-delimited numbers.

Assignment: Write a Matlab script that:

1) Makes a new folder called “processed\_data”

2) In the “processed\_data” folder, saves a text file that contains the **mean** of the numbers in each data file.

* The new file should be named something similar to the file it is taking data from, but not exact. For example: the mean of “dat1.txt” can be saved in “mean\_dat1.txt”.

3) Plots a histogram of all the **means** from the data (should look similar to below).

