Lecture 2 Outline: Matlab Functions and Scripts

- 20 minutes: go over previous week's homework, demonstrating all answers to the questions.

2. Matlab functions:
    - a function is a command that takes input and (optionally) returns some output.
        - A functions's form is: **[output] = functionName(input1,input2,etc)**
    - To use a function, Matlab needs to be able to find the function's corresponding file. It looks inside its **path** to see if any matching files are present.
        - To see Matlab's path and edit it, click on the **"set path"** icon in the ribbon.
            - Demonstration: Show that a function (created by us) is not available (produces an error when called), then add its path, then show it is available now.
        - Matlab has lots of functions included in its standard library (in collections called **toolboxes**), and so by default Matlab added its own toolbox folders to its path.
            - Because these functions are usually very high-quality (they correctly do what they say they will do), using Matlab is made easier by simply learning the names of the Matlab functions you want to use.
                - We will practice a few during the course, but you'll learn a lot more as you practice.
            - The best place to find a desired function is in the Matlab documentation. Type **doc** to display it.
            - To quickly find out how to use a function and learn what it does, type **help functionName** and a short summary with instructions will appear. For more info (and sometimes examples), type **doc functionName**
                - 15-minute Exercise: find out what the following functions do and how to use them using the help command: **mean()**, **linspace()**, **std()**, **sum()**, **plot()** etc.
            - The Mathworks website also has extensive documentation available, including "Matlab Central", a public forum where people post their own custom functions for others to download and use.
3. Matlab Scripts: Introduction
    - A script is a list of commands that you want to perform in a certain order (like a movie script, or a cookbook recipe). In Matlab and most programming languages, each step happens on a new line of a text file that has the **.m** extension.

- Matlab's editor is very nice for writing these scripts, because it:
  - uses colors to highlight different parts of the script (e.g. comments are green, closed parens are highlighted, for/end statements are blue), so you can easily read your script and understand it.
  - uses the tab button for auto-completion, to make it quicker to type your scripts and less likely to make spelling errors.
  - Displays errors and warnings in red and orange, to help show you where there might be problems. This helps you make less mistakes and quickly write code that works!
- To use a script, simply type the commands, in order, on their own seperate lines. To run it, make sure it is saved, then click the green run button (or press the F5 key)! Simple!
- Exercise (20 minutes): Make a script that does the second part of last week's homework (the jumping experiment).
  - Comment different sections of your script to separate them
  - Add a section to the script that finds the mean and standard deviation of each group's jumping height, and plot the mean of each group's jumping height in a bar graph. (Don't worry about making it look nice. That's next week's topic!)

4. Make your own Matlab functions:
   - Why write your own functions, when Matlab has so many nice ones? Functions allow you to:
     - Easily do the same task repeatedly, without copy-and-paste programming practices (this is a bad habit to get into, as it produces error-prone code that is long and unwieldy).
     - Combining many small functions together helps to abstract away the complexity of a task, letting you focus on what your are trying to accomplish, rather than every little detail of its implementation.
     - share code in a flexible, self-contained way.
   - A Matlab function is written the same way as a script, with three additions:
     - at the very top of the script, you write:
       - **function [output(s)] = functionName(input(s))**
     - Next, you write comments describing what your function does. This is optional, but very helpful, as it is what appears when you type **help functionName**.
     - Reminder: Functions can be used only if they are part of Matlab's path (so Matlab can see them). Remember to add the path of any functions you want to use! You can use the Set Path button to do this, and the **addpath()** function is a convenient function to do the same thing, which you can put at the beginning of a script to automate the process.
   - Demonstration (15-minutes): Make two simple functions as a class: a custom **MyMean()** function that produces the mean of two numbers, and a custom **myName()** function that displays a "Hello, <yourName>" output.

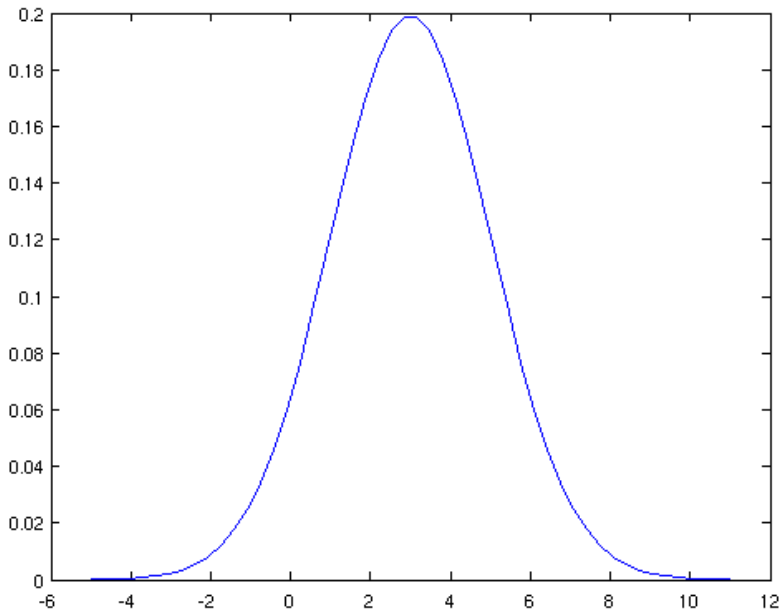
Lecture 2 Homework

Due Before 5 p.m. on Sunday before class.

-e-mail your m-file to delgrosso(dot)nick(at)gmail(punkt)com

Assignment: Write the Matlab function **probDensity(mean,std)**, a function which takes two inputs (a number for the mean, and a number for the standard deviation) and makes a figure out of them:

1. a line plot that displays a normal probability density function (the "bell curve") with the mean (a single number) and standard deviation (another single number) as inputs.
    - The data plotted should be between -4 and +4 standard deviations from the mean.
    - Useful functions: **linspace()**, **normpdf(), plot()**
        - Reminder: you can learn how to use these functions by simply typing **help functionName** into the Command Window.
    - ex) probDensity(3,2) should create a plot centered around 3, with the values -5 to 11 plotted on the x axis, and showing a plot of a bell curve with a standard deviation of 2 (shown below).



2. Write help comments for the help function when typing **help probDensity**, including your name to it to show who authored it.
3. Comment each line of your code to describe what it does, by putting the "%" sign in front of your comment in the line above the code.