Review: Week 1

1. You initialize a variable as follows:

   x = zeros(10, 20)

   a. What kind of data structure is x?
   b. What is the size of x?
   c. What does the zeros(a, b) function do?
   d. How can you inspect the contents of variable x?
   e. What happens if you add a semicolon to the end of the declaration?

2. You initialize an array as follows:

   y = randi(100, 1, 20)

   a. What does the randi(a, b, c) function do?
   b. How do x and y differ?
   c. Is y a row or column vector?
   d. Write a statement that:
      - Finds all elements of y that are greater than 50
      - Returns a logical vector with the *same size* as y
      - The logical vector has 1s where elements of y are greater than 10
      - The logical vector has 0s otherwise
   e. Using your result from d, return an array w that consists of elements of y that are greater than 50.
   f. What is the result of the following statement:

      z = y.^2

   g. What is a *.mat file? Save x and y together into one.
   h. Where is the *.mat file saved to if you don't specify a specific location?
   i. How do you load a *.mat file?
   j. What's the difference between a *.m and a *.mat file?

3. Scripts
   a. What is a script?
   b. How do you run only a highlighted segment of your script?
   c. What's the use of writing your analysis code in a script vs. directly in the Command Window?
   d. How do you write a comment in a script? (In other words, a line that will not actually be executed by MATLAB.)
   e. When you save a MATLAB script, what file extension does it have? For example, the file extension of a Word document is *.docx.
   f. Sometimes, certain lines of code will be underlined with orange or red. What does that mean?

Week 2: Choice and loop control flow statements

Control flow statements allow you to:

- Repeat a set of actions either:
    - A predetermined number of times, or
    - Until a certain condition is met
- Execute a certain set of actions *if* a condition is met

There are three main structures we'll talk about and are incredibly important in programming:

- For loops (execute a specific number of times)
- While loops (execute until a condition is met)
- If/else statements (execute certain actions based on conditions)

Intuition: control flows in real life

Assume you are a TA for a class with 50 students. ***For*** each student, you have a certain number of things you have to do, including marking their exam and entering the grade into your computer. So, you might think to yourself:

for each of the 50 students
> grade their exam
> enter the marks into a database

Now, assume that you *don't* know how many students there are, but you're given a stack of exams. Then instead, you want to repeat grading and entering marks ***while*** there are still papers in your stack. So, you might think to yourself:

while there is paper left in the stack to mark
> grade their exam
> enter the marks into a database
> check the number of papers left in the stack

You get an email from the professor letting you know that the stack of exams got mixed up: ***if*** students are in group 1, they should be marked by you, ***else*** students in group 2 should be marked by your fellow TA. So, using the while loop, you modify your process to separate out the students:

while there is paper left in the stack
if the student is in group 1
> grade their exam
> enter the marks into a database
else (they are in group 2)
> put aside for the other TA
> check the number of papers left in the stack

Writing out the structure of your control statement in plain English, without using any MATLAB syntax, is called "pseudocode" and can be useful to help you organize your logic before you begin implementing the flow.

Exercises

1. You have the following for loop:

```
for i = 1:1000
        disp(i + 20)
end
```

   a. What is the index variable in this for loop? (Hint: it's the variable that changes on each loop iteration.)
   b. For the first five loops, what is displayed in the Command Window?
      i = 1 →
      i = 2 →
      i = 3 →
      i = 4 →
      i = 5 →
   c. Erase the "1:" from the for loop statement. Explain the output of this code.

```
for i = 1000
        disp(i+20)
end
```

2. Now, instead of using the index directly in the result, we can also use it to get the values in an array. For example, load the cities dataset (load cities.mat). The variable "names" is a matrix of characters, with a different city name on each row.

   a. How many cities are in the list? Then, how many for loop iterations need to happen? Modify the for loop accordingly to match the number of cities.
   b. How would you modify the disp() statement in the for loop to print each city name?
   c. What would happen if you hadn't made the changes in question a? (i.e., if the for loop remained from 1 to 1000?)
   d. Instead of just displaying the city name, we now want to modify the loop so that, for each city, we take the first letter of the city name and store it in a variable called first_letter. Modify the for loop to achieve this. (Hint: you can index on either side of the equal sign!) Is the output a row or column vector?
   e. How would you ensure that the first_letter variable from d is a *column* vector?

3. In contrast to for loops, while loops are used when you don't know the exact number of repetitions that need to happen, but know that a certain condition should stop the code's execution.

   a. Implement the code from question 1 as a while loop. (Hint: what "condition" must be met for your while loop to stop? How do you keep track of that value?)
   b. What's the difference between the indexing variable in a for loop (i.e., i = 1:1000) and the indexing statement in your while loop?

4. Use an if/else statement to modify the code from question 3 such that something different is displayed if i is less than 500 versus if i is equal to or greater than 500.

5. Load the lsat dataset. This includes the LSAT and GPAs of 15 students in law school. We want to examine the relationship between these two measures.

   a. What do you hypothesize is the relationship between these scores?
   b. Come up with two ways to compare LSAT and GPA scores visually (i.e., with plots).
   c. What are the means, medians, and standard deviations of the LSAT and GPA scores?
   d. What is the correlation between LSAT and GPA scores? What is the significance of this relationship (i.e., p-value)?
   e. What is a NaN value? When might it be useful?
   f. Append a NaN to the end of the LSAT and GPA arrays and repeat the calculation of descriptive statistics from c. What happens to the results? What functions can you use instead to handle NaNs?

6. More descriptive statistics and visualization:

   a. What do the randi(imax, [sz1 sz2]) and normrnd(mu, sigma, [sz1 sz2])[1] functions do? Be sure to also explain what the parameters within the function do.
   b. Generate arrays with 5000 values with each of these functions and choose parameters that result in similar means between the two arrays. Use the histogram function to examine the distributions. What's the difference?
   c. Create a third array using the normrnd[1] function that has the same mean value, but a different standard deviation. Use the histogram plot to inspect the distributions. What's the difference?
   d. Find the range, minimum, and maximum of one of the arrays.
   e. Find the 5^{th} and 95^{th} percentile of one of the arrays.

7. Generate your own data and create an analysis that uses elements of what you learned today, including either a for or while loop, and an if/else statement.

[1] If you do not have the Statistics & Machine Learning Toolbox, replace the normrnd function with the following:

```
y = sigma.*randn(num_datapoints,1) + mu
```

Look into what the function randn(…) accomplishes, and how its output is affected by the sigma and mu terms.