Homwework 1

Table of Contents

by Simon Roy

Part 1:

Make a Github account using your @u.boisestate.edu email address. Then, using the Github Desktop app, clone the master branch of the GEOS397 project to your local directory. Make a new branch called GEOS397_Lastname, where you insert your last name.

I was able to complete this part. I have my Github account, the desktop app on my laptop, a clone of the master branch on my computer, and a branch with my last name attached. I actually made several of these when I was trying to figure out how it worked. I think I have them cleaned up, but if you are still seeing multiple branches, let me know and I'll try to figure out how to remove them.

Part 2:

In your new branch, make an new file in the HW1 directory called GEOS397_HW1_Lastname.m. Use a MATLAB script to write a summary of how you would go about ensuring that (if the clas had 10 students) you would partner with every other student for the 9 homework sets (you can write some equations if you want). (Hint: Use the publish tab on the top ribbon to help format the document.) Keep in mind that a constraint imposed on this problem is that no two students in the class can have repeat partners.

This document was created using the homework file specified, which is named as asked and in the folder that was reqested.

The solution that comes to mind first, and probably the one that you will get from the majority of the class, is borrowed from a speed dating problem that presents the same requirement: How to pair an even number of people, without repeating a pairing, such that everyone gets to meet. One solution, which I can't claim as my own, would be to sit the people across from each other, five on each side, and have them talk to (in our case, work with) the person across from them. Then one person would stay put (throughout the entire process) while the others rotate one diretion or the other until all people have met.

I'm sure it would be possible to implement this into code, but I'm not sure exactly how to do it.

Here are some other solutions that I've come up with, although I still don't have the skills to implement them in code on my own.

I thought the best way to go about it was to get some kind of representation of all the possible pairings. We can simply treat the students as numbers 1 through 10 and assign names later.

One idea is to generate a simple list of all possible combinations, then use a random number generator to make random pairs in sets of five that don't have any numbers in common. There would

have to be a structure determine if each set of five pairs was a valid choice. As each group of five is selected, those pairs could be removed from the list and added to another list labeled week n. This could work, but it may not converge. I tried selecting random numbers and eventually got to a point where I couln't get the five good pairs by selecting individually. We would have to check the five pairs at once, and it still may give us trouble.

Another idea I had was to represent the choices in a 10x10 matrix, with all the entries above the diagonal representing the possible choices (corresponding to the indices of the matrix). I thought this could lead somewhere, so I actually spent a good amount of time trying to get it to work, just for fun. I ended up getting my code to do %what I had in mind, which included building an upper triangular matrix, checking values, checking entire rows and columns for values, etc. In the end, like I said, I got it to do what I was trying to do, but it didn't give me the result I wanted. Then I remebered something my first programming teacher (Python) told me: *If you can't get your code to work, you probably don't understand the problem well enough.* I went back and studied the problem, and realized that I couldn't get what I wanted from the approach I was using. The problem was, in fact, a little more complicated than I thought. Still, I think this matrix representation of the possibilities has promise. If I get some free time (LOL) I might play with it some more.

Part 3: and Part 4

*In the same file, list all of the possible variable types in MATLAB that are covered in the **MATLAB style guide reading assignment**. Also, give a description of each type and list why this is a useful type of variable.*

Based on the reading MatlabStyle1p5.pdf, give an example variable name for each of the variable types you identified in Part 3. Then publish your MATLAB script as a pdf file; also commit your changes to your specific GIT branch; DO NOT publish though.

I combined these sections because it seemed to make sense to do so. I will discuss each type mentioned and give an example at the same time.

This was a little difficult to interpret, as there are many more types of data in MATLAB than were discussed in the style guide, but I will describe the variables that are discussed in the provided style guide, as well as mentioning a few other important types.

- **The first mention of Variables is under naming conventions. Typically it's best to give variables a meaningful name. The name should begin with a letter and have the first letter of each subsequent word in the name have a capitol letter for readability.**

Example: dataList, elevationTable

- **Commonly variables will be assigned a value, if it's a number it can be an integer, a single precision floating point number, or a double precision float. Each takes a different amount of memory and is appropriate for different uses.**

Example: integers: 1,2,3.. floating point: 3.233756383, the different types of floating point variables keep track of different numbers of decimal places.

- **The document also mentions that variables representing a number of objects, the prefix n is added.**

Example: nChoices, nAssignments.

- **This seems like a good place to mention strings, which are variables that are defined by characters of text.**

Example: 'myName','simon'

- **The document also implies the existence of data in matrix form, not only as numeric data, but also as strings, or other types of data. These all vary slightly, but mostly consist of data arranged in a grid, table, list or matrix that can then be manipulated with various tools in MatLab. I imagine the different specific types are created such that some forms, like numerical arrays, can preform operations efficiently, while other structures are built to handle more complicated data, but probably require more computing power and might not be compatible with some operations.**

Example: These could all be named in a similar fashion as the previous types, the difference is how they are stored in the computer and how they can be used.

- **The document mentions pluralization, and suggests using the suffix Array with variables that represent many objects.**

Example: point, pointArray

- **For clarity, an i should preceed varibles that will be iterated.**

Example: iFiles

- **There is also mention of boolean variables, or logic variables. These can be used to test conditions and control the flow of the program. It's often helpful to name them by what they mean.**

Example: isGreaterThan, ~isGreaterThan, true, false

- **The document goes on to talk about constants, which are basically variables that don't change, and are represented by all capitol letters.**

Example: NUMBER_PARTICLES

- **The last two that I will mention are structures and function handles. Structures are often complicated arrays of varying types of data. The style guide suggests using a capitol letter to begin the name of a structure. A function handle is used to call a function within the code. It will have a name (that should describe it use) and an arguement.**

Examples: (from guide) structure: Segment.length, function handle: myDerivative(.), areaCircle(.)

Comments

I feel like this isn't exactly what you were looking for in part 3 & 4, but I did my best to be thorough and complete. Sorry if my answer is way more (or less) than you wanted. I did my best to format this file in a readable way, but I may have over-done it there as well :) Thanks for letting me enroll in you class late, I look forward to lots of learning (and frustration, as is typical when learning to write code.)

Simon Roy

Published with MATLAB® R2015b