**Video #10: Working with Cell Arrays**

From early on in the semester, we've used cell arrays to store collections of strings, like some of my pet names here. We list the strings inside curly braces, and if we want to refer to the string at a particular location of the cell array, we can provide an index inside curly braces, such as myPets at index 1, which is cleo, my current siamese cat Cleopatra. The for loop here prints all the strings in the cell array using an index from 1 up to the length of myPets.

This is a common use of cell arrays, but their real power is that they allow us to store multiple types of data in one place. Here I create myCell that has 4 different things - the string Ellen, the number pi, a vector and a matrix (note the semi-colon that makes it a 2x2 matrix). This shows how MATLAB describes its value. We learned that structures in MATLAB also allow us to store things of different types together in one place, but in the case of a structure, you refer to different fields by their name. For a cell array, the different things are placed in the cell array in a particular order, and we refer to them with an index, as we did earlier with our cell array of strings. There's a built-in function called celldisp that displays the contents of a cell array in a different way, referring to the locations with their index. It tells us that myCell at index 1 is Ellen, at index 2 is the number 3.1416, index 3 is this vector of four values, and at index 4, the 2x2 matrix. When I created myCell, I listed all the contents in one assignment statement. But you can also create an initial empty cell array of a particular size using a function called cell that takes a number of rows and columns that you want the cell array to have, and then you can later add content to each individual cell using indices.

You may be wondering if there's a way to refer to the inner contents of the vector or matrix that are stored in the cell array, or maybe the characters of the string. Here I've written the contents more spread out, and we know so far that we can refer to each element as myCell with one index in curly braces. We can think of myCell{1} as the name of this string, and we know we can refer to characters within a string using an index in parentheses, so we can just add a second index like (4) to refer to the 4th character in the string, the lower case e. In the third location of the cell array, we have a vector, and we can think of myCell{3} as the name of the vector, and then refer to the content of a particular location of the vector by adding another index inside parentheses - this expression refers to the second location of the vector, which has the number 5. Finally the last thing in the cell array is a matrix, and we refer to a location inside a matrix using two indices specifying a row and column number. myCell{4} is the name of the matrix and we can add a pair of indices afterwards inside parentheses, in this case, the second row and first column, which would give us the number 3.

The next topic we'll learn about in this class is how to access information for our programs that's stored in external files - this may be information that's numerical data or text, or some combination of the two. When MATLAB loads this information into the workspace of our program, it will automatically store the information in a cell array. If the information contains text, it will be stored in a cell array that has another cell array nested inside it. To give you a taste of this idea, we're going to explore one last example here that involves working with

information about mountains, and we'll look at the actual code file in the editor, so we can also run the code from there.

The data is stored in a cell array named mountains, and take a close look at the curly braces to see the overall structure of mountains. We have one set of curly braces surrounding all the information <show starting/ending braces>. Inside there are four things - an inner cell array with the names of 11 different mountains <show start/end>. Then there's another cell array of 11 strings that are the names of the mountain ranges that these mountains are part of (e.g. Everest is in the Himalayas, K2 is in the Karakoram range, and so on). There's a third cell array of 11 strings that are the names of countries where these mountains are located, and sometimes there's more than one country (Everest is in Nepal and China, for example). The last thing in the mountains cell array is a vector of numbers that are the heights of the mountains in feet. Let's then see what we're doing with this information. We ask the user to enter the name of a mountain, and then see if we can locate that mountain in the names that are listed in the first cell array. We can compare a single string, like mount, with a whole cell array of strings, like mountains at index 1, all at once, and here I'm using new function strcmpi which stands for string compare independent of case, so capitalization doesn't matter for this comparison. If there's a match, find will get the index. If the name is not found, then the find function returns an empty vector, and we can check that with a function called isempty. If this is true, we'll tell the user their mountain wasn't found, otherwise we'll use that index to print out all the information about their mountain. But take a close look at the indices - the name is in mountains{1} and it's a cell array so the second index needs to be written inside curly braces. Similarly, the range and location are also in cell arrays, so index again is inside curly braces. But the last piece of information is the height, which is stored inside a vector of numbers, so the index here needs to be placed in parentheses and we need to add num2str. So let me run the script. I'll enter k2 so I don't need to worry about spelling, and there's the information about K2. Let's run it again and enter Ellen, but that mountain was not found.

So we can store different kinds of things within cell arrays, including other cell arrays, and we use indices to refer to the different items stored in a cell array. And cell arrays are used a lot in MATLAB applications. You see them in the new Apps you're creating - they're used to store the items in drop-down menus and lists, for example. And you'll see them again when you learn about text files.