- What did we talk about last class?
- Do you have any questions about the reading?
- Using end for indexing.
Last time we looked at the for loop. We also have the following:

- While loop
- If statements – includes if, else, end, and elseif.
- Switch-case – allows cases with multiple expressions and otherwise clause.
- Try-catch blocks for error handling. Variable lasterr gives information about the last error.

In general you can get away with only using loops and the if conditional. Switch is occasionally helpful and the try-catch is nice in situations where something might go wrong.
Examples of Control Flow

- Write a few lines of code that will do Serpinski's triangle and put each new point in a single Nx2 array. Put 5000 points into it.
- You can plot that array with the following:
  - `plot(data(:,1),data(:,2),'.')`
  - Now write a loop that will do a Mandelbrot check for a single point. Have it loop until $z_n$ has a magnitude greater than 2 or you get through 100 iterations.
Matlab functions are a bit different from what you are used to in other languages. They can take a variable number of arguments and return a variable number of arguments.

To return a value, we set a variable with the name specified on the first line of the function to the value we want to return. That is what will be returned when the function terminates.

You can also have local functions or nested functions. When either of these is used the main function must be terminated with end.

Function handles are also discussed in the book and they are worth noting.
We have our Matlab code to do the iteration for a single pixel of the mandelbrot set. Now make an m-file and put that code into the m-file as a function that takes a point and returns how many iterations it went.

Let's try to write code that will plot up a full Mandelbrot set for us.
Arrays in Matlab can have more than 2 dimensions to them.

We aren't going to deal with these much in this course and your book doesn't really deal with them outside of this one chapter either.
By default, everything in Matlab is a double. However, you can force things to be other types.

Integer types can be signed or unsigned with 8, 16, 32, or 64 bits. When you construct an array with one of the construction methods, you can pass in a string giving the integer type. The class function lets you see the type of something that you have.

You can also do a casting like operation to make something a particular type.

If you want, there are also single precision floating point numbers.
You make these with {} or the cell function. They don't have to be rectangular and they can hold any data. Each cell can hold a different type of data.

We can index into cell arrays with {} as well. If you index a cell array with () you get a cell containing data. If you use {} you get the data that was in the cell.

Multiple elements can't normally be pulled out with content addressing unless you put them into multiple variables with a comma separated list.

Anything that produces multiple cells will be turned into a comma separated list. This can be difficult to get your brain around.
You can use the dot notation to put fields into a variable to make a structure. Unlike normal imperative languages, the format of the structure isn't predefined.

Matlab deals with arrays of structures just like numeric arrays.

The struct function can build arrays of structures from existing cell arrays.

You can pull out all the values of certain field with 'dynamic addressing'.
Like most other languages, Matlab does give you the ability to use strings, though that isn't a real strength.

A Matlab string is simply a row array of characters.

A downside of this is that an array with multiple strings must have all the strings be the same length. The char function can help with that.

You can also convert from numbers to strings and back with str2num and num2str.

Matlab also has fprintf and sprintf functions that work much like the C functions.

Similarly, sscanf will pull numbers out of strings.

eval and evalc let you process a string like it were a Matlab function.
■ Remember to submit assignment #2 today.