public with sharing class WeekTwoHomework {

public static void introToConditionals() {

/\*Comparison and logical operators are an important tool and one that we'll dive into much deeper with Conditional Statements next week.

For now, let's just get familar with some of the syntax.\*/

//First, let's get some variables that we can work with

//A single equals sign such as =, signals that we're assigning a value to a variable

Integer i = 4;

Integer j = 6;

Integer k = 2;

//Comparison operators let us do just that...Compare!

//Let's declare a boolean variable to hold the result of our comparsion

Boolean isBigger;

isBigger = i > k; //the comparison operater here is the greater than symbol: >

//Ok, so what we just did is compare the values of i and k, which will have a true/false result and assigned it to the boolean variable isBigger

//since i is in fact greater than k (4>2), the boolean value is true. If you run this method in Execute anonymous you can see the result

System.debug('isBigger: ' + isBigger);

//We will use these more next week when we discuss if/then Conditional constructs and control statements.

//You can find a complete list of operators here: https://developer.salesforce.com/docs/atlas.en-us.198.0.apexcode.meta/apexcode/langCon\_apex\_expressions\_operators\_understanding.htm?search\_text=ternary

}

public static void conditionalsExercise() {

Boolean result;

// 1. Write a comparison statement that evauates to false and assign the result to our result variable

//Don't forget to maintain indentation!

// 2. Declare another boolean variable and write a comparison that evaluates to true, assign it to your variable

}

public static void introToLists() {

//As you know from your reading, lists are a collection of data of the same type (integers, strings, decimals etc.)

//In Apex, a list can only contain data of the same type

//Let's declare a list of strings

List<String> listOfStrings = new List<String>();

//great, we have a list declared, but we haven't yet put anything in it. Before we do, let's look at another way to declare a list

String[] anotherListOfStrings = new String[]{};

//The above works just the same, it's simply a matter of preference. You will see the first version syntax more freqently in examples and sample code

//Lets add some values to our list using the .add() method. Salesforce has a number of built-in methods that you can use on lists

//including .add() and also .size() to get the size of a list. You use .add() as shown below:

anotherListOfStrings.add('Item1');

anotherListOfStrings.add('Item2');

anotherListOfStrings.add('Item3');

//Now our list anotherListOfStrings has 3 items in it. we can validate that using System.debug

System.debug('size of anotherListOfStrings: ' + anotherListOfStrings.size()); //this will print out 3 in the debug log.

//You can find all the built-in list methods here: https://developer.salesforce.com/docs/atlas.en-us.198.0.apexcode.meta/apexcode/apex\_methods\_system\_list.htm#apex\_System\_List\_methods

//Can we declare a list and add values at the same time? Of course! Let's declare a list of integers and add 1, 2 & 3 to the list

List<Integer> listOfIntegers = new List<Integer>{ 1, 2, 3 }; //Everything in the curly braces, separated by commas, is added to the list

System.debug('Contents of listOfIntegers: ' + listOfIntegers);

/\*Let's talk about indexing. Lists are ordered and you can refer to the entries in a list by using their index

The first item in a list or other collection is in the 0th position, the next is in position 1, then 2 etc.

Let's use a list of first names to demonstrate.

\*/

List<String> firstNames = new List<String> { 'Angelica', 'Mario', 'Luis' };

//Because Angelica is in the first position, that entry has an index of 0

System.debug('Value of 0 Index:' + firstNames[0]);

/\*if you open up the execute anonymous option in the developer console, and run this method by typing in: WeekTwoHomework.introToLists();

the debug log will show celery. The syntax used to reference by index number is as you see above,

the name of the lists followed by square brackets and the number of the position you want to reference

\*/

/\*Careful! if you attempt to access a value that does not exist, you will get an error.

And it's still ok! Errors are a common and universal part of coding. You might as well get used to it, so let's throw an error

Because the list below only has 3 entries, you can only directly reference the indexes at 0, 1, 2

When you try and get the value at the index 3 or higher, there is nothing there, and you get an error

paste the code below into an execute anonymous window, uncomment out the line that declares a new string and check out the error you get

\*/

List<String> cities = new List<String>{'Chicago', 'Los Angeles', 'Portland'};

//Uncomment out the line below to try throwing the error

//String s = cities[3];

}

public static void listsExercise() {

//You do this part!

//1. Declare a list of Strings called myStringList

//anotherListOfStrings.add('gelatin');

//anotherListOfStrings.add('ice cream');

//anotherListOfStrings.add('yogurt');

//2. Add three string values to your list and print it out in the debug log

//3. Print out the third value in your list of strings using list indexing.

//4. Declare a list of integers and add the values 1,2,3,4,5 all in one line

//'Gelatin' = 1

//'Ice cream' = 2

//'Yogurt' = 3