Hello class!

First, I will go through the homework assignment, then I will do a play by play of everything we went through in the first class. Don't worry if it hasn't completely sunk in yet, there will be many opportunities to go through all the basics in the upcoming classes.

Homework:
On slides 26 and 27 (the fourth and third to last slides), you will see some code that is supposed to go into the application/views/bloggers/list.php and application/view/posts/list.php.

The first line is going to be a little magical for now until we learn about controllers and models. Basically what is happening is the variable $data has the value of an ARRAY of data pulled from the posts table in the database.

The foreach statement should loop through all the posts in the database and display the title, text, date_created, and username of each post. You can use the code as is (although I recommend writing it out yourself to really think through how it is working.

This is the endpoint of the Model->Controller->View workflow. So your job is just to get data showing up. The html and css are there for it to look like my example. If you want to change the layout, feel free to change the code, or to go into the public/css/styles.css file and play with my styles.

PHP 101
What is PHP?
PHP stands for PHP Hypertext Protocol. It was designed to work with HTML and CSS and make web pages more dynamic. It is still the most popular language for dynamic sites on the web, used by Facebook, Amazon, Wordpress and Drupal.

Variables
Variables exist in every programming language. They have a name and represent some kind of data, whether words (strings), numbers, arrays(see explanation below), or other objects. In PHP, you can tell something is a variable by $.

```
$bananas = 3;
```
Stands for "I am a variable, my name is bananas, my value is 3". The semicolon is like a period on the end of a sentence.

```
$name = "Izzy";
```
Stands for "I am a variable, my name is name, my value is the word Izzy." The quotation marks mean that the value is a word or set of words and not a number. So "3" and 3 are different values to PHP.

Concatenation
Concatenation is a way of putting words together with other words. You 'concatenate' in PHP with a period (.).

```
$name = "Izzy";
$greeting = "Hi";
$sentence = $greeting . " " . $name;
```

The empty space between $greeting and $name is because concatenation will put everything right up next to the next word, so unless you explicitly add a space, PHP will squish the two sets of words up next to each other.

Math
Just like you can do math to numbers, you can do math to variables whose values are numbers.

```
$bananas = 3;
$oranges = 4;
$fruit = $bananas + $oranges;
```

You can do all the basics addition(+), subtraction (-), multiplication(*), division(/). Plus some special things, like finding the remainder(%), adding one(++), subtracting one(--).

Conditional statements
Some of the power of coding languages is the ability to make the code do different things in different cases. A super common example of this is make words plural or singular depending on the number. We do this all the time when speaking, because we instinctively know when to say "inch" and when to say "inches", since we learned the difference as kids. However, with computers, you have to tell them.

```
$length = 5;
if($length ==1){
     echo $length . " inch";
}
else{
     echo $length . "inches";
}
```

In this example, $length is NOT 1, so the else statement takes over. However, if $length were 1, the code inside the { } brackets of the if part of the statement would come go.

You can do a few kinds of comparisons. Equals?(==) Greater than?(>) Less than?(<) Not equal to?(!=). Some times you need to check two things to make code go. In that case you would use AND (&&) to check if two things were true or you would use OR (||) to check if either one thing or another were true.

Conditional Loops
Sometimes you want to check if a condition is true more than once. In that case, you have a few options. A while loop, a for loop and a foreach loop. I will explain foreach loops with arrays below.

```
$i = 0;
while($i<10){
     echo $i;
     $i++;
}
```
This checks if $i is less than 10, echos out $i and then adds 1 to $i. It then checks again (is 1 <10?), and again (is 2<10?). etc. until it gets to (is 10<10?) which is NO, so it stops looping. The danger of while loops is that if we forgot $i++;, the code would check (is 0<10?) for LITERALLY forever…or until your computer crashed.

A safer way of doing the while loop is the for loop
```
for($i=0; $i<10; $i++){
     echo $i;
}
```
The for loop is just like the while loop, but all the conditions are in the first line and if you forget the $i++, the code won't work at all.

Arrays
Arrays are special variables that can hold multiple kinds of data through a mechanism called an "index". An index can either be a number or a name. If the data your array is holding has a natural order (like the days of the week), you will usually use a number. If however, the data has a meaning, like a color, it will usually have a name. See below:
```
$day[0] = "Sunday";
$day[1] = "Monday";
$day[2] = "Tuesday";
```
…
(Did I mention nerds are weird and like to start counting at 0?
```
$person['name'] = "Izzy";
$person['pet'] = "Winifred the Rabbit";
$person['favorite_color'] = "orange";
```
You can also have arrays WITHIN arrays!
```
$teacherA = array();
$teacherA['name'] = "Izzy";
$teacherA['email'] = "izzy@girldevelopit.com";

$teacherB = array();
$teacherB['name'] = "Leo";
$teacherB['email'] = "leo@girldevelopit.com";

$teachers = array();
$teachers[0] = $teacherA;
$teachers[1] = $teacherB;
```

So, if you wanted to know the SECOND teacher's name, you would write:

```
echo $teachers[1]['name'];
```

Reusing code

One thing you should know about programmers is that we are a smartly lazy group of people. If we don't have to repeat ourselves, we don't really want to. So, we often put code into "functions" that can be called anywhere in our code. We write the function once and use it over and over again wherever we need to. Things like conversion formulas, getting data from a database, formatting code, anything, really, that we would do more than once in our website. We will see a lot more functions in week 2.

Including code

In order to use the code that we write in functions in other files, we "include" the files into other files. It is a way of allowing files to recognize each other. There are three ways of doing so:
include – this just adds the code
require – this adds the code and breaks if it can't find the file you asked for
require_once – this adds the code ONLY if it hasn't been added somewhere else.

MVC 101

MVC stands for Model View Controller. It is a way of breaking up your code into logical chunks by functionality. Many, many, many people use this. Almost all websites that have been built since 2006 use this format, and many mobile apps use the format too!

Model

The model part of MVC defines the data structure. If your website is a library catalog, its model would define what data you can use to describe a book (title, author, subject, ISBN, etc.). It would probably also define what an author is (first name, last name, birth date, death date, etc.). And so on until everything you need for the data that makes the website useful is defined.

View

Views are usually the easiest to start with. They are simply the code that makes the webpage that people see. The stuff that is translated to HTML, any thing that a person visiting your website can see, click on, or play with will be in a view.

Controller

Controllers are the glue. The get the data (structured in the model), do any manipulation, like math, combining data from two different tables in the database, or any other behind the scenes work and sends it to the view.

For you, I have already built the main view in index.php. It calls two other views (blogger/list.php and posts/list.php) that you will work on for class1 homework described above.

Phew! That was long. Hope it helps!
Izzy