1. Data Structures and Algorithms
   a. One of the most important things behind medium-sized, complex programs is data management.
   b. This course introduces the most common data structures underlying good software design.
   c. Data structure: any data representation and its associated operations. Usually, the term refers to data structures that store multiple data items, such as a list of multiple floats.
   d. What data structures have you used? (lists, stacks).
   e. Efficiency matters. You want insertion, look-up, and deletion to be relatively fast. (Time, space trade-offs)

2. Logistics
   a. Stephanie teaches 9am lecture, Zadia teaches 10am lecture
   b. Caitrin teaches Mon labs, Zadia teaches Tues lab.
   c. Life in a week of cs231
      i. Monday: Correct quiz & Lecture. Attend lab on Mon or Tues. Project assigned in lab. Previous project due Monday night at midnight.
      ii. Wednesday: Lecture & assign Homework
      iii. Friday: Go over homework & give inclass Quiz. Start new lecture if possible
   d. We will do our best to post notes before lecture, but make no promises.

3. Java
   a. There are several pages on comparing Java and Python available on the course web site. I'm giving them to you mainly for reference in case you forget the Java way.
   b. The best way to understand Java is to jump right in, so we'll do so.
   c. Here are 4 of the most important things you need to understand about Java that distinguishes it from Python:
      i. Java has different syntax. No big deal. You'll be surprised how easily you'll be able to adapt to the new syntax.
      ii. Java is compiled not interpreted and so running a program is a 2-step process (Java is a compiled language. The code is read by the java compiler (javac) and converted into Java bytecode, which is stored in a .class file. To execute the code, you run the java virtual machine (java) and tell it the name of the class whose main method you want to execute. Java bytecode is not code that is directly executable by a CPU (though some native Java computers have been built). Instead, the bytecode runs
on a Java Virtual Machine [JVM], which is a program that runs on your computer and translates the Java bytecode into actual machine instructions).

iii. Java is *statically typed*, which means that variables have types, not just data. Variables can only store data of the declared type of the variable.

iv. Java is an OO language, meaning virtually everything in it consists of classes, objects and methods. **What is a class, object, and method?**

d. A typical program creates some objects and has them execute their methods. Note there are a few non-objects (called primitives). These are integer, character, boolean, and floating point values. We'll see them a lot but you should mostly focus on the objects and classes.

e. To create objects to do things for you, you first have to declare a class. What is a class? [a template indicating the data and methods that can be executed by that object]

f. Java has thousands of built-in classes that you can use (API) - **Java application programming interface** is a list of all classes that are part of the Java development kit (JDK). It includes all Java packages, classes, and interfaces, along with their methods, fields, and constructors. These prewritten classes provide a tremendous amount of functionality to a programmer.

g. But you can also create new classes of your own. In fact, creating your own new classes is almost all the code that you'll ever write in Java.

h. Overall format: Java source code is stored in a simple text file with suffix ".java". You give the file the same name as the class declared in the fie. If you declare a class named Hello, then the file should have the name Hello.java.

i. First program (The code will be posted online) - Let us take a simple program—it is probably the one we all create when learning a new programming language! Almost all you'll ever do in a Java program is declare new classes and when you run the program, you are just running a starting "main" method, which might call other methods.

```java
//Hello.java
public class Hello {
    public static void main(String[ ] args) {
        System.out.println("Hello World");
    }
}
```
j. Single-line comments begin with // characters. Multi-line comments are enclosed within /* ....*/

k. Java is case-sensitive; all keywords belonging to the language itself are spelled as in the examples using lower-case characters.

l. The import java.io.* line tells the compiler to use methods from the Java input/output package, or class library, to provide output.

m. You declare a class by specifying the class keyword followed by a non-reserved identifier that names it. By convention, the first letter of a class's name is uppercased and subsequent characters are lowercased (for example, Employee). If a name consists of multiple words, the first letter of each word is uppercased (such as SavingsAccount). This naming convention is called CamelCasing. They can be any combo of letters, digits, underscores, and dollar signs that doesn't begin with a digit and that isn't a keyword. It is recommended that you don’t use the dollar sign.

n. The start of a class is the public/private designation followed by the name of the class.

o. “public” is optional but you should include it by default. It means the class is usable by every other part of the program. The other accessibility options for classes are useful only in large projects.

p. The {...} braces enclose a block of code—the outer ones being the class definition, and the inner ones the block of the main function.

q. Whitespace doesn’t matter to Java (but it will to your readers!)

r. Java programs must have one named main( )

s. The main has no 'return' value (void), and it is public (accessible).

t. The keyword static applied to a function or variable, states it must remain in memory until the program ends

u. The term in parentheses represents arguments that may be passed to main when it runs; normally they may be omitted. Strings args[]: communication with the command line
v. `System.out.println()` outputs data to the output stream. Here, it sends the character string in quotes to the screen, followed by a 'newline' character (`System.out.print( )`) without In does not send an end-of-line character.

w. Each executable statement is terminated with a semicolon. (an error if you don’t have it)

x. Compile and run
   i. Make sure you have JDK installed on your computer to be able to compile Java
   ii. In the Terminal, we navigate to the appropriate directory.
   iii. We compile: `javac Hello.java`
   iv. We run: `java Hello`