01-0: **Syllabus**

- Office Hours
- Course Text
- Test Dates & Testing Policies
  - Check dates now!
- Grading Policies
- Coding Standards

01-1: **How to Succeed**

- Come to class. Pay attention. Ask questions.

01-2: **How to Succeed**

- Come to class. Pay attention. Ask questions.
  - A question as vague as "I don't get it" is perfectly acceptable.
  - If you're confused, there's a good chance someone else is confused as well.

01-3: **How to Succeed**

- Come to class. Pay attention. Ask questions.
  - A question as vague as "I don't get it" is perfectly acceptable.
  - If you're confused, there's a good chance someone else is confused as well.
- Come by my office
  - I am *very* available to students.

01-4: **How to Succeed**

- Come to class. Pay attention. Ask questions.
  - A question as vague as "I don't get it" is perfectly acceptable.
  - If you're confused, there's a good chance someone else is confused as well.
- Come by my office
  - I am *very* available to students.
- Start the Labs and Projects early

01-5: **Class Format**

- First part of class will *usually* be lecture
- Second part of class with *usually* be a lab
- Might switch off between lecture and lab several times during a single class period

01-6: **Labs vs. Projects**

- Labs:
  - Typically done in class
    - may need to spend some time outside class to finish labs
  - Lots and lots of help from TA and myself – often do parts of the labs together as a clas
- Projects
  - Typically done outside of class
  - Do more of the work on your own (but TA and I will still help!)

01-7: **Expectations**

- Come to class
- Pay attention
  - No email, twitter, facebook, etc
- Spend 10 hours / week *outside of clas* working on labs, projects, and reading assignments
- All submitted code must be your own original work
  - Absolutely no copying of code!

01-8: **Python vs. Java**

- Last semester you programming in Python, this semester we will be using Java
- Similar in many ways – almost everything that you learned last semester will transfer over
- How you approach Python and Java programming is quite different

01-9: **Warning, Danger Ahead**

- Java is much more verbose than Python
  - Occasionally have to put off some explainations for later – but will get to everything before the end of the semester!
- Java and Python are similar in many ways – almost everything that you learned last semester will transfer over
- How you approach Python and Java programming is quite different

01-10: **Similarities**

- Start with similarities between languages
- Show some simple constructs in Python, Java equivalents
- Go on to the real meat of the differences

01-11: **Variables**

- Both Java and Python both have variables
- Python is pretty lax
  - Use a variable, and the system figures out what you want.

    - Can store pretty much any value in any variable
- Java is much more strict
    - Must declare a variable before you use it, giving the type of the variable
    - Can only store values of the declared type in that variable

01-12: **Variables**

- Python

```
principal = 3000
```

- Java

```
int principal;
principal = 3000;
```

    - Declare the variabe to be of type integer
    - Now can only store integers in variable principal (strongly typed)
    - Semicolins are like end-of-lines in Python (more on formatting in a bit)

01-13: **Variables**

```
int principal = 3000;
```

- We can both declare a variable, and give it an inital value, at the same time
- Looks very similar to Python – but principal can now *only* hold integer values (not strings, lists, etc)

01-14: **Conditionals**

- Python

```
if principal > 5000:
    print "you're rich"
```

- Java

```
if (principal > 5000)
{
    System.out.println("you're rich");
}
```

01-15: **Conditionals**

```
if (principal > 5000)
{
    print "you're rich";
}
```

- () required around test
- whitespace (including end of lines!) is optional

- Java uses ; for end of lines, and { } to group code blocks
- For if statements with one line, { } is optional (for compiler, not for this class!)

01-16: **Iteration**

- Python

```
number = 0
while number < 5
    print number
    number = number + 1
```

- Java

```
int number = 0;
while (number < 5)
{
   System.out.println(number);
   number++;
}
```

01-17: **Iteration**

```
int number = 0;
while (number < 5)
{
   System.out.println(number);
   number++;
}

for (int number = 0; number < 5; number++)
{
   System.out.println(number);
}
```

01-18: **Formatting**

- Python
  - Denote end of statements with end-of-line character
  - Block grouping using indenting / tabs
- Java
  - Denote end of statements with semicolons
  - Block grouping using curly braces { and }
  - Whitespace (spaces, tabs, end-of-lines) are completely ignored

01-19: **Formatting**

- Compiles and runs just fine:

```
int number = 0;
while (number < 5)
{
   System.out.println(number); number++;  }
```

01-20: **Formatting**

- Compiles and runs, but doesn't do what you want ...

```
int number = 0;
while (number < 5)
   System.out.println(number);
   number++;
```

01-21: **Objects!**

- Python is (typically) a functional language
  - Python code is a collection of funtions
  - Can create / use objects in Python, not required
  - "Verb based"
- Java is an Object Oriented langage
  - Java code is a collection of objects
  - *Must* use objects in Java
  - "Noun Based"

01-22: **Objects?**

- What is an object?
  - Collection of data and functions
- Think of an old-fashioned calculator, that allows you to store and recal numbers
- Similar to a Java object: Store data, do calclations on that data

01-23: **Designing a Program**

- Java programs are collections of classes
- A class is *NOT AN OBJECT*. A class is a template that allows you to create objects
- From a single class, you can create multiple objects
  - We could create a whole fleet of calculator objects, each of which has its own store/recall data

01-24: **Classes**

- Java Classes contain
  - *data*, usually called *instance variables*
  - *code*, usually called *methods*
  - Special method, called a *constructor*, which is invoked when objects of this class are created

01-25: **Classes**

```
// Filename must match class name
// For instance, this class would need to be saved as
//  NameOfClass.java
public class NameOfClass
{
   // Data Members (also called instance variables)
   //  * ''private'' means that only methods defined in
   //     this class can see/modify sampleVariable
   private int sampleVariable;

   // Constructor
   //  * Same name as the name of the class
   //  * May take parameters, though this is not required
   //  * Must be public
   public NameOfClass()
   {

   }
}
```

01-26: **Classes**

- "Name" class from website
- public / private modifiers
- Constructor

01-27: **Creating Objects**

- Creating a .class file does not create any objects
- Just a template for creating objects
- Objects need to be created with a call to "new"

```
NameOfClass silly = new NameOfClass();
Name name1 = new Name("John", "Adams");
Name name2 = new Name("Abraham", "Lincoln");
```

01-28: **Static**

- Classes are templates, not objects
- Need to create a new object using a class method before we can use it
- Chicken and egg problem – how do we create our first object?
- Static methods are special: One per class instead of one per object
- "main" is a special case function: entry point for the start of the code

01-29: **Driver Class**

```
public class Driver
{
    public static void main(String args[])
    {
        // Main Program
        //   Typically create one or more objects
        //   Call methods on these objects
        //   The "real work" is done in the classes/objects
        Name name1 = new Name("John", "Adams");
        Name name2 = new Name("Abraham", "Lincoln");
    }
}
```

01-30: **Methods**

- Methods are like functions in Python, with a few key differences

- Need to declare the return type, and type of all parameters
- Methods are associated with a class / object
  - We need to call method *from a created object*
  - We can access object data from within method, using "this"

01-31: **Methods**

```
public static void main(String argss[])
{
     Name n1 = new Name("John", "Smith");
     System.out.println(n1.getFirst());
     n1.setFirst("Adam");
     System.out.println(n1.getFirst());
}
```

01-32: **Methods**

```
public static void main(String argss[])
{
     Adder addr = new Adder();
     int value = addr.add(5,7);
     System.out.println(value);
}
```

01-33: **Project Composition**

- For this class, projects / labs will consist of:
  - Collection of one or more classes (each in a separate file)
  - Driver class, which contains a (small!) main static method. This method will create one or more objects, and call their methods
- So for each executable program in this class, you will have at least two different files