Classes and Constructors
A class in Java looks pretty much like a class in Python, except that the instance variables are defined outside any method. For example, we might have
public class Person {
    private String name;
    private int age;

    public Person(String who, int old) {
        name = who;
        age = old;
    }

    public Person(String who) {
        this(who, 0);
    }
}
// Here are some methods:
  public String getName() {
      return name;
  }
  public void setName(String who) {
      name = who;
  }
  public int getAge() {
      return age;
  }
  public void setAge(int old) {
      age = old;
  }
  public void birthday() {
      age += 1;
  }
  public toString() {
      return name;
  }
// Here is a main method:

class Person {
    String name;
    int age;

    Person(String name) {
        this.name = name;
    }

    public void setAge(int age) {
        this.age = age;
    }

    public int getAge() {
        return age;
    }

    public void birthday() {
        System.out.println(name + " birthday!");
    }
}

public static void main(String[] args) {
    Person x = new Person("bob");
x.setAge(68);
x.birthday();
    System.out.println(x.getAge());
}
A constructor for a class has an access attribute (usually *public*) but no return type, not even *void*. As in Python and any other object-oriented language, the job of a constructor is to initialize the instance variables of the class.

You can have multiple constructors for a class as long as they have different arguments.
Here is a subclass of student. Note that the subclass extends the parent class.

```java
public class Student extends Person {
    String college;

    public Student(String who) {
        super(who); // calls the Person constructor
        college = "Oberlin";
        setAge(18);
    }
}
```
Here are some more methods of class Student:

```java
public double getCollege() {
    return college;
}
```

```java
public void setCollege(String c) {
    college = c;
}
```

These are methods of class Student but not class Person. On the other hand, all methods of class Person are also methods of class Student.
Here is a method of class Student that overrides a method of the same name from class Person:

```java
String toString() {
    return String.format("%s is a student at %s", name, college);
}
```