Building a an extensible, Python-like list, in Java.

1. Questions?

2. A quick note about the type heirarchy in Java.
   (a) Every new type of object is a subtype of some older type (its supertype).
   (b) The ultimate supertype of every object in Java is an Object.
   (c) Type extension tells us that if a piece of code performs a calculation based on a specific type, say an Automobile, any object of a more specific subtype, like Tesla will work in the calculation as well.
   (d) It's always legal to assign a subtype value to a supertype variable:
      ```java
      Automobile a = new Tesla();
      ```
   (e) The other way is not advisable. Not all Automobiles have a charge() method, so the assignment of a supertype (Automobile) to a subtype variable (a Tesla) is generally illegal.
   (f) On occasion, we know the assignment is legal. We can inform Java of this fact by specifying or casting the object's known type. Consider the following:
      ```java
      Automobile a = new Tesla();
      a.driveTo("Toronto");
      Tesla t = (Tesla)a; // this is a cast: we *know* a is a Tesla
      t.charge();
      ```

3. Implementing a List-like class.
   (a) Based on arrays of Objects. We allocate arrays with the following type of notation:
      ```java
      Object a[] = new Object[n];
      ```
      Where n is the number of cells required. Note that this allocates an array of n Object references; there is no need to think about how they are constructed. Notice that an array is allocated with a specific length. We can determine the length of the array with the length instance variable. The length cannot be changed. That's our motivation for building the List class.
   (b) What would we need to keep track of, privately, to maintain state?
   (c) What would be an appropriate constructor? For each constructor, we should be able to determine reasonable values of the state variables; the constructor should leave the List in a valid state.
   (d) How do we implement int size()?
(e) Suppose we want to add a value to the List. How can we check to see if an List has the ability to hold n values? Thinking about how this is implemented is the most important part of the engineering of this structure.

(f) How do we implement void append(Object v)?

(g) How do we implement void insert(int i, Object v)? Where should it be added? What is the cost of implementing this method?

(h) How do we implement Object get(int i)? How does the user treat the return value? Do we have similar problems with void set(int i, Object v)?

(i) How do we implement boolean contains(Object v)?

*