We develop a data structure that branches.

1. Questions?

2. We investigate a new data structure, a *binary tree*.
   (a) The tree is composed of nodes, each of which holds data item.
   (b) The nodes can optionally reference up to two other subtrees.
   (c) Nodes without subtrees are called *leaves*.
   (d) The node not referenced by any other node is the tree's *root*.

3. Another definition. A *binary tree* is:
   (a) Empty, or
   (b) A data-carrying node that has two subtrees.

4. There are notions of *depth* (maximum distance from root to leaf), *degree* (number of non-empty subtrees), *ancestor* and *descendant* (nodes encountered rootward or leafward, respectively), *balance* (symmetry), etc. All can be identified or computed recursively.

5. Example: A database for a “twenty questions” game:
   (a) The database is a tree whose interior nodes represent questions and whose leaves represent objects-to-be-guessed.
   (b) A trivial database contains a single object.
   (c) A question partitions a collection of guessable objects into two sub-databases.

6. Data persistence. Often we wish to have a program's memory-resident data persist between runs of the program. Once process for handling this is generically referred to as *data serialization*.
   (a) When data needs to be preserved, it is handed over to a serializer that saves the data directly to a file, or develops a string that that can be written to a files.
   (b) When data is restored, it is read from the file, reconstructing the exact structure (an object that is equal but not the same).
   (c) Worry: writing the data is easy. Preserving references is harder. (How would you write an circular list?)
   (d) Worry: the data is written in some order; it is serialized. What order is necessary? sufficient?
   (e) Worry: the system needs to be *extensible* so that it can seamlessly serialize user-designed classes and objects.
   (f) In Python, the process is called *pickling*. The supporting module is *pickle*.
i. Idiom for saving a database:
   ```
   with open('database.pickle', 'wb') as handle:
       pickle.dump(db, handle, protocol=pickle.HIGHEST_PROTOCOL)
   ```

ii. Idiom for restoring a database (make sure it exists, first!):
   ```
   with open('database.pickle', 'rb') as handle:
       db = pickle.load(handle)
   ```

iii. The 'rb' and 'wb' access modes indicate that the file produced is not human-readable; it’s binary.

(g) Another approach is to produce JSON (JavaScript object notation) files which are human-readable. The json module is used.

i. Idiom for saving db:
   ```
   with open('database.json','w') as handle:
       json.dump(db, handle)
   ```

ii. Idiom for restoring db:
   ```
   with open('database.json') as handle:
       db = json.load(handle)
   ```

iii. Here, the database is human-readable. If you’re using Unicode characters (e.g. emoji and the like), add the binary 'wb' or 'rb' mode character.

*