6.00 Recitation 6

Topics: Object-Oriented Programming, Classes, Inheritance, Abstract Classes, Try...Except

Classes
- Collections of **objects** with similar characteristics
- Have already seen them
  - Eg class dict, class list
- Clients of a class should only care about the specifications of its methods, not the implementation details. The implementation details can change.
- What does a class need?
  - Variables
    - Should never be accessed directly, but always through a getter method (part 6 of the style guide). If your getter method is returning a mutable type, the getter method should take care of cloning
  - Methods
    - functions that are inside classes
    - call methods with dot notation, first argument (normally called self) is what comes before the dot
  - Often an initializer method (**init**)?
    - used to create **instances** of the class, ie make the actual objects
    - also know as a constructor
    - automatically invoked when a class object is instantiated. You do not need to call **init** directly
  - Other special methods
    - **str** (useful for printing out an object)
    - **getitem**(if x is your object, used for x[i])
    - **lt** (less than, used for .sort)

Try...Except
- Different types of errors
  - Examples that you might have seen from writing code: ValueError, IndexError, TypeError, NameError
- unhandled vs. handled
- execute in try block. If nothing goes wrong and the try block is completed, skip the except block
- As soon as an exception is raised in the try block, jump to the except block
- can make an exception happen using “raise”
- Fail fast

Inheritance
- Code example: Mother, Father, Child all inherit from FamilyMember
• Class variables vs instance variables
  o Class variables are associated with the class. Changing a class variable will change that variable for all of the objects associated with the class
  o Instance variables are associated with an object. Changing those will only affect the object you change them in.
• Getter and Setter methods
  o Never access or modify variables directly (part 6 of the style guide)
    ■ Modifying the variables might break how different parts of the object work
    ■ The implementation of the object might change, breaking your code
• You can use methods in parent classes
• Subclass vs. superclass