CSI 31 LECTURE NOTES (Ojakian)

Topic 13: Class Details

OUTLINE

References: Goldwasser/Letscher ch 10.1, 10.2, 10.3

1. References to Classes

   Running example: Use Account with balance

   (a) Identifier as: Reference or Pointer, separate from actual data of object.

   (b) (Roughly) Identifier = Name = Alias = Reference = Pointer

   (c) Implications of multiple names for same object.
   Example: Do two names, one object.

   (d) Equlity Testing and __eq__ over-ride.
   Example: Do two objects, "same" data. Use `id`.

   (e) Objects with no pointer: garbage collection.

   (f) (Possible) exception to rule: Immutable primitive classes.
   Example: Str

2. Implications of sending objects to functions

   (a) Call by value versus call by reference.

   (b) Calling a function is equivalent to assigning the *formal parameters* in the function definition to the arguments used in the function call.

   (c) Examples

   (d) Call by Reference: Advantage in space and speed. Disadvantages in protecting unwanted modifications to an object.

3. The "self" argument

   (a) Just a variable. Try different names.

   (b) Refers to the constructed object.

   (c) Object use own method - with "self"

4. References in lists and tuples

   Are you changing a pointer or the data being pointed to?

   (a) Lists store references, not the actual data (picture page 342)
   Example: Make list of objects and note change to one is change to all ...

   (b) Example: Trying to change value in a tuple.

   (c) Example: Trying to modify an object in a tuple.

5. Copying an Object

   (a) With lists, sets, dictionaries

      i. Shallow list copy
      ii. Deep list copy.

   (b) In classes: Make your own `copy` method!

6. Power of Aliasing

   Example (ch. 10, p.344): You and spouse with accounts, 3 ways to store accounts:

   (a) Completely dependant (reference same list)
   (b) Partially dependant (different lists, same references),
   (c) Completely independant (different lists, different references).