Objectives

• Designing our own classes
   Representing attributes/data
   What functionality to provide
• Using our defined classes

Reflection on Lab

• Data File Length:
   Female first names: 1025
   Male first names: 1119
   First names: 2144
   Last names: 2144
• How would you need to change your code to handle files that had 10,000 names? 100,000 names?
• How would you need to change your code to handle files that had 10,000 majors?

Reflection on Lab

• Became really abstract
   Partly a problem with Python
• Lose track of data types
   Keep telling yourself “This object is type X. That means I can do these operations on it…”
   Example:
     Dictionary: mapped strings → integers
     Dictionary: mapped strings → FrequencyObject
     Values from dictionary: list of FrequencyObjects

Where We Are

• With what you now know (OO programming)
   Opens up the possibilities for what you kinds of programs you can write
   Just about anything computational is possible

Review: Classes and Objects

• We’re all of type homo sapien
• Each of us has these attributes:
   Height
   Weight
   Hair color
   Hair type
   Skin color
• Methods
   Breathe
   Speak …

Classes and Objects

• c1 = Card(14, “spades”)
• c2 = Card(13, “hearts”)

Instanc variables, attributes, or fields
Creating a Deck Class (Partial)

- List of Card objects

```python
from card import *

class Deck:
    def __init__(self):
        self.cards = []
        for suit in ['clubs', 'hearts', 'diamonds', 'spades']:
            for rank in xrange(2, 15):
                self.cards.append(Card(rank, suit))

    def __str__(self):
        result = ''
        for c in self.cards:
            result += c.__str__() + 'v'
        return result

```

Adding Deck Functionality

- Functionality:
  - Shuffle the cards
  - Deal one card
  - Number of cards remaining
- What do the method headers look like?
- What should they return?
- How do we implement them?

Deck API

- What methods should our Deck class provide?
  - `shuffle()`
    - Shuffles the cards
  - `draw()`
    - Draws one card from the Deck object
  - `deal(numplayers, numcards)`
    - Deals numcards to each of the numplayers players

Deal Discussion

- Return proposals, given that a hand is a list of cards
  - Return a dictionary of hands
  - Preferred: Return a list of hands
    - Dictionaries take up a lot of space, much more than a list that's as long as the number of players

Extra Credit Opportunity

- Write additional code for Deck and Card classes
  - Leading to a game...
  - Possible functionality
    - Dealing a hand to a player
    - Texas Hold’em?
    - Alternative comparisons of cards for your game
- Adding a Player class for a particular game
- Due next Tuesday before lab
Creating a Counter Class

- Has a fixed range
- Starts at some low value, increments by 1, loops back around to low value if gets beyond some maximum value
- Example application of the counter: Caesar cipher for letters ‘a’ to ‘z’

What is the API for this object/class?

Object o of type Counter

- What are the attributes of an object in the class?
- What data should be used to represent an object in the class?

Creating a Counter Class

- Data: Instance variables
  - High, Low, Current Value
- API (methods)
  - Counter(low, high)
  - increment([amount])
  - setValue(value)
  - getValue()
  - getLow()
  - getHigh()

This Week

- Lab 9 due Friday
- One Environmental Monitoring article