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: 876
  - Male first names: 865
  - First names, last names: 1741
- How would you need to change your code to handle files that had 10,000 names? 100,000 names?

Reflection on Lab

- Became *really* abstract
  - Partly a drawback of Python
- Lose track of data types
  - Keep telling yourself "This object is type X. That means I can do these operations on it…"
  - Examples:
    - Dictionary: mapped `string` → `integer`
    - Dictionary: mapped `string` → `FrequencyObject`
    - Values from dictionary: list of `FrequencyObjects`
- Use example programs

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

- What is the keyword to create a new class?
- How do you create a new object of a given class?
  - What method does this call?
- What parameter is needed in every method?
- How do we access instance variables in other methods?

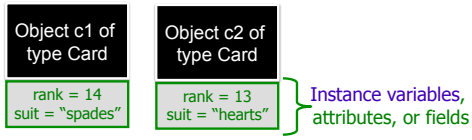
Review: Classes and Objects

- We're all of type *homo sapien*
- Each of us has these **attributes**:
  - Height
  - Weight
  - Hair color
  - Hair type
  - Skin color

  We all have these attributes, different values for the attributes
- Methods
  - Breathe
  - Speak …

Classes and Objects

- c1 = Card(14, "spades")
- c2 = Card(13, "hearts")



Creating a Deck Class (Partial)

- List of Card objects

```
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 += str(c) + "\n"
        return result
```

Initialize instance variable, `self.cards`

Creates and returns a string

Displays cards on separate lines

Adding Deck Functionality

- Shuffle the cards
- Number of cards remaining
- Draw one card
- Deal out cards

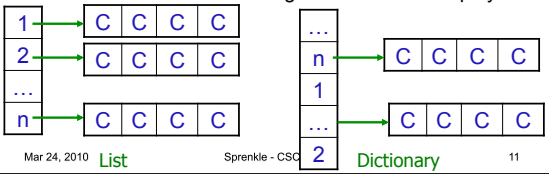
- What do the method headers look like?
- What should they return?
- How do we implement them?

Deck API

- `shuffle()`
  - Shuffles the cards
- `draw()`
  - Removes one card from the `Deck` and returns it
- `numRemaining()`
  - Returns the number of cards that are in the deck
- `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



Deck API

- `Deck()` ← Constructor
- `shuffle()`
- `draw()`
- `numRemaining()`
- `__str__()`

Show help(Deck), help(Card)

Extra Credit Opportunity

- Write additional code for Deck and Card classes
  - Leading to a game…
- Adding a Player class for a particular game
- Due next Tuesday before lab

Extra Credit Functionality Ideas

- Return the card's color (Red/Black), using a constant defined at the top for each color
  - What game is this useful for?
- Boolean methods: isBlack(), isRed()
- Boolean method: isOppositeColor(card)
- Boolean method: isSameSuit(card)
- Create a Hand class (very similar to Deck class)
  - Methods that check if all same suit, all same rank
- Player class for various games …
- Test/Demonstrate your methods

Due 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()

counter.py

This Week

- Lab 9 due Friday
- One Environmental Monitoring article