Objectives

- Designing our own classes
  - Representing attributes/data
  - What functionality to provide
- Using our defined classes
- Broader Issue: environmental monitoring

Discussion

- Jan Cuny talk
  - No class on Monday
- Reflection on lab
  - Anyone get a tester?
  - Lists, Functions

Review

- When defining a function, how can we make a parameter have a *default value*?
- Compare some properties of dictionaries and lists
  - When should you use one over the other?

Review: Object-Oriented Programming

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

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

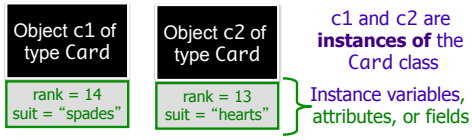
- We're all of type *homo sapien*
- Each of us has these **attributes**:
  - Height
  - Weight
  - Hair color
  - Hair type
  - Skin color

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

Each of us is an **instance of** the homo sapien class

Review: Classes and Objects

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



Review: Card API

- Card(rank, suit)
- getRank()
- getSuit()
- getRummyValue()
- __str__()
  - Can call using str(card)

Where We Left Off…

Now that we have the Card class, how can we **use** it?

- Can make a **Deck** class
  - What data should a Deck contain?
  - How can we represent that data?
- To start: write methods **__init__** and **__str__**
  - What do the method headers look like?

Creating a Deck Class (Partial)

- List of Card objects

So we can use the Card class

Initialize instance variable, self.deck

```
from card import *

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

Next: write **__str__** method
- What is the method header?
- What data type does it return?

Creating a Deck Class (Partial)

- List of Card objects

So we can use the Card class

Initialize instance variable, self.deck

Creates and returns a string

Displays cards on separate lines

```
from card import *

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

    def __str__(self):
        result = ""
        for c in self.deck:
            result += str(c) + "\n"
        return result
```


Deck API

- What does the Deck API look like so far?

Deck API

- `Deck()`
- `__str__()`

How could we test these methods?

Deck API

- `Deck()` Constructor
- `__str__()`

What would be useful to add to the API?

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?

Filling in the 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(numcards)`
  - Deals `numcards` out

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

Broader Issues: Environmental Monitoring

- Interdisciplinary projects involving sensor networks
  - Important new-ish CS research area
- Disclaimer:
  - Not a seismologist or a biologist
- Spring Term: Kamin Whitehouse, UVa
- Groups:

Zebra:	Zebra:	Volcano:
Callie	Meng	Colin
Lida	Nick	Will
Yates	Ola	Anh
Minh	Jean Paul	

Discussion

- What is the project?
- What are the CS challenges to the projects?
  - Any challenges only applicable to one project?
- How does the environment impact the CS research problems/solutions?
- How did the researchers address these challenges?
  - How would ***you*** address the challenges?

Overview of Challenges: Efficiency

- Some programmers thought that efficiency didn't matter anymore
  - GB of memory, terabytes of storage on machines
- Now: small and embedded devices
  - Need to be efficient!
- Energy in battery powered nodes
- Amount of data stored (when to delete?)
- When, amount of data transferred

Overview of Challenges: Reliability

- Data delivery
  - Missing data
  - Connectivity (good signal?)
  - Duplicate data (different sources?)
  - Dead sensor nodes
  - Calibration of data (time synchronization)
- Nodes
  - Withstand extreme weather, conditions
  - Battery life
- Robustness: recover from software failure/ malfunction or bad data

Overview of Challenges

- Testing
  - Accurately simulate conditions (which will vary widely over long periods of time)
- Different goals from domain scientists
  - CS: push boundaries of sensor networks
    - Example: Improve reliability of data to 95%
    - Seismologists: need 100% reliable data

Looking ahead to next week

- Monday: No class
  - Study for exam; practice problems
- Tuesday: Lab
  - Practice with dictionaries, object-oriented programming
- Friday
  - Exam, in class, on paper
  - Review document on line
    - no creating your own classes
  - No broader issue