Objectives

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

Review

- What did yesterday's lab bring together?
  - What were some different things you practiced?
- If I gave you a file of all the names from the US Census, how much code would you need to change to process/graph the most common names?
- How long did it take the computer to write the outputs of all four files?
- Why classes and objects?
- How do we create new data types?

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
- Example: Student – for the Registrar
  - Data to model for a Student?
  - API for a Student?

Review: Classes and Objects

- Student class
- Each student has these **attributes**:
  - First name
  - Last name
  - Expected graduation
  - Majors
  - Minors

Students all have these attributes, different values for the attributes

- Methods
  - getExpectedGraduationYear()
  - getFirstName()
  - declareMajor(major)

Each student is an **instance of** the Student class

Review: Object-Oriented Programming

- Why do we want to define classes/new data types?
- 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 methods?

Card Class (Incomplete)

Doc String

```
class Card:
    """ A class to represent a standard playing card.
    The ranks are ints: 2-10 for numbered cards, 11=Jack,
12=Queen, 13=King, 14=Ace.
    The suits are strings: 'clubs', 'spades', 'hearts',
'diamonds'."""
    def __init__(self, rank, suit):
        """Constructor for class Card takes int rank and
             string suit."""
        self._rank = rank
        self._suit = suit

    def getRank(self):
        "Returns the card's rank."
        return self._rank

    def getSuit(self):
        "Returns the card's suit."
        return self._suit
```

Methods

Methods are like *functions* defined in a *class*

card.py

Algorithm for Creating Classes

1. Identify need for a class
2. Identify state or attributes of a class/an object in that class
3. Write the constructor (`__init__`) and `__str__` methods
4. Identify methods the class should provide
   - How will a user call those methods (parameters, return values)?
     - Develop API
   - Implement methods

Using the `Card` class

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

```
from card import *

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

Initialize ***instance variable***, `self._listOfCards`

How would we want to display a deck?

Actual code should have doc strings

Creating a Deck Class (Partial)

- List of Card objects

```
from card import *

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

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

Creates and returns a string

Represents cards on separate lines

Actual code should have doc strings

Deck Class

- What does the Deck API look like so far?

Deck API

- `Deck()` Constructor
- `__str__()`
  - `str(<deck>)`

Deck API

- What additional methods should our Deck class provide?
- What do the method headers look like?
  - Deck's API
- What should they return?
- How do we implement them?

Deck API

- `Deck()` ← Constructor
- `shuffle()`
- `draw()`
- `deal(num_cards)`
- `numRemaining()`
- `isEmpty()`
- `__str__()`

Implement them!

Exam 2 Questions

- Content
  - Everything up through dictionaries
    - (Not creating our own classes)
  - Cumulative
- What types of questions are you expecting?

Looking Ahead

- Exam 2 on Friday
- Lab 9 due on Friday