

Objectives

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

Mar 19, 2012

Sprenkle - CSCI111

1

Review

- Compare properties of dictionaries and lists
 - When should you use one over the other?
- Problem from last week
 - Missing a JR from the lastname→year mapping
- Reason: two students have the same last name!

Mar 19, 2012

Sprenkle - CSCI111

2

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: Car
 - Data to model for a Car?
 - API for a Car?

Mar 19, 2012

Sprenkle - CSCI111

3

Review: Classes and Objects

- Car class
- Each car has these **attributes**:
 - Make
 - Model
 - Year
 - Transmission
 - Exterior color

Cars all have these attributes, different values for the attributes
- **Methods**
 - getYear()
 - setGear()
 - ...

Each car is an **instance of** the Car class

Mar 19, 2012

Sprenkle - CSCI111

4

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 other methods?

Mar 19, 2012

Sprenkle - CSCI111

5

Review: Classes and Objects

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

Object c1 of
type Card

rank = 14
suit = "spades"

Object c2 of
type Card

rank = 13
suit = "hearts"

c1 and c2 are
instances of
the Card class

Instance variables,
attributes, or fields

Mar 19, 2012

Sprenkle - CSCI111

6

Card Class (Incomplete)

```
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
```

Doc String

Methods are like functions defined in a class

Methods

Mar 19, 2012 Sprenkle - CSC1111 card.py 7

Defining the Constructor

- `__init__` method is like the **constructor**
- In constructor, define **instance variables**
 - Data contained in every object
 - Also called **attributes** or **fields**
- Constructor **never returns anything**
 - First parameter of **every** method is **self**
 - pointer to the object that method acts on

```
def __init__(self, rank, suit):
    """Constructor for class Card takes int rank
    and string suit."""
    self.rank = rank
    self.suit = suit
```

Instance variables

Mar 19, 2012 Sprenkle - CSC1111 8

Using the Constructor

```
def __init__(self, rank, suit):
```

- As defined, constructor is called using **Card(<rank>, <suit>)**
 - Do not pass anything for the **self** parameter
 - Python handles for us, passing the parameter automatically
- Example:
 - `card = Card(2, "hearts")`
 - Creates a 2 of Hearts card
 - Python passes `card` as **self** for us

Object card of type Card

```
rank = 2
suit = "hearts"
```

Mar 19, 2012 Sprenkle - CSC1111 9

Accessor Methods

- Need to be able to get information about the object

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

- Have **self** parameter
- Return data/information

- These will get called as **card.getRank()** and **card.getSuit()**
 - Python plugs `card` in for **self**

Mar 19, 2012 Sprenkle - CSC1111 10

Another Special Method: `__str__`

- Returns a **string** that describes the object
- Whenever you **print** an object, Python checks if the object's `__str__` method is defined
 - Prints result of calling `__str__` method
- `str(<object>)` also calls `__str__` method

```
def __str__(self):
    """Returns a string describing the card as
    'rank of suit'."""
    result = ""
    if self.rank == 11:
        result += "Jack"
    elif self.rank == 12:
        result += "Queen"
    elif self.rank == 13:
        result += "King"
    elif self.rank == 14:
        result += "Ace"
    else:
        result += str(self.rank)
    result += " of " + self.suit
    return result
```

self is a Card object

Mar 19, 2012 Sprenkle - CSC1111 11

Using the Card Class

```
def main():
    c1 = Card(14, "spades")
    print(c1)
    c2 = Card(13, "hearts")
    print(c2)
```

Invokes the `__str__` method

Displays:

Ace of spades
King of hearts

```
Object c1 of type Card
rank = 14
suit = "spades"
Object c2 of type Card
rank = 13
suit = "hearts"
```

Mar 19, 2012 Sprenkle - CSC1111 12

Example: Rummy Value

- **Problem:** Add a method to the Card class called **rummyValue** that returns the value of the card in the game of Rummy
- **Procedure** for defining a method (similar to functions)
 - What is the input?
 - What is the output?
 - What is the method header?
 - What does the method do?
- How do we call the method?

Mar 19, 2012

Sprengle - CSCI111

card2.py

13

Card API

- Based on what we've seen/done so far, what does the Card class's API look like?

Mar 19, 2012

Sprengle - CSCI111

14

Card API



Implementation of methods is hidden

- Card(<rank>, <suit>)
- getRank()
- getSuit()
- rummyValue()
- __str__()

Mar 19, 2012

Sprengle - CSCI111

15

Using the Card class

- Having the Card class means that we can represent a Card in code

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

Mar 19, 2012

Sprengle - CSCI111

16

Review

```
from graphics import *
win = GraphWin("Picture")
win.setBackground("black")
```

```
from card import *
c = Card(7, "diamonds")
print(c.getRank())
```

- Same programming as before
- Just defining our own classes

Mar 26, 2010

Sprengle - CSCI111

17

Using the Card class

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

- Let's write a simplified version of the game of War
 - Basically just part of a round
- What are the rules of War?

Mar 19, 2012

Sprengle - CSCI111

war.py

18

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?
 - Consider other functionality we may want to implement
- To start: write methods `__init__` and `__str__`
 - What do the method headers look like?

Mar 19, 2012

Sprengle - CSCI111

19

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
```

Initialize instance variable, self.listOfCards

Creates and returns a string

Displays cards on separate lines

Mar 19, 2012

Sprengle - CSCI111

20

Deck Class

- What does the Deck API look like so far?

Mar 19, 2012

Sprengle - CSCI111

21

Deck API

- Deck() Constructor
- __str__()

How could we test these methods?

Mar 19, 2012

Sprengle - CSCI111

22

Using the Deck Class

- How can we use the Deck that we just wrote?

Mar 19, 2012

Sprengle - CSCI111

23

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?

Mar 19, 2012

Sprengle - CSCI111

24

Adding Deck Functionality

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

Mar 19, 2012

Sprenkle - CSCI111

25

Algorithm for Creating Classes

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

Mar 19, 2012

Sprenkle - CSCI111

26

Looking Ahead

- Lab 9: Analysis of student names
- Fri: Exam 2
- Wed: Bring your questions about the exam

Mar 19, 2012

Sprenkle - CSCI111

27