Objectives

- Wrap up dictionaries
- Defining our own classes

CSCI Courses

- CSCI 112: Fundaments of Programming II - MWF 11:15a-12:10p, R 9:05a-12:10p
- Spring 2013: CSCI 250: Introduction to Robotics (prereq: CSCI111)
- Spring 2014: CSCI 251: iPhone Development (prereq: CSCI112)

- CSCI minor: 6 CSCI courses
- CSCI major: 10 CSCI courses + 2 math courses

Review: Dictionaries

- What is a dictionary in Python?
- What is the syntax for creating a new dictionary?
- How do we access a key's value from a dictionary?
  - What happens if there is no mapping for that key?
- How do we create a key → value mapping in a dictionary?
- How can we iterate through a dictionary?

Review

- Given a file of the form
  - <lastname> <year>
- Create a mapping between the last name and year, i.e., I want to be able to quickly find out what a student's class year is
  - How do we want to model the data?
  - What is the key? What is the value?
  - How to display the mapping in a pretty way?
  - What order is the data printed in?

`years_dictionary.py`

Problem

- Modify the previous program to keep track of the *number* of students of each year
  - How do we want to model the data?
  - What is the key? What is the value?
  - Could we solve this using a list?

`years_dictionary2.py`

Analyzing years_dictionary2.py

- Anything useful/general that we could put in a function?

Equivalent Solutions

```
if key not in dictionary :
   dictionary[key] = 1
else:
   value = dictionary[key] + 1
   dictionary[key] = value
```

```
if key not in dictionary :
   dictionary[key] = 1
else:
   dictionary[key] += 1
```

Discussion

- Compare lists and dictionaries
  - What are their properties?
  - How are they similar?
  - How are they different?
  - When do you use one or the other?

Lists vs. Dictionaries

Lists	Dictionaries
integer *positions* (0, …) to any type of value	Map immutable *keys* (int, float, string) to any type of value
Ordered	Unordered
Slower to find a value (`in`)	Fast to find a value (use key)
Fast to print in order	Slower to print in order (by key)
Only as big as you make it	Takes up a lot of space (so can add elements in the middle)

ABSTRACTIONS

Abstractions

- Provide ways to think about program and its data
  - Get the jist without the details
- Examples we've seen
  - Functions and methods `encodeMessage(phrase, key)`
    - Used to perform some operation but we don't need to know how they're implemented
  - Dictionaries
    - Know they map keys to values
    - Don't need to know how the keys are organized/ stored in the computer's memory
  - Just about everything we do in this class…

Classes and Objects

- Provide an abstraction for how to organize and reason about data
- Example: `GraphWin`
  - Has ***attributes*** (i.e., data or state) background color, width, height, and title
  - Each `GraphWin` object has these attributes and its own values for these attributes
  - Used methods (**API**) to modify the object's state, get information about attributes

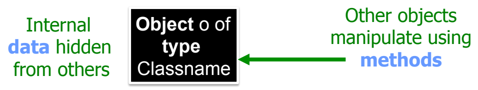
`GraphWin`

Defining Our Own Classes

- Often, we want to represent data or information that we do ***not*** have a way to represent using *built-in types* or *libraries*
- Classes provide way to *organize* and *manipulate* data
  - Organize: data structures used
    - E.g., ints, lists, dictionaries, other objects, etc.
  - Manipulate: methods

What is a Class?

- Defines a new ***data type***
- Defines the class's ***attributes*** (i.e., data or state) and ***methods***
  - Methods are like **functions** *within* a class and are the class's **API**



Defining a Card Class

- Create a class that represents a playing card
  - How can we represent a playing card?
  - What information do we need to represent a playing card?

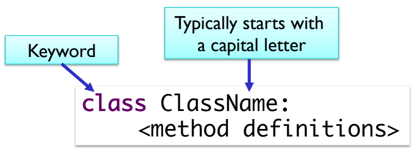
Defining a Card Class

- Create a class that represents a playing card
  - How can we represent a playing card?
  - What information do we need to represent a playing card?
- Do we ***need*** a class to represent a card?
  - Does any built-in data type naturally represent a card?

Representing a Card object

- Every card has two attributes:
  - Suite – a string that's either "hearts", "diamonds", "clubs", "spades"
  - Rank – an integer
    - 2-10: numbered cards
    - 11: Jack
    - 12: Queen
    - 13: King
    - 14: Ace
- Alternative: use a unique string or integer that encodes the suite and rank

Defining a New Class



```
class ClassName:
     <method definitions>
```

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

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

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
    - Passes the parameter *automatically*

Object card of type Card

rank = ?
suit = ?

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”

Accessor Methods

- Need to be able to get information about the object

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

```
def getRank(self):
   "Returns the card’s rank."
   return self.rank

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

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

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

self is a Card object

```
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
```

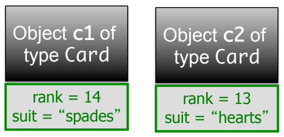
Using the Card Class

Invokes the `__str__` method

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

Displays:

Ace of spades
King of hearts



Mar 16, 2012 Sprenkle - CSCI111 25

Broader Issues: Environmental Monitoring

- Interdisciplinary projects involving sensor networks
  - Important new-ish CS research area
- Disclaimer:
  - Not a seismologist or a biologist
- Groups:

Zebra: Sam, Hang, Cory, Deirdre

Zebra: Kari, John K, Drew, Liu

Zebra: John G, Haley, Phil, Trang

Volcano: Mary, Josh, Gabi, Colby, Will

Mar 18, 2011 Sprenkle - CSCI111 26

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?
- What are other projects where we could apply sensor networks?

Mar 18, 2011 Sprenkle - CSCI111 27

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

Mar 18, 2011 Sprenkle - CSCI111 28

Overview of Challenges: Reliability

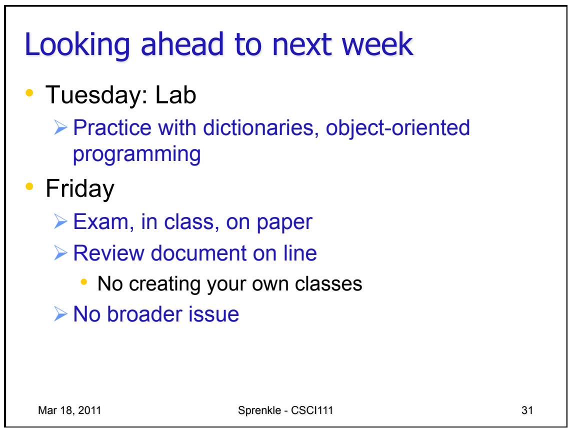
- Data delivery
  - Missing data
  - Connectivity (good signal?)
  - Duplicate data (different sources?)
  - Dead sensor nodes
  - Calibration of data (time synchronization)
- Nodes
  - Withstand extreme weather, conditions
  - Battery life
- Robustness: recover from software failure/ malfunction or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

Mar 18, 2011 Sprenkle - CSCI111 29

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

Mar 18, 2011 Sprenkle - CSCI111 30



Looking ahead to next week

- 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