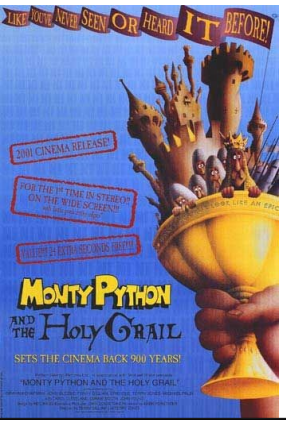
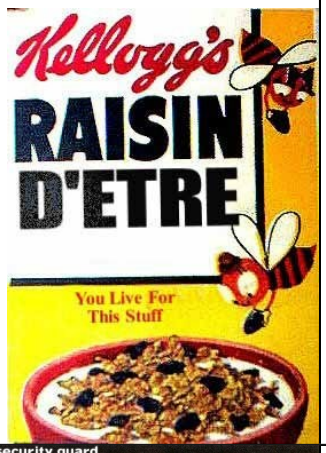
Python and Object-Oriented Programming



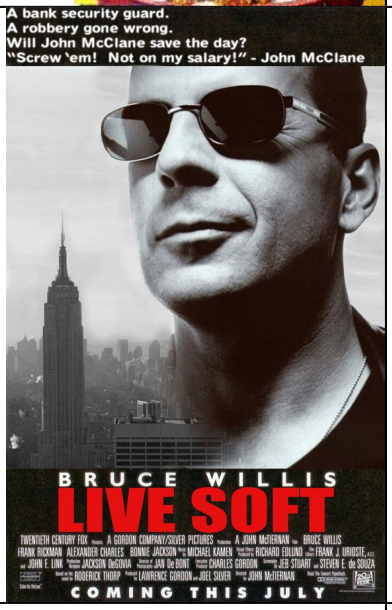


One-Slide Summary

- Real databases, unlike PS5, have many concerns, such as scalability and atomic transactions.
- An **object** packages state and procedures.
- A procedure on an object is called a **method**. We **invoke** a method by sending the object a **message**.
- **Inheritance** allows one object to refine and reuse the behavior of another. This is a good thing.

Outline

- PS5 vs. the Real World
- Problem Sets and PS9
- Python
- Object-Oriented Programming
  - Object = State + Methods
- Inheritance



Interlude: PS5 vs. Wild

How are commercial databases different from what you implemented for PS5?

UVa's Integrated Systems Project to convert all University information systems to use an Oracle database was originally budgeted for **$58.2 Million** (starting in 1999). Actual cost ended up over $100 Million.

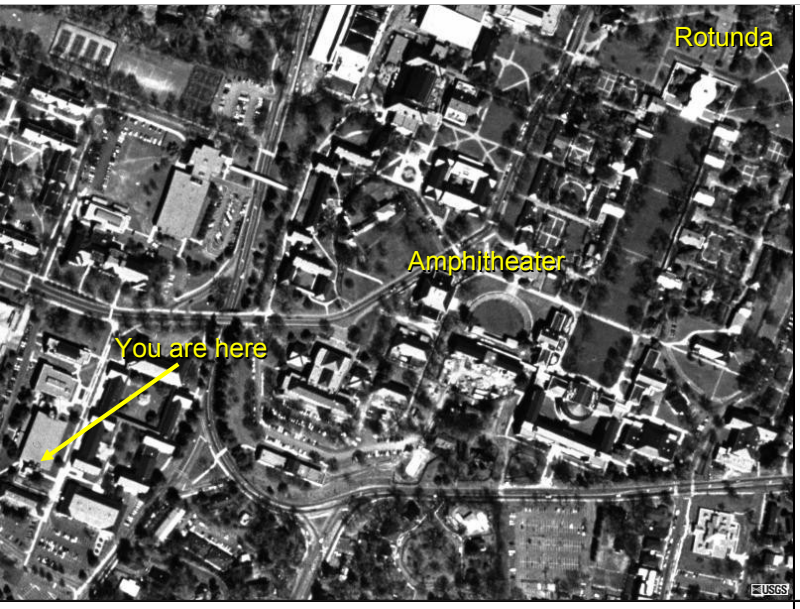
http://www.virginia.edu/isp/

Real Databases

- Atomic Transactions: a transaction may involve many modifications to database tables, but the changes should only happen if the whole transaction happens (e.g., don't charge the credit card unless the order is sent to the shipping dept)
- Security: limit read/write access to tables, entries and fields
- Storage: need to efficiently store data on disk, provide backup mechanisms
- Scale: to support really big data tables, real databases do lots of clever things

How big are big databases?

- Microsoft TerraServer
  - Claimed biggest in 1998
  - Aerial photos of entire US (1 meter resolution)
  - Let's see an example ...



Big Databases

- Microsoft TerraServer
  - 3.3 Terabytes (claimed biggest in 1998)
  - 1 Terabyte = 2^{40} Bytes ~ 1 Trillion Bytes
- Google Maps (possibly bigger?)
  - Better color …
- Wal-Mart
  - 285 Terabytes (2003)

- Stanford Linear Accelerator (BaBar)
  - 500 Terabytes (30 KB per particle collision)

How much work?

- Suppose we have a huge database.
- table-select is in $\Theta(n)$ where n is the number of entries in the table
  - Would your table-select work for Wal-Mart?
  - If 1M entry table takes 1s, how long would it take Wal-Mart to select from 285TB ~ 2 Trillion Entries?

How much work?

- table-select is in $\Theta(n)$ where n is the number of entries in the table
  - Would your table-select work for Wal-Mart?
  - If 1M entry table takes 1s, how long would it take Wal-Mart to select from 285TB ~ 2 Trillion Entries? 2 000 000s = ~ 23 days

How do expensive databases perform table-select so much faster?
Hint: How did we make sorting faster?

Objects

An ***object*** packages:

- **state** ("variables")
- **procedures** for manipulating and observing that state ("methods")

Why is this useful?

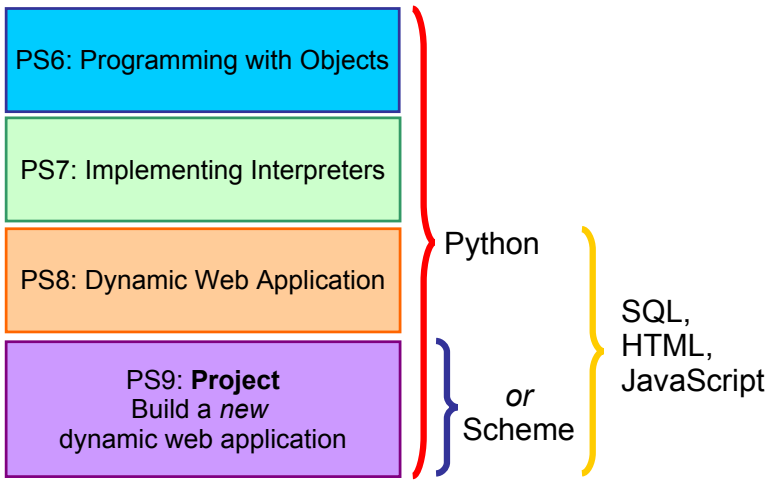
Problem-Solving Strategies

- PS1-PS4: **Functional Programming**
  - Focused on **procedures**
  - Break a problem into procedures that can be combined to solve it
- PS5: **Imperative Programming**
  - Focused on **data**
  - Design data for representing a problem and procedures for updating that data

Problem-Solving Strategies

- PS6: **Object-Oriented Programming**
  - Focused on **objects**: package procedures and state
  - Model a problem by dividing it into objects
  - Lots of problems in real (and imaginary) worlds can be thought of this way

Problem Sets after PS5

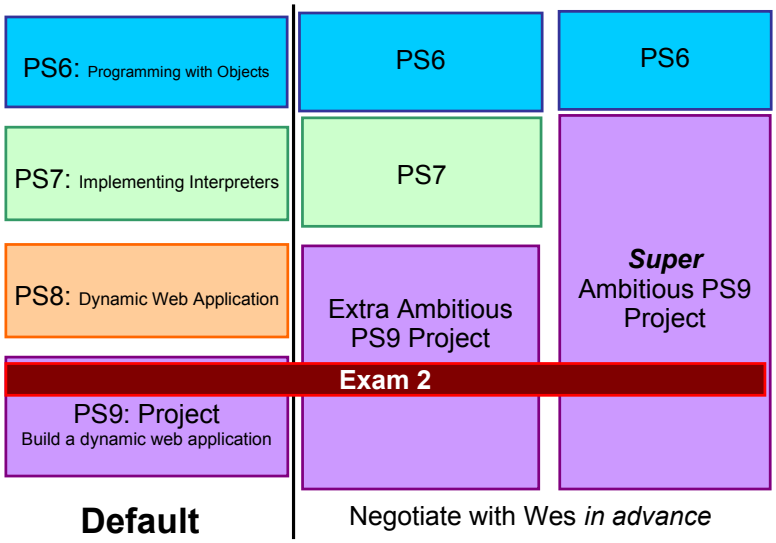


PS9 Assignment

Problem: Make an interesting dynamic web site.

- Teams of 1-40 students
- Can be anything you want that:
  - Involves interesting computation
  - Follows University's use policies (or on external server)
  - Complies with ADA Section 508 (accessible)

A list of example topics is provided.



Liberal Arts Trivia: Biology

- This egg-laying, venomous (from a calcaneus spur found on the hind limb), beaver-tailed, otter-footed mammal is perhaps best known for its "nose", which follows the style of the Anatidae family of birds. It is native to eastern Australia and Tasmania, and occurs on the Australian 20 cent coin.

Liberal Arts Trivia: Art History

- Name the Spanish surrealist artist who painted *The Persistence of Memory* (oil on canvas, 1931).

Most Popular Programming Languages

1. Java
2. C
3. PHP
4. C++
5. Visual Basic
6. C#
7. Python
8. Perl
9. Delphi
10. JavaScript

TIOBE Index, March 2010

The Reveal

- Scheme is secretly JavaScript
  - JavaScript is actually ECMAScript
  - Brendan Eich, the creator of JavaScript, is on record as saying that "ECMAScript was always an unwanted trade name that sounds like a skin disease."

Scheme vs JavaScript 1

```
(define (length x)
 (if (null? x)
  0
  (+ 1 (length (cdr x)))
))
```

```
function length(x) {
 return isNull(x) ?
  0 :
  (1 + (length(cdr(x))))
; }
```


Scheme vs. JavaScript 2

```
(define (map f lst)
 (let ((result null))
 (iter (lambda (elt)
   (set! result
    (cons (f elt) result))
 ))
 result))
```

```
function map(f,lst) {
 var result = [] ;
 forEach(lst,function
  (elt) {
 result.push(f(elt));
 });
 return result;
}
```


Scheme vs Javascript

- JavaScript is a dynamic, not-strongly-typed language with first-class functions.
- Scheme is a dynamic, not-strongly-typed language with first-class functions.
- For more information, see for example:
  - The Little JavaScripter
  - http://www.crockford.com/javascript/little.html
  - The Little Schemer
  - http://www.amazon.com/exec/obidos/ASIN/0262560992/wrrrldwideweb

Why Learn New Languages?

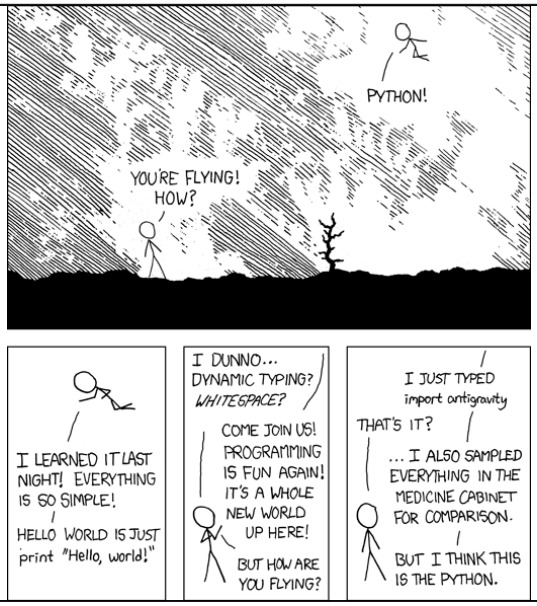
- Languages change the way we think.
  - The **linguistic relativity principle** (also known as the **Sapir-Whorf Hypothesis**) is the idea that the varying cultural concepts and categories inherent in different languages affect the cognitive classification of the experienced world in such a way that speakers of different languages think and behave differently because of it. Roger Brown has drawn a distinction between weak linguistic relativity, where language limits thought, and strong linguistic relativity, where language determines thought. [Wikipedia]
- See also: Orwell's *1984*

Why Learn New Languages?

- Deepening Understanding
  - By seeing how the same concepts we encountered in Scheme are implemented by a different language, you will understand those concepts better (especially procedures, assignment, and data abstraction).
- Building Confidence
  - By learning Python (mostly) on your own, the next time you encounter a problem that is best solved using a language you don't know, you will be confident you can learn it (rather than trying to use the wrong tool to solve the problem.)

Why Learn New Languages

- Fun! Programming in Python is fun (possibly even more fun than programming in Scheme!)
- Especially because:
  - It is an elegant and simple language
  - Most programs mean what you think they mean
  - It is dynamic and interactive
  - It can be used to easily build web applications
  - It is named after *Monty Python's Flying Circus*
  - It was designed by someone named Guido



Python

- **Python** is a **universal programming language**.
  - Everything you can compute in Scheme you can compute in Python, and vice versa
  - Chapter 11 and PS 7: implement a Scheme interpreter in Python
  - Chapter 12: more formal definition of a universal programming language
- Python is an **imperative language**.
  - Designed to support programming where most of the work is done using assignment statements
  - x = sqrt(4) + 1;

Objectifying Python

- Python is also an **object-oriented language**.
  - **Objects** encapsulate state (i.e., variables and information) and the methods that operate on that state together.
  - In Python, all data are objects.
  - Problem Set 6 covers programming with objects.
  - Python has built-in support for classes, methods and inheritance.

Learning New Languages

- **Syntax**: Where the {, !, (, :, etc., all go
  - If you can understand a BNF grammar, this is easy
  - But it still takes some getting used to
- **Semantics**: What does it mean?
  - Learning the evaluation rules
  - This is harder, but most programming languages have very similar rules (with subtle differenecs)
- **Style**: What are the idioms and customs?
  - Many years to be a "professional" Python programmer, but not long to write a program

Python If

- *Instruction* ::=
  if *Expression* **:**
    Block
  else:
    Block
- Semantics: Evaluate the *Expression*. If it evaluates to a true value, evaluate the first *Block*. Otherwise, evaluate the second *Block*.
- True value:
  - Python: anything not False, None, 0, empty string, empty container

Python If Example

```
if []:
  print "Empty is true!"
else:
  print "Empty is false!"
```

Empty is false!

Learning Python

- We will introduce (usually informally) Python constructs in class as we use them (and in example code in PS6)
- The "Schemer's Guide to Python" is an introduction to Python: covers the most important constructs you need for PS6, etc.
- Course book: Chapter 11 introduces Python
  - Read ahead to Section 11.1
- On-line Python documentation

Making Objects

ClassDefinition ::= **class** *Name* **:**
FunctionDefinitions

```
class Dog:
  def bark(self):
    print "wuff wuff wuff wuff"
```

In Washington, it's dog eat dog.
In academia, it's exactly the opposite.
- Robert Reich

Making a Dog

Assignment ::= *Variable* **=** *Expression*

```
class Dog:
  def bark(self):
    print "wuff wuff wuff wuff"
spot = Dog()
```

Python assignments are like both define and set! If the Variable name is not yet defined, it creates a new place. The value in the named place is initialized to the Expression.

Python Procedures

```
class Dog:
  def bark(self):
    print "wuff wuff wuff wuff"
```

FunctionDefinition ::= **def** *Name* **(** *Parameters* **):** *Block*
Parameters ::= *SomeParameters* | epsilon
SomeParameters ::= *Name* | *Name*, *SomeParameters*
Block ::= *Statement*
| <newline> indented(*Statements*)
Statements ::= *Statement* <newline> *MoreStatements*
MoreStatements ::= epsilon
| *Statement* <newline> *MoreStatements*

Some Python Procedures

```
def square(x):
  return x*x;
```

```
def bigger(a,b):
  if a > b:
    return a
  else:
    return b
```

FunctionDefinition ::= **def** *Name* **(** *Parameters* **):** *Block*
Parameters ::= *SomeParameters* | epsilon
SomeParameters ::= *Name* | *Name*, *SomeParameters*
Block ::= *Statement*
| <newline> indented(*Statements*)
Statements ::= *Statement* <newline> *MoreStatements*
MoreStatements ::= epsilon
| *Statement* <newline> *MoreStatements*

Whitespace Matters

```
def bigger(a,b):
  if a > b:
    return a
  else:
    return b
```

```
def bigger(a,b):
  if a > b:
    return a
 else:
    return b
```

```
File: <pyshell#1>", line 4
 else:
^
IndentationError: unindent does not match any outer indentation level.
```


Barking: Invoking Methods

ApplicationStatement ::= *Name*(*Arguments*)
Arguments ::= epsilon | *MoreArguments*
MoreArguments ::= *Argument* , *MoreArguments*
| *Argument*
Argument ::= *Expression*

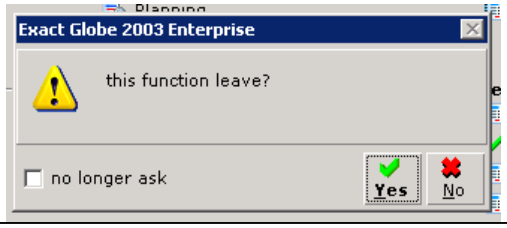
```
class Dog:
  def bark(self):
    print "wuff wuff wuff wuff"
spot = Dog()
spot.bark("Hello")
```

wuff wuff wuff wuff

Object.Method(*Arguments*) : Invoke method on object, with object as the first (self) argument!

Object Lingo

- "Apply a procedure" = "Invoke a method"
- We apply a procedure to parameters.
- We invoke a method on an object, and pass in parameters.
  - With the object itself as the first (self) parameter.

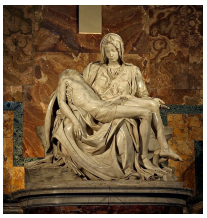


Liberal Arts Trivia: Art History and American Literature

- Give the Renaissance master (or Ninja Turtle) associated with each work of art:

(a) Tomb of Antipope John XXIII

(b) Mona Lisa



(c) Pieta

(d) Transfiguration

Liberal Arts Trivia: Polish History, Chemistry, and Physics

- This physicist and chemist of Polish upbringing and French citizenship was the first person honored with two Nobel prizes, the first woman to win a Nobel prize, and the first woman to serve as a professor at the University of Paris. The world's first studies into the treatment of cancers using radioactive isotopes were conducted under her direction.

Liberal Arts Trivia: Cooking

- This Japanese delicacy is vinegared rice, usually topped with other ingredients, including fish. The dish as we know it today was invented as a fast food by Hanaya Yohei at the end of the Edo period (19th century) in Tokyo: it could be eaten on the road side or in a theatre using fingers or chopsticks. The basic idea can be traced back to 4th century BCE China as a preservative: the fermentation of the rice prevents the fish from spoiling.

Dogs with Names


```
class Dog:
  def __init__(self,n):
    self.name = n
spot = Dog("Spoticus")
spot.name
Spoticus
ginger = Dog("Ginger")
ginger.name
Ginger
```

__init__ is a **constructor**
It creates a new object of the type.
It is called when Dog(n) is evaluated.

Review: Making a Dog

```
class Dog:
  def bark(self):
    print "wuff wuff wuff wuff"
spot = Dog()
```

Python assignments are like both define and set! If the Variable name is not yet defined, it creates a new place. The value in the named place is initialized to the Expression.

Python Lists

- Python has built-in datatypes for both mutable lists [ ] and immutable tuples ( ) .

```
>>> m = range(1,1000)     # range(1,100) ~ intsto(99)
>>> m[0]                  # m[0] ~ (mcar m)
1
>>> m[-1]                 # constant time in Python!
999
>>> len(m)                # constant time in Python!
999
>>> m[1:]                 # m[1:] ~ (mcdr m) ?
[2, ..., 999]
```


Is m[1:] like mcdr?

```
>>> m = range(1,1000)
>>> m1 = m[1:]
>>> m1[0]
2
>>> m[1]
2
>>> m1[0] = 3          # m[1:] is a new copy
>>> m1                 # of the list except for
2                      # the first element!
```


Implementing list-map in Python

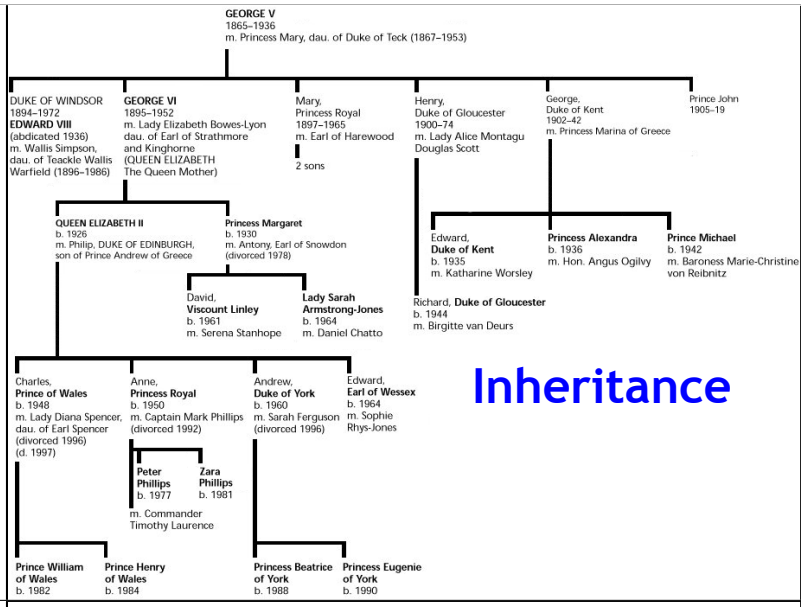
```
def scheme_map(f,p):
  if not p:
    return []
  else:
    return [f(p[0])] + scheme_map(f,p[1:])
```

- This "literal" translation is not a good way to do things.

Pythonic Mapping

```
def mlist_map(f, p):
  for i in range(0, len(p)):
    p[i] = f(p[i])
  return p
```

- Unlike the previous one, this mutates p.
  - This slide is **map!**.
- Python has a built-in map.



Inheritance

Hey, Scooby!

```
class Dog:
  def __init__(self,n):
    self.name = n
  def bark(self):
    print “wuff wuff wuff wuff”
class TalkingDog (Dog):
  def speak(self,stuff):
    print stuff
```

```
>>> scooby = TalkingDog(“Scooby”)
>>> scooby.speak(“Scooby Snack!”)
Scooby Snack!
```


Subclasses

```
class TalkingDog (Dog):
  def speak(self,stuff):
    print stuff
```

ClassDefinition ::= **class** *SubClassName* (*SuperClassName*) : FunctionDefinitions

- TalkingDog is a **subclass** of Dog.
- Dog is the **superclass** of TalkingDog.
  - Every TalkingDog is also a Dog.
  - (But not vice-versa.)

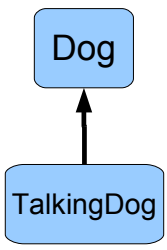
Every Dog Has Its Day

```
class Dog:
  def __init__(self,n):
    self.name = n
  def bark(self):
    print “wuff wuff wuff wuff”
class TalkingDog (Dog):
  def speak(self,stuff):
    print stuff
```

```
>>> ginger = Dog(“Ginger”)
>>> scooby = TalkingDog(“Scooby”)
>>> scooby.speak(“Scooby Snack!”)
Scooby Snack!
>>> ginger.speak(“Blah blah blah”)
AttributeError: Dog instance has no attribute 'speak'
>>> scooby.bark()
wuff wuff wuff wuff
```


Speaking About Inheritance

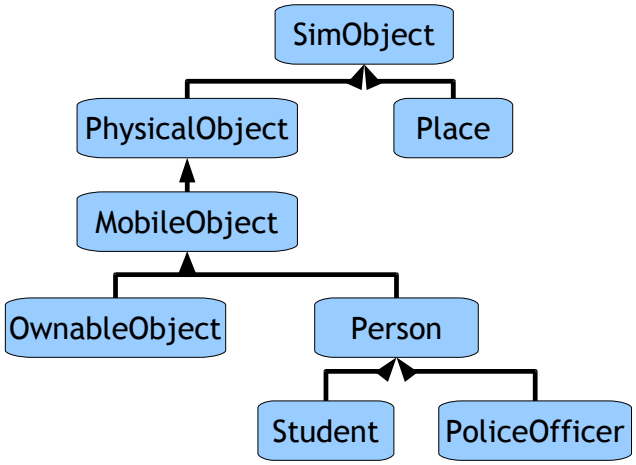
- Inheritance is using the definition of one class to define another class.
- TalkingDog **inherits** from Dog.
- TalkingDog is a **subclass** of Dog.
- The **superclass** of TalkingDog is Dog.
- *These all mean the same thing!*



Problem Set 6

- Make an adventure game by programming with objects.
- Many objects in our game have similar properties and behaviors, so we use inheritance.

PS6 Classes



Object-Oriented Terminology

- An **object** is an entity that packages state and procedures.
- The state variables that are part of an object are called **instance variables**.
- The procedures that are part of an object are called **methods**.
- We **invoke** (call) a method. The object is the first parameter (self).
- **Inheritance** allows one class to refine and reuse the behavior of another.
- A **constructor** is a procedure that creates new objects (e.g., __init__).

Charge

- Start PS6 early
  - PS6 is challenging
  - Opportunity for creativity
- Start thinking about PS9 Project ideas
  - If you want to do an "extra ambitious" project convince me your idea is worthy before Nov 10 (ps7 and 8) / Nov 17 (ps8)
  - Discuss ideas and look for partners **on the forum**

Homework

- PS 5 due *Today*
- PS 6 due Wednesday October 27th
- Read Chapter 11 (in particular, 11.1)