INTRODUCTION TO PROGRAMMING FOR SCIENTISTS

Lecture 5
Classes
PIL

STEVEN LUDTKE
N410, SLUDTKE@BCM.EDU
Homework Review
• Use ‘#’ to add comments (use often)
  
x=x*5      # This command multiplies x by 5

• When declaring new functions, an initial string after the declaration serves as documentation:
  
def f(x):
    "This function will return x*2"
    return x*2

  
help(f)
Programming

• Procedures
• Functions
• Objects
• Aspects
Object Oriented Programming

- A Class is a grouping of associated data elements and optionally associated code elements.

```python
class person:
    "This class describes a person"
pass
```
In C++/Java

class person {
    string firstname;
    string lastname;
    string address;
    string city;
    char state[2];
    int zipcode;
    int phone;
};
class Person:
    "This class represents a person"

a=Person()

a.firstname="Steve"
a.lastname="Ludtke"
a.address="1 Baylor Plaza"

print "%s %s
%s"%(a.firstname,a.lastname,a.address)
a={} 

a["firstname"]="Steve"
a["lastname"]="Ludtke"
a["address"]="1 Baylor Plaza"

print "%s %s
n%s"%(a["firstname"],a["lastname"],a["address"])
class Person:
    def show(self):
        print "%s %s
%s"(self.firstname, self.lastname, self.address)

    def fixcase(self):
        self.firstname = self.firstname.title()
        self.lastname = self.lastname.title()
Special Methods

- `__init__` - After the class is created
- `__del__` - Before the class is destroyed
- `__repr__` - What happens when you ‘print’
- `__str__` - String conversion
- `__cmp__` - Compare 2 objects (sort !)
- `__getitem__`, `__setitem__`, `__delitem__` - list, dict emulation
Special Methods

• `__cmp__(self, other)` - compare objects
• `__len__(self)` - length of object
• `__contains__(self, item)` - check if item in this object
• `__add__`, `__sub__`, `__mul__`, `__div__`, ...
class person:
    def __init__(self, firstname=None, lastname=None, address=None, city=None, state=None, zipcode=None, phone=None):
        self.firstname = firstname
        self.lastname = lastname
        self.address = address
        self.city = city
        self.state = state
        if zipcode != None:
            try:
                self.zipcode = int(zipcode)
            except:
                raise Exception, "zipcode must be a number"
        else:
            self.zipcode = None
        if phone != None:
            self.setphone(phone)

    def __repr__(self):
        return "%s %s
%s
%s, %s  %s
%s"%(str(self.firstname), str(self.lastname), str(self.address), str(self.city), str(self.state), self.getzipcode(), self.getphone())
        return "Undefined"

    def getzipcode(self):
        try:
            return "%05d"%self.zipcode
        except:
            return "None"

    def setphone(self, phone):
        phone = str(phone)
        elif len(phone) == 12 and phone[3] == "-" and phone[7] == "-" :  
            self.phone = phone[0:3] + phone[4:7] + phone[8:]
        elif len(phone) == 10 :
            self.phone = phone
        elif (len(phone) == 8 and phone[3] == "-") or len(phone) == 7 :
            raise Exception,"Please provide a full 10 digit number"
        else:
            raise Exception,"I don't recognize that format"
        try:
            x = int(self.phone)
        except:
            raise Exception,"You didn't provide a number in a valid format"

    def getphone(self):
        try:
            return "(%s)%s-%s"%(self.phone[:3], self.phone[3:6], self.phone[6:])
        except:
            return "Unknown"
Inheritance

Person

Employee

Faculty

Staff

Student

Grad Student

Undergrad
class student(person):
    def __init__
        (self, firstname=None, lastname=None, address=None, city=None, state=None, zipcode=None,
         phone=None, date_started=None, mentor=None):
            person.__init__(firstname, lastname, address, city, state, zipcode, phone)
            self.date_started = date_started
            self.mentor = mentor

        def __repr__(self):
            return "%s
mentor: %s
started: %s"%(person.__repr__(self), self.mentor, self.started)
#!/usr/bin/env python

# Author: Steven Ludtke, 01/03/07 (sludtke@bcm.edu)
# Copyright (c) 2000-2007 Baylor College of Medicine
#

import os

def main():
    a=mine()
    print a.f(5)
    a.mult=10.0
    print a.f(5)

class mine:
    "This is my class"
    def f(self,x):
        try : return x*self.mult
        except : return 0

if __name__ == "__main__":
    main()