6.00 Handout, Lecture 16
Not intended to make sense outside of lecture

```
class Person(object):
    def __init__(self, family_name, first_name):
        self.family_name = family_name
        self.first_name = first_name
    def familyName(self):
        return self.family_name
    def firstName(self):
        return self.first_name
    def __cmp__(self, other):
        return cmp((self.family_name, self.first_name),
                   (other.family_name, other.first_name))
    def __str__(self):
        return '<Person: %s %s>'%(self.first_name, self.family_name)
    def say(self,toWhom,something):
        return self.first_name + ' ' + self.family_name + ' says to ' +
toWhom.firstName() + ' ' + toWhom.familyName() + ': ' + something
    def sing(self,toWhom,something):
        return self.say(toWhom,something + ' tra la la')

class MITPerson(Person):
    nextIdNum = 0
    def __init__(self, familyName, firstName):
        Person.__init__(self, familyName, firstName)
        self.idNum = MITPerson.nextIdNum
        MITPerson.nextIdNum += 1
    def getIdNum(self):
        return self.idNum
    def __str__(self):
        return '<MIT Person: %s %s>'%(self.first_name, self.family_name)
    def __cmp__(self,other):
        return cmp(self.idNum, other.idNum)

##p1 = MITPerson('Smith','Fred')
##p2 = MITPerson('Foobar','Jane')
##print p1.getIdNum()
##print p2.getIdNum()

class UG(MITPerson):
    def __init__(self, familyName, firstName):
        MITPerson.__init__(self, familyName, firstName)
        self.year = None
    def setYear(self, year):
        if year > 5: raise OverflowError('Too many')
        self.year = year
    def getYear(self):
        return self.year
    def say(self,toWhom,something):
        return MITPerson.say(self,toWhom,'Excuse me, but ' + something)
##
##me = Person("Grimson", "Eric")
##ug = UG('Doe', 'Jane')


class Prof(MITPerson):
    def __init__(self, familyName, firstName, rank):
        MITPerson.__init__(self, familyName, firstName)
        self.rank = rank
        self.teaching = {}
    def addTeaching(self, term, subj):
        try:
            self.teaching[term].append(subj)
        except KeyError:
                self.teaching[term] = [subj]
    def getTeaching(self, term):
        try:
            return self.teaching[term]
        except KeyError:
            return None
    def lecture(self,toWhom,something):
        return self.say(toWhom,something + ' as it is obvious')
    def say(self,toWhom,something):
        if type(toWhom) == UG:
            return MITPerson.say(self,toWhom,'I do not understand why you say ' +
something)
        elif type(toWhom) == Prof:
            return MITPerson.say(self,toWhom,'I really liked your paper on ' +
something)
        else:
            return self.lecture(something)


##me = Prof('Grimson', 'Eric', 'Full')
##me.addTeaching('F08', '6.00')
##me.addTeaching('S09', '6.00')
##me.addTeaching('S09', '6.xxx')
##print me.getTeaching('F08')
##print me.getTeaching('S09')
##print me.getTeaching('S08')
##print me.teaching

class Faculty(object):
    def __init__(self):
        self.names = []
        self.IDs = []
        self.members = []
        self.place = None
    def add(self,who):
        if type(who)!= Prof: raise TypeError('not a professor')
        if who.getIdNum() in self.IDs: raise ValueError('duplicate ID')
        self.names.append(who.familyName())
        self.IDs.append(who.getIdNum())
        self.members.append(who)
    def __iter__(self):
        self.place = 0
        return self
    def next(self):
        if self.place >= len(self.names):
            raise StopIteration
        self.place += 1
        return self.members[self.place-1]


##grimson = Prof('Grimson','Eric', 'Full')
##lozano = Prof('Lozano-Perez', 'Tomas', 'Full')
##guttag = Prof('Guttag', 'John', 'Full')
##barzilay = Prof('Barzilay', 'Regina', 'Associate')
##course6 = Faculty()
##course6.add(grimson)
##course6.add(lozano)
##course6.add(guttag)
##course6.add(barzilay)
##
##for p in course6:
```

```
##    print p.familyName()
##
##
##print ug.say(grimson,'I do not understand')
##print grimson.say(ug,'you do not understand')
##print grimson.say(guttag,'why the sky is blue')
##
##print ug.sing(ug,'I think I finally understand')
```