

6.00 Handout, Lecture 13
(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 __gt__(self, other):
        return self.family_name + self.first_name\
            > other.family_name + other.first_name
    def __str__(self):
        return self.first_name + ' ' + self.family_name
```

```
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 __gt__(self, other):
        return self.idNum > other.idNum
    def __eq__(self, other):
        return str(self) == str(other)
```

```
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()
```

```
class G(MITPerson):
    pass
```

```
class CourseList(object):
    def __init__(self, number):
        self.number = number
        self.students = []
    def addStudent(self, who):
        if type(who) != UG and type(who) != G:
            raise TypeError('Not a student')
        if who in self.students:
            raise ValueError('Duplicate student')
        self.students.append(who)
    def __iter__(self):
        self.place = 0
        return self
    def next(self):
        if self.place >= len(self.students):
            raise StopIteration
        self.place += 1
        return self.students[self.place-1]
```