

**6.00 Handout, Lecture 15**  
**(Not intended to make sense outside of lecture)**

```
class Person:
    def __init__(self, family_name, first_name):
        self.family_name = family_name
        self.first_name = first_name
        self.birthday = None
    def familyName(self):
        return self.family_name
    def setBDay(self, date):
        self.birthday = date
    def getBDay(self):
        return self.birthday
    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)
```

```
fNames = {0: 'John', 1: 'Alice', 2: 'Olga', 3: 'Chi', 4: 'Estevan'}
lNames = {0: 'Jon', 1: 'Liu', 2: 'Jones', 3: 'Wong', 4: 'Ramirez'}
pList = []
for i in range(0, 10):
    fIdx = random.randint(0, 4)
    lIdx = random.randint(0, 4)
    p = Person(lNames[fIdx], fNames[lIdx])
    pList.append(p)
for p in pList: print p

pList.sort()
print 'Sorted list'
for p in pList: print p
```

```
class MITPerson(Person):
    nextIdNum = 0
    def __init__(self, family_name, first_name):
        self.family_name = family_name
        self.first_name = first_name
        self.idNum = MITPerson.nextIdNum
        MITPerson.nextIdNum += 1
        self.birthday = None
    def getIdNum(self):
        return self.idNum
    def __str__(self):
        return '<MIT Person: %s %s>'%(self.first_name, self.family_name)
```

```
class UG(MITPerson):
    """year is an integer between 1 and 7"""
    def setYear(self, year):
        self.year = year
    def __str__(self):
        return '<UG: %s %s>'%(self.first_name, self.family_name)
```

```
class Prof(MITPerson):
    def addTeaching(self, term, course):
        try:
            self.teaching[term].append(course)
        except KeyError:
            self.teaching[term] = [course]
        except AttributeError:
            self.teaching = {}
            self.teaching[term] = [course]
    def getTeaching(self, term):
        return self.teaching[term]
    def __str__(self):
        return '<Professor: %s %s>%(self.first_name, self.family_name)
```

```
me.addTeaching('F07', '6.00') #Should fail

me = Prof('Guttag', 'John')
me.addTeaching('F07', '6.00')
```