1 Administrative Topics

• I return graded quizzes.
• Any questions about the project?

2 Memory Model of Inheritance

Our goal today is to look at the memory model for the code in Monday’s notes. Here is a recap of it.

First, we have a Student class.

class Student:
    def __init__( self, name, id ):
        self.name = name
        self.id = id
        self.grades = []

    # accessors
    def getName( self ):
        return self.name

    def getId( self ):
        return self.id
```python
def getGPA( self ):
    total = 0.0
    for grade in self.grades:
        total += grade[1]
        # total = total + grade[1]
    return total / len(self.grades)

# mutators
def addGrade( self, grade ):
    self.grades.append( grade )
```

Then we have a MatriculatedStudent. A MatriculatedStudent is a Student with a year of expected graduation.

```python
class MatriculatedStudent(Student):
    def __init__( self, name, id, year ):
        Student.__init__( self, name, id )
        self.year = year

    def getYear():
        return self.year

    def setYear( self, year ):
        self.year = year
```

And finally, we have test code.

```python
if __name__ == '__main__':
    s = MatriculatedStudent( "Fred Firstyear", 897987, 2013 )
    s.addGrade( ('CS151', 2.3) )
    s.addGrade( ('BI163', 2.5) )
    s.addGrade( ('HI140', 3.3) )
    s.addGrade( ('MA121', 3.1) )
    print s.getName()
    print s.getGPA()
```

First, I show the symbol tables of just the two classes. The subclass (MatriculatedStudent) inherits the getName, getId, getGPA, and addGrade methods from the Student class. We indicate that by having entries for each of those methods in the MatriculatedStudent class. The values are arrows to the corresponding entries in the symbol table for the Student class (Figure 1).

Then we run the test code, which creates a MatriculatedStudent object.
Figure 1: Symbol tables for Student class and MatriculatedStudent class.
Figure 2 shows the object’s symbol table, the Student class symbol table and the MatriculatedStudent class symbol table.

Figure 2: MatriculatedStudent object created by the test code.
3 Parameters to __init__

When we design the class, we decide who should have control over the initial values:

- The user/caller/main code must determine it. Then we include it as a parameter, and force the user to supply a value. We use that value to initialize the field.
  ```python
def __init__(self, name):
    self.name = name
  ```

- The user has the option of supplying the value. We include it as a parameter, but provide a default value, so the user can include or exclude a name argument.
  ```python
def __init__(self, name='George'):
    self.name = name
  ```

- We mandate the initial name is some specific value (e.g. 'Milo'). We exclude it as a parameter, so the user has no ability to set the value.
  ```python
def __init__(self):
    self.name = 'Milo'
  ```

We use the first option for a Student’s name. We could use the second option for the year if we are creating a bunch of MatriculatedStudents at once, and most are going to graduate the same year. We use the third option for a Student’s grades (the list must start empty).
4 Dictionaries

I need to tell you about a very neat data structure that Python supports, that is similar to a list, but doesn’t require the indexing to use numbers. It is called a dictionary and it maps “keys” to values.

Suppose we want to build a dictionary mapping names to favorite color.

<p>| | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>George</td>
<td>black</td>
</tr>
<tr>
<td>Stephanie</td>
<td>purple</td>
</tr>
<tr>
<td>Scott</td>
<td>green</td>
</tr>
<tr>
<td>Milo</td>
<td>blue</td>
</tr>
</tbody>
</table>

Here is the code to do it:

```python
# make empty dictionary
d = {}
# 'George' => 'black'
d[ 'George' ] = 'black'
# 'Stephanie' => 'purple'
d[ 'Stephanie' ] = 'purple'
# 'Scott' => 'green'
d[ 'Scott' ] = 'green'
# 'Milo' => 'blue'
d[ 'Milo' ] = 'blue'
```

We can explicitly test to see if certain values are in the dictionary. And we can access values.

```python
# Is there a color for Linda?
print 'Linda' in d

# Is there a color for George?
print 'George' in d

# What is George’s favorite color?
print d[ 'George' ]
```