CS 61A Challenge Problems: Object Oriented Programming

Solutions at http://alextseng.net/teaching/csm61a/

Alex Tseng

1 Defining a Class

We define a new class to create objects that represent high school students. A basic high school student has a name, a grade, and a favorite subject.

(a) Fill in the functions below to complete the class definition so that the doctests pass.

```
class Student:
    students_enrolled = 0
    student_years = {}
    def __init__(                      ):









    def set_favorite_subject(                      ):




    def students_in_grade(                      ):




>>> tiffany = Student("Tiffany", 9)  # a student by default has no favorite subject
>>> tiffany.name
"Tiffany"
>>> tiffany.grade
9

>>> mike = Student("Michael", 11)
>>> mike.set_favorite_subject("Biology")
>>> mike.fave_subject
"Biology"
>>> tiffany.fave_subject
"No favorite"

>>> Student.students_in_grade(9)
1
>>> Student.students_in_grade(10)
0

>>> zack = Student("Zackary", 11)
>>> Student.students_in_grade(11)
2
```

```
>>> Student.students_enrolled
3
```

(b) What would happen if we called `mike.students_enrolled`?

(c) What would happen if we called `Student.student_years[9]`?

(d) High school students also get crushes on each other. A student can have a crush on any other student (or none at all). Many students may have a crush on the same person, but a student can only have a crush on one person. Write a bound method `develop_crush` that stores a student's crush as an instance attribute `crush`. By default, a student has no crush (`crush` is `None`). What change would you need to make to `__init__`?

(e) *Challenge* Now we also wish for the *class* to know how popular certain students are. That is, we want to know how many people have crushes on each student. You will need a class variable to store how many crushes each person is the receiver of. You may also need to alter `__init__` and `develop_crush`.

(f) Now that you've implemented (e), write a class method `crush_on` that is able to find the number of people who have a crush on some student, by name. For example, if `tiffany` and `zack` both have a crush on `mike`, then `Student.crush_on("Michael")` should return 2. What happens if we call it on a student that doesn't exist?

2 Special Methods

Python actually has many special methods, most of which are used quite rarely. We will explore the uses of several of the more useful ones. Consider the following basic definition of molecules:

```
class Molecule:
    def __init__(self, formula, name, weight):
        self.formula = formula
        self.name = name
        self.weight = weight
```

(a) Two molecules are equivalent if and only if they have the same name (we can't use the formula because constitutional isomers have the same formula but can have very different structures!). Add a bound method so that the following doctests pass. Recall that `__eq__(self, other)` returns a boolean allows equality testing using `==`.

```
>>> ethanol = Molecule("C2H6O", "Ethanol", 46.07)
>>> alcohol = Molecule("C2H6O", "Ethanol", 46.07)
>>> dimethyl_ether = Molecule("C2H6O", "Dimethyl Ether", 46.07)
>>> ethanol == alcohol
True
>>> ethanol == dimethyl_ether
False
```

(b) We can also compare molecules using molecular weights with `<` and `>`. Hint: `__lt__` and `__gt__` work just the same as `__eq__`.

```
>>> ammonia = Molecule("NH3", "Ammonia", 17.03)
>>> ammonia < ethanol
True
>>> ammonia > dimethyl_ether
False
```

(c) Consider the following definitions of `__str__` and `__repr__`. What would Python print?

```
def __str__(self):
    return self.formula + ": " + self.name
def __repr__(self):
    return "Molecule('{0}', '{1}', {2})".format(self.formula, self.name, self.weight)

>>> print(ammonia)

>>> ethanol

>>> str(alcohol)

>>> repr(ammonia)

>>> eval(repr(ethanol))  # eval evaluates a string as native Python

>>> print(repr(ethanol))
```