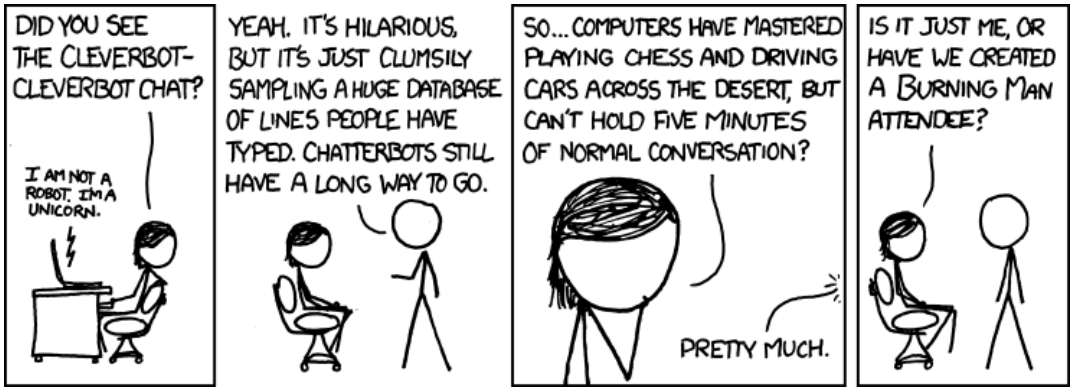
CS30 - Assignment 7

Part A: Due Tuesday March 22, at 11:59pm

Part B: Due Friday March 25, at 6pm



http://xkcd.com/948/

Before Spring Break, we began to discuss classes. For this assignment, you're going to be writing your own class! Read through the entire handout before you start coding. This assignment involves two parts: 1) designing your class and then 2) implementing the class and using it to write a small program.

Specifications

You are to design and implement a class of some kind - like the "Person" class (or Rectangle, Stack and Queue). Your class should include the following pieces:

- a constructor

  `__init__`

- an str function

  `__str__`

- a method with at least two optional arguments (in combination with one or more regular arguments)
- 4 additional methods inside the class which meet the following constraints:

  - at least one must be a mutator method and at least one an accessor method
  - at least two of these methods must be nontrivial methods: 6 lines of code or more
  - at least one of these methods must involve a loop
  - at least one of these functions should involve reading/writing something from/to a file. These are both concepts that we will cover in class AFTER spring break.
- 3 or more instance variables (e.g. `self.name` is an instance variable for the `Person` class).

In addition to writing the class, you MUST demonstrate how it works with several lines of code which create at least two instances of the class and call its methods (including various calls illustrating how the optional arguments work). Include this demonstration at the END of your submission file. For example, see Appendix B for a small example program using the `Person` class. Yours will be slightly more involved since you class will have more methods, but this should give you the basic idea.

Some ideas for the type of class you could write are:

- cow
- vending machine
- car
- college
- student

Part A: Planning and design

For the first part of the assignment, you are to figure out the *design* of your program. Specifically, you should write:

- The name of the class.
- The instance variables you plan to have.
- The methods that your class will have, including the parameters they will take.
- Docstrings for the class and for each method describing what they do.
- One or two sentences stating what your program will do that uses the class.

When thinking about the design of the class, think about how you're going to use it. For example, in `Person` class, the main motivator was for people to be able to wear clothes (well, shirts) and change their clothes throughout the week. Think about a collection of things and what you'd like to make them do.

You should write your design in Wing as if you were writing the class, but leave off the actual bodies of the methods. If you use the keyword `pass` as the body of the method, Wing will still do the proper indenting. Appendix A shows a sample design for the `Person` class.

Submission:

Submit Part A by the deadline in a file named with your first name and last name followed by `assign7A.py`

Part B: my first "class"

Implement your class and program! As you implement it, you may realize that you want to change your design. That's fine, just make sure that your final submission meets the requirements laid out in the specifications section.

When you're done

Put your class and program using the class in a file named with your first name and last name followed by `assign7B.py`. Your program should run without having to make any method calls, i.e. it should run when you press the green arrow in Wing.

- You should have comments at the very beginning of the file stating your name, course, assignment number and the date.
- Each function should have an appropriate docstring.
- Your class should have an appropriate docstring.
- Include other miscellaneous comments to make things clear.

If your program requires reading from a file, include an example file with your submission. To do this, put your .py file and the example file into a folder with your name on it and create a .zip file. Submit this .zip file.

Submit your .py file online using the courses submission mechanism.

Grading

		points
`Part A`		
	properly formatted and commented	2
	`__init__` and `__str__`	1
	optional parameter method	1
	3 instance variables	1
	4 other methods	1
	program descriptions	1
`Part B`		
	`__init__`	2
	`__str__`	1
	optional parameter method	3
	have 4 methods total	3
	2 or more nontrivial methods	3
	method with loop	1
	method that does I/O	3
	program	4
comments/style		3
total		30

Appendix A

```
class Person:
    """ Class to represent a person """

    # will have two instance variables:
    # self.name: stores the persons name
    # self.shirt_color: stores the color of the shirt the person is wearing

    def __init__(self, persons_name, shirt_color = "blue"):
        """ create a new person named persons_name wearing shirt_colored shirt """
        pass

    def get_shirt_color(self):
        """ get the color of the shirt this person is wearing """
        pass

    def get_name(self):
        """ get the name of this person """
        pass

    def change_shirt(self):
        """ randomly change the shirt the person is wearing """
        pass

    def __str__(self):
        pass

# The program will generate two different people.  It will then run through a normal
# week (7 days) changing their shirt at the end of each day.  If the shirts match, it
# will print this fact out.
```

Appendix B

Sample program using the `Person` class:

```
p1 = Person("Steve")
p2 = Person("Amy", "green")

for i in range(1,8):
    print "-" * 10
    print "Day " + str(i)
    print p1
    print p2

    if p1.get_shirt_color() == p2.get_shirt_color():
        print p1.get_name() + " and " + p2.get_name() + \
              " are wearing the same shirt color!"

    # print a blank line
    print

    p1.change_shirt()
    p2.change_shirt()
```

and here is the output from a sample run:

```
----------
Day 1
Steve, wearing a blue shirt
Amy, wearing a green shirt

----------
Day 2
Steve, wearing a red shirt
Amy, wearing a blue shirt

----------
Day 3
Steve, wearing a funny shirt
Amy, wearing a red shirt

----------
Day 4
Steve, wearing a red shirt
Amy, wearing a red shirt
```

```
Steve and Amy are wearing the same shirt color!

----------
Day 5
Steve, wearing a red shirt
Amy, wearing a blue shirt

----------
Day 6
Steve, wearing a blue shirt
Amy, wearing a funny shirt

----------
Day 7
Steve, wearing a green shirt
Amy, wearing a green shirt
Steve and Amy are wearing the same shirt color!
```