6.149 Checkoff 7

http://web.mit.edu/6.149/www/materials.html

Due: Wednesday, January 28 at 5 p.m.

7.0 MITx Exercises

Complete Lectures 9-14 on MITx by 11:59 p.m. on Friday, January 30. We suggest you space these exercises out for practice.

7.1 More object oriented programming

For the following questions, write your answers in the spaces provided.

1. Consider the following code:

```
class Clock(object):

    def __init__(self, time):
        self.time = time

    def print_time(self):
        time = '6:30'
        print self.time

clock = Clock('5:30')
clock.print_time()
```

   (a) What does the code print out? Guess first, and then create a Python file and run it.

(b) Why does the code print this?

2. Consider the following code:

```
class Clock(object):

    def __init__(self, time):
        self.time = time

    def print_time(self, time):
        print time

clock = Clock('5:30')
clock.print_time('10:30')
```

   (a) What does the code print out? Guess first, and then create a Python file and run it.

   (b) What does this tell you about giving parameters the same name as object attributes?

3. Consider the following code:

```
class Clock(object):

    def __init__(self, time):
       self.time = time

    def print_time(self):
       print self.time


boston_clock = Clock('5:30')
paris_clock = boston_clock
paris_clock.time = '10:30'
boston_clock.print_time()
```

   (a) What does the code print out? Guess first, and then create a Python file and run it.

   (b) Why does it print what it does? (Are `boston_clock` and `paris_clock` different objects? Why or why not?)

Exercise 7.2 Inheritance

This problem is brand new, so please let us know if you spot any bugs or need additional clarifications!

Create a new file, `checkoff7_inheritance.py`

In this problem, you will create a class `Phone` and two subclasses, `CellPhone` and `SmartPhone`.

(a) Define a class `Phone`.

i. The class should have one attribute, `self.number`, the phone number stored as a string. The number may include symbols, such as dashes and parentheses.

ii. Add a getter method, `get_number`, that returns the phone number.

iii. Add a `call` method that takes an input string, `call_number`, and prints "Calling ", followed by `call_number`. You may assume that all string inputs for phone numbers will be valid phone numbers.

For example, the `Phone` class should have the following behavior in Python Shell:

```
>>> phone = Phone("123-456-7890")
>>> print phone.get_number()
123-456-7890
>>> phone.call("222-345-6565")
Calling 222-345-6565
```

(b) Define `CellPhone`, which is a subclass of `Phone`. In addition to having the attribute `self.number`, `CellPhone` should also have:

i. the attribute `self.sent_texts`, which should be initialized to an empty list.

ii. the attribute `self.received_texts`, which should be initialized to an empty list.

iii. the attribute `self.contacts`, which should be initialized to an empty dictionary.

iv. a method called `add_contact`, which takes as input a string `name` and string `contact_number`. `add_contact` adds an entry to the `contacts` dictionary with the name as the key and the number as the value.

v. a method called `receive_text`, which takes as input three strings: `name`, `number`, and `message`. If `name` is not in `self.contacts`, add the name and number. (Hint: How can you reuse code you've already written to perform this operation?) Finally, add `message` to `received_texts`.

vi. a method called `send_text`, which takes as input `name` (a string), `number` (a string), and `message` (also a string). If the name cannot be found in `contacts`, add the name and number. The sent message should be added to `sent_texts`.

vii. a method called `get_sent_texts` that returns `self.sent_texts`.

viii. a method called `get_received_texts` that returns `self.received_texts`.

ix. a method called `get_contacts` that returns `contacts`.

The subclass should exhibit the following behavior in Python Shell:

```
>>> phone = CellPhone("123-456-7890")
>>> print phone.get_contacts()
{}
>>> phone.add_contact("Michelle Szucs", "222-343-5656")
>>> print phone.get_contacts()
{"Michelle Szucs": "222-343-5656"}
```

(c) Write a function, `text`, that takes as input three strings (`from_phone_number`, `to_phone_number`, and `message`) and a list of CellPhones, `phone_list`. Send a text from `from_phone` to `to_phone` using the methods defined in part b. You may assume phone numbers are unique.

(d) Test `text` by creating at least five phones and sending at least two different texts. Use the getter methods to print out the sent and received texts for each phone.