Advanced Python Exam 3

Spring 2017

1. Do all work on this exam, using backs of pages and the extra pages for extra space for answers or scratch
2. Closed book, no notes or electronic devices of any kind
3. Print your name on each page of the exam
4. No speaking except to the TA or the Instructor

Name (Print legibly): Matthew Kramer

USF ID #: U20891900

Leave the box below blank; your score will entered there

Score:

Adv. Python S17 Exam 2 **Name** Matthew Kramer

1. (a) [13 points] Complete the definition of the generator function **`oddsquares()`** that will generate the squares of all positive odd integers.

```
def oddsquares():
    x=1
    while(true):
        yield x**2
        x+=2
```

(b) [12 points] You are to write a decorator **`map_generator(f)`** which turns a function `f` into a generator that produces f(1), f(2) , ... This should be a straightforward modification of your answer to (a).

```
def map_generator(f):
    x=1
    while(true):
        yield f(x)
        x+=1
```

Adv. Python S17 Exam 2 **Name** Mathew Kramer

3. [20 points] Fill in the missing parts of the definition of the memoize decorator

```
def memoize(f):

    memo = {}

    def helper(x):
        if x not in memo:
            memo[x] = f(x)
        return memo[x]



    return helper
```

Adv. Python S17 Exam 2 **Name** Matthew Kramer

3. [15 points] Define a class **Person** with two visible attributes: a string **name** and integer **id**. **id** must be a read-only property. The methods should include the **__init__** method, the **__str__** and **__repr__** methods, and methods that will allow two **Person** objects to be tested for equality and for "less than". For the latter two, the comparison should be based on the **Person**'s **name** and **id**, with the **name** checked before the **id**.

```
class Person:
    def __init__(self, name, id):
        self.name = name
        self._id = id

    @property
    def id(self):
        return self._id

    def __str__(self):
        return ("{0.name!r}, {0.id!r}".format(self))

    def __repr__(self):
        return ("Person({0.name!r}, {0.id!r})".format(self)

    def __eq__(self, other):
        return self.name == other.name and self._id == other._id

    def __lt__(self, other):
        return self.name < other.name and self._id < other._id
```

Adv. Python S17 Exam 2 **Name** Matthew Kramer

(b) [15 points] Define subclass **`Student`** of **`Person`**. In addition to the inherited Person attributes, A **`Student`** object should have two additional attributes: **`major`** (a string) and **`class`** , also a string. Use properties to ensure that the value for class must be one of the following: `freshman`, `sophomore`, `junior`, `senior`.

The class must have **`__init__`**, **`__str__`** and **`__repr__`** methods. Use must use inheritance where possible.

```
class Student(Person):
    def __init__(self, name, id, major, classs):
        super().__init__(name, id)
        self.major = major
        self.classs = classs

    @property
    def classs(self):
        return self.classs

    @classs.setter
    def classs(self, classs):
        assert classs in ["freshman", "sophomore", "junior", "senior"]
        self.classs = classs

    def __str__(self):
        return ("{0.name!r}, {0.id!r}, {0.major!r}, {0.classs!r}"
                .format(self))

    def __repr__(self):
        return ("student({0.name!r}, {0.id!r}, {0.major!r},
                {0.classs!r})".format(self))
```

Adv. Python S17 Exam 2 **Name** Matthew Kramer

4. (a) [5 points] The immutable version of `list` is `tuple`. What `type` is the immutable version of `set`?

frozenset

(b) [15 points] Using the `set` type, write a single line statement that will produce a list `L` of the unique values of an arbitrary list `K`.

Example: if `K = [ 8, 2, 3, 8, 2, 6, 3]`, then `L` could be `[2, 8, 3, 6]`.

`L =` list(set(K))

(c) [5 points] Assuming the above, fill in the sorting key to rearrange the values in L so that they in the order they first appear in K. In the above example, the sorted version of L would be [8, 2, 3, 6].

`L.sort(key =` lambda x: K.index(x) `)`