CSC108: Review Exercises

Daniel Zingaro
University of Toronto
daniel.zingaro@utoronto.ca

December 2012

1 Fall 2010 Exam: Dictionary Difference

Define the **dictionary difference** between two dictionaries to be a new dictionary containing keys (and their values) that occur in one but not both of the dictionaries. If any key is in both original dictionaries, it does not appear in the new dictionary.

For example, if we have these two dictionaries:

```
d1 = {1: 'a', 2: 'b', 3: 'c', 9: 'h'}
d2 = {1: 'k', 6: 'f', 2: 'g', 5: 'e', 8: 'c'}
```

their difference is `{3: 'c', 9: 'h', 6: 'f', 5: 'e', 8: 'c'}`.

Write the following function according to the docstring and the definition above.

```
def dict_diff(d1, d2):
    '''d1 and d2 are dicts. Return a new dict which is the dictionary
    difference between d1 and d2.'''
```

2 Fall 2010 Exam: Reading MCQs

A **question bank** is a text file consisting of one or more multiple choice questions. Each multiple choice question consists of the following lines, in order:

- A line containing only the characters `MC`
- One line, containing the question
- One or more lines, containing possible answers to the question

For example, here is a question bank with two questions. Notice that the number of answers to each question may be different.

```
MC
The instructor that has taught CSC108 the most is:
Diane
Dan
Steve
MC
The funniest CSC108 instructor is:
Dan
Daniel
Dan Z
Daniel Z
```

We want a function that will read such a file and produce a list of strings, each of which contains the full text of one of the multiple-choice questions (including the answers). For example, the list produced from the file above would be:

```
['The instructor that has taught CSC108 the most is:\nDiane\nDan\nSteve\n',
'The funniest CSC108 instructor is:\nDan\nDaniel\nDan Z\nDaniel Z\n']
```

(Notice that the newlines are preserved.) Such a list could be used to choose random questions and generate a test, although you will not be writing any such code today.

On the following page, write function `read_mcqs` according to its docstring.

```
def read_mcqs(f):
  '''f is an open question bank file with n >= 1 multiple choice questions.
  Return a list of n strings, each of which contains the entire question
  text and responses for one question.'''
```

2.1 Strategy

A for-loop on a file reads every line of the file. There is no way to have a for-loop stop when it finds an `MC` line, for example. But, inside of the for-loop, you could detect the start of a new question, and then add the completed question to your list of question strings.

Another way to solve this is by using two nested while-loops. The outer while-loop detects end-of-file, and the inner loop reads everything until it hits end-of-file or the next `MC`. This way, when the inner loop finishes, it has collected all of the current question into a string, and it can be added to the growing list of questions.

Try it!

3 Summer 2008 Exam: Column Sums

Consider text files where each line contains one or more integers. For each line, the first integer is at position 0, the second is at position 1, the third is at position 2, and so on. A **column of data** is defined as all of the integers in the file with the same position. Column 0 therefore represents each integer at position 0, column 1 represents each integer at position 1, and so on. For example, in this file:

```
2 4
6
10 15 22 30
```

column 0 is:

```
2 6 10
```

column 1 is:

```
4 15
```

column 2 is:

```
22
```

and column 3 is:

```
30
```

On the following page, write a function that takes an open file object (not a filename!) referring to a file of this format, and returns a list of the **sums** of each column. For the sample file above, your function would return:

```
[18, 19, 22, 30]
```

```
def column_sums (f):
  '''Return the list of column sums from open file f.'''
```

There are at least two completely different ways to solve this function. The first is to use dictionaries.

It doesn't seem that dictionaries make much sense here. After all, you want to return the column sums in order, and you can't rely on dictionaries keeping the sums in order. But we know how to sort the keys of a dictionary. If the keys are positions, and the values are the sum of each position, then we can sort the keys to get our sums in order.

The function can also be written using only lists, no dictionaries. It's slightly tricky because of this:

```
>>> lst = [4, 8, 12]
>>> lst[3] = 16
Traceback (most recent call last):
File "<stdin>", line 1, in <module>
IndexError: list assignment index out of range
```

That is, you can't just assign to the next index like you can do with a dictionary.

4 Fall 2010 Exam: Classes

Consider the following class:

```
class Member(object):
    '''A member of facebook, with a str name, a str status, and zero or
    more str friends.'''

    def __init__(self, s):
        '''A new member with name s, no friends, and status "no status yet".'''

        self.name = s
        self.friends = []
        self.status = "no status yet"

    def __str__(self):
        '''Return a str describing this member.'''

        result = "Name: %s; Status: %s" % (self.name, self.status)
        if self.friends == []:
            result += "; no friends yet"
        else:
            result += ", and friends: "
            for person in self.friends:
                result += person + ", "
            # Strip off the final comma.
            result = result[:-2]
        return result

    def add_friend(self, friend):
        '''Add str friend as a friend of this member.'''

        self.friends.append(friend)

    def update_status(self, new_status):
        '''Update this member's status to str new_status.'''

        self.status = new_status
```

A. Create a `Member` variable called `m1`. His name is “Danny Z”, his status is “torturing 108 students” and he has one friend named “Diane”.

B. Write the following new method, to be added to the class.

```
def friendliness(self):
    '''Return the number of friends this member has.'''
```

C. Write the `__lt__` method below, also to be added to the class. Define it so that if we call `sort` on a list of members, they will come out in order from least friendly (having the fewest friends) to most friendly (having the most friends).

```
def __lt__(self, other):
```

5 Fall 2011 Exam: Minimum Value

Consider the following function:

```
def min_value(L):
    '''L is a list of ints that are >= -1.  Return the minimum value in L that
    is > -1.  If L doesn't have any value in it other than -1, return -1.'''
```

Suppose that we want to test `min_value`. Describe three test cases that each test different "categories" of inputs. To describe each test case, give the `list` that you would pass to `min_value`, the return value you expect, and the purpose of the test case. Do **not** write any code. We have given you one test case as an example; add three more.

Value of `L`	Return value	Purpose of this test case
`[]`	`-1`	empty list

Now write the function according to its docstring specification.