ENGI E1006 HW4 (Chapters 6, 7, 9)

Assignment Due on 11/17/2016 at 11:59PM

November 6, 2016

Academic Honesty Policy

You are expected to submit your own work. Assignments are to be completed individually, NOT in groups. No collaboration is permitted, unless otherwise specified. Please do not include any of your code snippets or algorithms in public Piazza posts. You cannot not use solutions from any external source. If you have any questions at all about this policy please ask the course staff before submitting an assignment.

For this, and all future homeworks, we'll have one main Piazza post for the homework overall, as well as more specific posts for each homework question.

General questions should be asked as follow-up discussions beneath the main Piazza post. Please note that we will be modifying this main Piazza post with clarifications on the homework where necessary, and you will be responsible for adhering to them.

If you have a question, please post a follow-up discussion on the appropriate individual-question post. Any relevant discussions will be incorporated as a clarification in the main post.

For Section A

This portion of Homework 4 is optionally collaborative. You may work in groups of up to 2 currently enrolled ENGI E1006 students (yourself included). Each student should submit their own work individually as hw4_section_A.txt. Include the name and UNI of your group mate, if any, at the top of your txt file.

Although it is a collaborative work, you must be actively involved in every part of your solution, and you may never simply copy any answer from anyone.

For Section B

A skeleton of each function has been provided for you. DO NOT modify the function signatures or return variables for any reason. Your job is to complete the body of each function. We've included test cases along with expected output so you can check your solutions. These test cases are under the line:

```
if __name__ == "__main__":
```

You can modify these test cases within the skeleton code.

Use Python 3.5 for all Python parts. To check your Python version, enter "python –version" (no quotes, two hyphens) in a terminal.

Make a comment at the top of each file explaining the high-level logic of your code. Make in-line comments detailing specific decisions and logic where necessary.

You will be graded on your code style. Please refer to the Python style guide available on Courseworks in the resources folder.

Download the skeletons from Canvas and keep the filenames the same:

```
matrix_operations.py
is_anagram.py
is_permutation.py
create_FOF.py
registration_system.py
game_of_life.py
```

For submitting, put all files into a single folder named UNI_hw4. Compress your UNI_hw4 folder into a zip file named UNI_hw4.zip (eg. abc1234_hw4.zip). Make sure to use an underline, NOT a hyphen. Submit that zip file via Courseworks.

Section A (Collaborative)

The Section A of this homework is optionally collaborative. This section is designed to help you review the important details of list, tuples, dictionaries and sets, and more specifically, help you better prepare for the upcoming midterm. We mimicked the type of problems you may expect to see in a timed exam (though we have no guarantee of similarity neither in terms of format nor the difficulty). Because of that, questions in this section are more conceptual and less computationally-hard in nature.

For many of the questions, you can easily get the answer by running the code in Python console, which is why there aren't many points assigned to these questions. You're also allowed to collaborate on the questions in this section so that you can reinforce the ideas behind the questions with a study partner. The hope is that these questions will encourage you to fully clarify the concepts for yourself before the next midterm.

Read Section 7.6 of the textbook. Answer the following questions:

1. (0.5 pts each) True/False:
   (a) `(5)==(5,)` returns True.
   (b) Any of the following: (integer, string, list) can be the key of a dictionary.
   (c) Any of the following: (integer, string, list) can be the value of a dictionary.
   (d) Any of the following: (integer, string, list) can be in a set.
   (e) `s[0]="a"` would result in an error if `s` is a string.
   (f) If `a = [(1,2),(3,4)]` performing `a[0]=3` would result in an error.
   (g) If a key `k` is not in a dictionary `d`, then `d[k]` will return None.
   (h) If `((x==y)!=(x is y))` is False, then x is an alias of y.
   (i) If you were to open and read (so in "r" mode) a file that does not exist, Python will error out.
2. (0.5 pts each) For each of the sub-question, answer what `A is B` and what `A == B` will return respectively:
   (a)
   ```
   A = [1,2,3]
   B = [1,2,3]
   ```
   (b)
   ```
   A = [1,2,3]
   B = A
   ```
   (c)
   ```
   A = [1,2,3]
   B = A[:]
   ```
   (d)
   ```
   A = ['a','b','c']
   B = ",".join(A).split(",")
   ```
   (e)
   ```
   A = (1,2,3) #note that A is a tuple
   B = A
   ```

(f)
```
A = [1,2,3]
def f(A):
    B = A
    return B
B = f(A)
```

3. (1 pts each) Code Tracing. Print the output for each of the following:

(a)
```
A = [1,2,3]
B = A
B[0] = 4
print(A)
print(B)
```

(b)
```
A = [1,2,3]
def f(A):
    A.append(4)
    A += [5]
    A = A + [6]
    return A
B = f(A)
B[1] = 7
print(A)
print(B)
```

(c)
```
A = [[0]]*10
A[0][0] = 1
B = []
for i in range(10):
    B.append([0])
B[0][0] = 1
print(A)
print(B)
```

(d)
```
import copy
A = [1,2,3]
B = [1,2,(4,5)]
A += B
C = copy.deepcopy(A)
B[0] = 3
B[2] = B[2] + (6,)
C[-1] += (7,)
print(A)
print(B)
print(C)
```

(e)
```
import copy
def f(A):
    B = copy.copy(A)
    C = A[0:len(A)]
    D = A
    A[0] += 1
```

```
        B[0] += 2
        C[0] = 3
        D.append(4)
        print(A,B,C,D)
    A = [5]
    print(f(A))
    print(A)
```

(f)
```
    import copy
    def f(A):
        S = set(A)
        T = copy.copy(S)
        for val in S:
            if (val%3 != 0):
                T.remove(val)
        print(sorted(S))
        print(sorted(S-T))
    A=[]
    for n in range(3,8):
        A = A + [(n**2)%10]
    f(A)
```

(g)
```
    import copy
    A = [[0,1]]*10
    B = copy.deepcopy(A)
    B[0][0] = 2
    print(A)
    print(B)
```

Section B (Solo)

1. matrix_operations.py (25 pts)

In this question you will write four basic matrix operations. A matrix is a two dimensional list containing only integers, used in linear algebra to represent the coefficients of multivariable expressions. The matrix will be given in the skeleton code as a variable which will be passed to the functions for their corresponding operations. For example, the system of expressions

$$x + 9y - 13z$$

$$20x + 5y - 6z$$

Can be represented by the 2x3 matrix:

$$\begin{bmatrix} 1 & 9 & -13 \\ 20 & 5 & -6 \end{bmatrix}$$

Which, in turn, can be represented in Python as a two dimensional list: [[1,9,-13], [20,5,-6]]. Note that a matrix should always be represented as a two-dimensional list, even if the matrix is a linear vector.
I.e. the matrix

$$\begin{bmatrix} 5 \\ 6 \\ 12 \end{bmatrix}$$

Would be the python list [[5], [6], [12]]
For the following functions, if you're forced to print an error for whatever reason, that function should return None.

(a) Write a function `matrix_transpose(matrix_A)` returns the transpose of matrix_A as a two dimensional list. The definition of transpose can be found here: `https://en.wikipedia.org/wiki/Transpose`

(b) Write a function `matrix_det (matrix_A)` that calculates the determinant (an integer) of matrix_A if it is a 1x1 or 2x2 matrix. If the input matrix is not square, print this error message – "Cannot calculate determinant of a non-square matrix";

If the input matrix is square but has a higher dimension, print this error message – "Cannot calculate determinant of a n-by-n matrix", where you substitute n with the dimension of the input matrix. (from wikipedia `https://en.wikipedia.org/wiki/Determinant`):

$$|A| = \begin{vmatrix} a & b \\ c & d \end{vmatrix} = ad - bc.$$

(c) First, read about matrix addition here `https://en.wikipedia.org/wiki/Matrix_addition`. Write a function `matrix_add(matrix_A, matrix_B)` that performs matrix addition if the dimensionality is valid. Note that the dimensionality is only valid if input matrix_A and input matrix_B are of the same dimension in both their row and column lengths. For example, you can add a 3x5 matrix with a 3x5 matrix, but you cannot add a 3x5 matrix with a 3x1 matrix. If the dimensionality is not valid, print this error message – "Cannot perform matrix addition between a-by-b matrix and c-by-d matrix", where you substitute a, b with the dimension of the input matrix_A, and c,d with the dimension of the input matrix_B.

(d) First, read about matrix multiplication here `https://en.wikipedia.org/wiki/Matrix_multiplication`. Write a function `matrix_mult(matrix_A, matrix_B)` that performs matrix multiplication if the dimensionality is valid.

Note that the dimensionality is only valid if the number of columns of input matrix_A is the same as the number of rows of input matrix_B. Multiplying a 5x3 matrix with a 3x7 matrix is valid (resulting in a

5x7 matrix), while doing so with a 5x1 and a 7x1 matrix would not be. NOTE: order matters in matrix multiplication! Even though it is valid to multiply a 5x3 matrix with a 3x7 matrix in that order, it would not be valid to multiply a 3x7 matrix with a 5x3 matrix.

If the dimensionality is not valid, print this error message – "Cannot perform matrix multiplication between a-by-b matrix and c-by-d matrix", where you substitute a, b with the dimension of the input matrix_A, and c,d with the dimension of the input matrix_B.

2. is_anagram(a, b) (10 pts)

An anagram is direct word switch or word play, the result of rearranging the letters of a word or phrase to produce a new word or phrase, using all the original letters exactly once. Uppercase and lowercase letters are considered the same and you may add white spaces wherever you want. For example, the name "Tom Marvolo Riddle" can be rearranged the sentence "I am Lord Voldemort". Now, given two strings a and b, write a function `is_anagram(a, b)` which returns True if a and b are anagrams of one another, and False otherwise. You may not use any sorting functionality.

Hint: You may find a dictionary useful in this case.

3. is_permutation(L) (10 pts)

For a list of integers `L` of length `n`, write a function `is_permutation(L)` returns True if L is a re-ordering of the integers from 1 up to n and False otherwise.

For example, if `L=[1,2,4,3,5]`, `is_permutation(L)` should return True, because since `n = 5`, `L` is just a reordering of the numbers from 1 to 5. `L=[1,3,2,5,4]` would also return True.

If `L=[1,2,3,-5,8,0]`, `is_permutation(L)` should return False. Similarly, `is_permutation([1,1,2,3,4,5])` should return False. You may not use any sorting functionality.

4. create_FOF(friends) (20 pts)

We can create a dictionary that maps people to sets of their friends. For example, we might say that a dictionary "friends" looks like:

```
friends["Caro"] = set(["Ben", "Yanlin", "Sahil"])
friends["Sahil"] = set(["Caro", "James", "Shreyas"])
friends["Vidya"] = set(["Caro", "Yanlin", "Sahil", "Shreyas"])
friends["Ben"] = set(["Yanlin", "Caro", "Vidya"])
```

With this in mind, write the function `create_FOF(friends)` that takes a dictionary which maps people to sets of their direct friends and returns a new dictionary `friends_of_friends`, which maps all the same people to sets that consist ONLY of friends of friends.

For example, since Sahil is a friend of Caro, and Shreyas is a friend of Sahil, Shreyas is a friend-of-friend of Caro. This set should EXCLUDE any direct friends (and the people themselves), so even though Ben is also a friend of Sahil, Ben does not count as a friend-of-friend of Caro (since he is simply a friend of Caro).

In the example above, the resultant dictionary would look like

```
friends_of_friends["Caro"] = set(["Vidya","James","Shreyas"])
friends_of_friends["Sahil"] = set(["Ben", "Yanlin"])
friends_of_friends["Vidya"] = set(["Ben", "James"])
friends_of_friends["Ben"] = set(["Sahil", "Shreyas"])
```

5. registration_system (20 pts)

You are building the course registration system for Columbia in the hope of making it better. The registration system needs to be able to respond to some queries.

In following questions, d is the registration record (represented by a dictionary) whose keys are course codes (like the string "ENGI E1006") and where the corresponding value of each key is a list of student names (strings) that are enrolled in that course.

(a) Write a function `count_all_enrolled_students(d)`, that takes the registration record and return the count the total number of enrolled students in all classes. Note that students may enroll in multiple classes.

(b) Some classes have labs associated with it. Students must enroll in both the class and its associated lab. Write a function `find_invalid_registrations(d, course_code, lab_code)`, where `course_code` is the course code for class that has an associated lab and `lab_code` is the course code for the lab associated with `course_code`. It returns a list of students who have not properly registered for the class indicated by `course_code`.

(c) In the final week, several classes may have the same exam time. Write a function `notify_exam_conflict(d, course_codes)` that takes the registration record d and a list of course codes which share the same exam time slot, and returns a list of students who will have an exam time conflict.

(d) You want to hold a Python workshop for everyone in Columbia, except to those who are taking 1006 (they are awesome at Python programming already!) Write a function `hold_workshop(d)` that returns a list of students who are not enrolled in 1006 right now. Note that 1006 may not be offered every semester, so there is no guarantee that it is in the registration record. In case you don't know, the course code for 1006 is "ENGI E1006" :)

(Bonus): Implement Conway's Game of Life (10 pts)

First, read about Conway's Game of Life here:

`https://en.wikipedia.org/wiki/Conway%27s_Game_of_Life#Rules`

Write a function `game_of_life(board)`, that takes a 2D list representing the initial n-by-n board setup and a number n, and returns the board after 1 step in time. The board will represent dead cell as one white space" ", and live cell as "X".

For example, `[[" "," "," "," "], [" ","X","X"," "],[" ","X","X"," "],[" "," "," "," "]]` represents the following board: