DSC 395T Algorithms and Data Structures
Friend of a Friend **Programming Assignment #1**

In this assignment, you will use Python sequence types (e.g. List, Dict, Tuple, Set, str) to store and manipulate a dataset of "friendship" relations, which will give you practice with basic Python data types and programming constructs.

1 Friends

For this assignment, define a "friendship" relation as a link between two persons. For example, the provided text file `friends.txt` lists all pairs of movie characters who appeared together in a scene of a Star Wars movie[1]. We refer to any pair of characters who were listed together as "friends".

We can then ask queries about this set of friendship relations. For example, who is the most popular character, that is, who appeared with the most other characters? Or, for some specific character, how many others are within two "degrees of separation", that is, who are friends of the character's friends?

You will answer these questions with Python code, using appropriate data types to read, store and look up the given friendship relations.

2 Your Assignment

We ask you to complete the following tasks, by completing the functions in the starter code provided in `myfriends.py` and in `py_friends/friends.py` to return the requested answers.

1. Complete the function `load_pairs()`: This function reads as input pairs of friends from a given text file, where each non-empty line in the file contains two names separated by one or more spaces. The function should return a List of 2-tuples, where each Tuple comprises a pair of names from a line of the input file. The list you return may contain duplicates if they also exist in the input file — the number of items (i.e. friendship tuples) in the list should always equal the number of non-empty lines in the input file.

2. The list format you have just created is not well-suited for the efficient look up of any person's friends, so you should complete the function `make_directory()` to convert a list of tuples to a more convenient directory format that uses the Python Dict data type. Each key of the dictionary will be the name of a person mentioned in the input list, while the associated value is the Set of names of that person's friends.

   Note:

   - You should infer that relationships are two-way: e.g. if a pair (x, y) appears in the list, then the dictionary should be constructed to show that y is a friend of x, and x is a friend of y.
   - Ignore own-pairs of the form (x, x).

3. You are now ready to answer some basic queries of the friendship relationships that you have loaded. First, complete the function `find_all_number_of_friends()`, which takes a friendship directory as input and returns an output list of 2-tuples, where each tuple comprises a person's name and that person's number of friends. You should ensure that this function lists all persons sorted—in descending order—by their number of friends. Ties should be broken alphebatically, so if Han and Leia each have two friends, then your function should list (Han, 2) before (Leia, 2).

4. Next, complete the provided function `make_team_roster()`, which takes a person and a friendship directory and returns the person's "team", which is the union of the person's friends and the person's friends-of-friends. This function should return a string label to represent the team's roster of characters. *Please read the docstring in the starter code function template for instructions on how the team roster string is to be encoded from team member names.*

   - As a simplified example, if the friendship directory has these items:

[1] This dataset is from Gabasova, E. (2016). Star Wars social network. DOI: https://doi.org/10.5281/zenodo.1411479.

```
{'CHEWBACCA': {'HAN', 'LUKE'},
 'HAN': {'CHEWBACCA', 'LEIA'},
 'LUKE': {'CHEWBACCA'},
 'LEIA': {'HAN'}},
```

then Han's team comprises friends Chewbacca and Leia, as well as Luke, who is a friend of Chewbacca; and the team roster label is `'HAN_CHEWBACCA_LEIA_LUKE'`.

5. Your next task is complete the provided function `find_smallest_team()`, which takes as input a friendship directory and returns the roster label of the smallest team that can be formed from a friendship directory: In other words, the function first determines the size of each person's team (defined above in step (4)) and then returns the smallest roster found. If there are ties, return the team roster label that is first alphabetically.

6. Finally, you should implement an *iterator* class that iterates in alphabetical order over all all unique friendship relationships in a given friendship directory. A Python class, `Friends`, has been provided for you to complete. This class is located in the module `py_friends/friends.py`. But it contains several small bugs, which you should find and correct, so that it works correctly as specified for any given friendship directory. You can read more about Python iterator and generator types in the standard library documentation at `https://docs.python.org/3.10/library/stdtypes.html#typeiter`.

To help you with testing, we provide a *pytest* script in the `tests` subfolder, containing a subset of test functions that the autograder will use on your final submission. You may also add your own test statements to those provided in the section of the `myfriends.py` source file after the `if__name__ == '__main__':` line, but realize that the autograder will ignore these statements, and will only call your functions (which you defined above that line) directly with its own testing codes. The autograder will use other examples of friendship relationships, so to test your code, you should consider finding or creating other friendship datasets.

2.1 Karma (Optional)

As an optional exercise, write a *generator* function that provides the same functionality as, and hence can replace, the `Friends` iterator class you completed above. You should name your function `generate_friends()` and add it to your `myfriends.py` submission with some comments describing how to use your function. Python's generators provide a convenient way to implement the iterator protocol: see the standard library documentation `https://docs.python.org/3.10/glossary.html#term-generator`.

3 What to Turn in

Use the **Gradescope** page of the **Canvas** website to submit your repository via a linked Github account. Please refer to the "Submission Instructions" page on edx for Gradescope submission instructions. Canvas is available at `http://canvas.utexas.edu/`.

Important Details .

- This assignment is due at 11:59pm on the due date. Late assignments will be penalized 20% per day (unless your Slip days are assigned to it at the end of the semester).
- You may work in a group of up to 3 students, but make 1 submission per person.

Acknowledgments. This assignment was created by Terence Lim with great feedback from Maxine Tao, Matthew Giordano, and Prachi Ingle.