Important

1. Due Date: **Tuesday, November 21st**
2. This homework is graded out of 100 points.
3. This is an Individual Assignment. You may collaborate with other students in this class. Collaboration means talking through problems, assisting with debugging, explaining a concept, etc. Students may only collaborate with fellow students currently taking CS 1301, the TA's and the lecturer. You should not exchange code or write code for others. For individual assignments, each student must turn in a unique program. Your submission must not be substantially similar to another student's submission. Collaboration at a reasonable level will not result in substantially similar code.
4. For Help:
   - TA Helpdesk (Schedule posted on class website.)
   - Email TA's or use Piazza Forums Notes:
   - How to Think Like a Computer Scientists [http://openbookproject.net/thinkcs/python/english3e/]
   - CS 1301 Python Debugging Guide [http://www.cc.gatech.edu/classes/AY2016/cs1301_spring/CS-1301-Debugging-Guide/index.html]
5. Don't forget to include the required collaboration statement (outlined on the syllabus). Failing to include the Collaboration Statement will result in no credit.
6. Do not wait until the last minute to do this assignment in case you run into problems.
7. Comment or delete all your function calls. When your code is run, all it should do is build without any errors. Having function calls or extraneous code outside the scope of functions will result in a lower grade. (import statements and global variables are okay to be outside the scope of a function)
8. **Read the entire specifications document before starting this assignment.**
9. **IF YOUR CODE CANNOT RUN BECAUSE OF AN ERROR, IT IS A 0%.**

Object-Oriented Programming

Python is an **object-oriented programming language**. That means it provides features that support object-oriented programming (**OOP**). Object-oriented programming has its roots in the 1960s, but it wasn't until the mid 1980s that it became the main programming paradigm used in the creation of new software.

In object-oriented programming the focus is on the creation of objects, which contain both data and functionality together. Usually, each object definition corresponds to some object or concept in the real world and the functions that operate on that object

correspond to the ways real-world objects interact. We've already seen classes like `Math`, `Random`, and many others. We are now ready to create our own user-defined class.
[http://openbookproject.net/thinkcs/python/english3e/classes_and_objects_I.html]

Introduction

The goal of this homework is for you to showcase your knowledge about how to properly use OOP. For this assignment you will need to implement 2 classes. Refer to the rubric to see how points will be rewarded for each function. You have been given `HW10.py` to fill out with instructions in the docstrings. However, below you will find more detailed information to complete your assignment. Read it thoroughly before you begin. Don't forget to include your name and your collaboration statement. Re-download your submission from T-Square after you submit it to make sure that your code runs successfully.

User

The User class can be used to create different instances which each represent a user of LinkedIn. Each user should have a name, a current employer (if specified), skills, and connections. You will be writing methods that allow you to alter the user's attributes and interact with other User objects.

Attributes

- `name: The name of the user`
- `skills: The skills that the user has mapped to a list of other users who have endorsed the user for the skill`
- `company: The company that the user currently works for`
- `connections: A list of of users the user is connected with`

Methods

`__init__()`

- `name: <string>`
- `skills: dictionary with KEY - skill <str>: VALUE - endorsers <list of User objects> pairs. Users start out with this attribute being None, unless specified during instantiation.`

- company: company <Company object> that the user currently works for. This attribute should be None until a user is hired by a company (see Company class)
- connections: <list of connections (User objects)>. Users start out with an empty list

__repr__()

- Return string representation of User in the following format:
- "A User named [name], employed by [Company]/unemployed, skilled in [most-endorsed skill; alphabetically first if there's a tie]/nothing if no skills, with [num connections] connections"
- Example 1: if we have a User named Bob who is unemployed, and has no skills or connections, this method should return "A User named Bob, unemployed, with 0 connections"
- Example 2: if we have a User named Jane who is employed by Facebook, has Python as her most-endorsed skill, and has 10 connections, this method should return "A User named Jane, employed by Facebook, skilled in Python, with 10 connections"
- HINT: Use string formatting and if statements to match this format exactly!
- HINT: When you print a list of User objects, you will see a list of their string representations as defined in the __repr__ function. Be aware that this list still contains User objects (not strings!), but the Python shell shows you their string representations when printing.

endorse(other <User object>, skill <string>)

- Endorse *other* for *skill*
- If *other* doesn't already have that skill, add it to their skill dictionary first and then add the user

connect(other <User object>)

- Connect with *other* (add User object to your network and add yourself to their network)
- If you're already connected, nothing should happen

add_skill(skill <string>)

- Add *skill* to dictionary of user's skills (skills should begin with no endorsements)
- If the user already has *skill*, nothing should happen

`connect_with_coworkers(network<LinkedIn object>)`

- Connect with all other Users in *network* who work for the same company as User (if not already connected)
- If User is unemployed, nothing should happen

Company

The Company class is used to represent a Company that a User works for. It has the following attributes: a name, employees who work at this company, and preferred skills for this company. Once a User object has been created, they can be hired at a specific company using the Company class' `hire` method.

Attributes

- Name: Name of the company
- preferred_skills: List of skills that the company prefers its employees to have
- employees: List of users who work at this company

Methods

`__init__()`

- Name: <string>
- preferred_skills: <list of skills (strings)> to look for in Users when recruiting
- employees: <list of User objects> employed @ Company. Should be empty when Company object is first instantiated.

`__repr__()`

- Return representation of Company in the following format:
- "A Company called [company name] with [# of employees] employee(s)"

`recruit(network<LinkedIn object>)`

- Return a list of all the User objects in *network* that have at least one of the Company's preferred skills

`hire(employee <User object>)`

- Add *employee* to list of employees; change *employee*'s employer attribute to Company
- If *employee* is already employed at Company, nothing should happen

`fire(employee <User object>)`

- Remove *employee* from Company's list of users; change *employee*'s employer attribute back to None

- If *employee* is NOT currently employed at this company, nothing should happen

downsize(target_num<int>)

- Find the employees with the fewest total endorsements for their skills (if there are ties, get the ones with names earliest in the alphabet) and fire them until Company is down to *target_num* employees
- If the *target_num* is greater than the current number of employees, nothing should happen

LinkedIn

The LinkedIn class is used to represent an entire LinkedIn network. It only has one attribute: a simple list of all the User objects that belong to the LinkedIn instance. The LinkedIn class is able to perform searches for Users in the network with a certain skill, find all unemployed Users in the network, and add Users to the network, among other methods.

Attributes

- users : List of all the users that belong to this network

Methods

__init__

- users : <list of User objects> The list begins as an empty list when the LinkedIn object is instantiated. Users are added to the network through the add_to_network function (see below)

__repr__

- Return representation of LinkedIn in the following format:
- "A LinkedIn network with [# of Users] users."

search(skill<str>, num<int>)

- Return list of User objects with specified *skill* endorsed more than *num* times

find_unemployed()

- Return list of all the User objects that are unemployed

strengthen_network()

- Return the number of User objects that have less than 3 connections

add_to_network(users <List of Users>)

- Add all User objects in *users* to LinkedIn network if they aren't already in it
- DO NOT add duplicates

get_employees(company <Company object>)

- Return list of all the User objects that work for *company*

Grading Rubric

User

- __init__	5
- __repr__	5
- endorse	5
- connect	5
- add_skill	5
- connect_with_coworkers	10

Company

- __init__	5
- __repr__	5
- recruit	5
- hire	5
- fire	5
- downsize	10

LinkedIn

- __init__	5
- __repr__	5
- find_unemployed	5
- strengthen_network	5
- add_to_network	5
- get_employees	5

Provided

The following file(s) have been provided to you. There are several, but you will only edit one of them:

1.HW10.py

This is the file you will edit and implement. All instructions for what the methods should do are in the docstrings.

2. hw_10_test.py

In this file we have created a series of tests for your usage. Feel free to add your own tests to the file to cover any additional cases you would like to test. To test your code, all you have to do is have the linked_in.py and the hw_10_test.py files in the same directory. Open and run hw_10_test.py. After running the test, you should see the results. Check the results and start debugging if needed. Read more about unittest here: [https://docs.python.org/3/library/unittest.html]

Disclaimer: **The tests found in hw_10_test.py are not intended to be an exhaustive list of all test cases and does not guarantee any type of grade. Write your own tests if you wish to ensure you cover all edge cases.**

Deliverables

You must submit all of the following file(s). Please make sure the filename matches the filename(s) below. Be sure you receive the confirmation email from T-Square, and then download your uploaded files to a new folder and run them.

1. `HW10.py`

If your file is named something else, you will get a 0%. If this file does not run (if it encounters an error while trying to run), you will get no credit on the assignment.