CSc 110, Autumn 2016
Programming Assignment #11: Geolocation
Due Wednesday, November 23, 2016, 11:30 PM

thanks to Stuart Reges of the University of Washington

This program focuses on using classes. You practice with being a client of one class and being the implementer of a second class. The classes store location information for various places of interest. The target application is something like Google Maps where a user can make requests such as asking for restaurants that are near a certain location.

This assignment should not take as long to complete as the other assignments, so it is worth 14 points instead of the usual 20. Turn in three files named `myplaces.txt`, `place_information.py` and `geolocation_client.py` on the Homework section of the course web site. You will need several support files you can find on the course web site.

Part A: Location Data

The code we are writing will be more interesting to use if we have a lot of data to work with. Companies like Google have invested significant resources in identifying places of interest and recording their names, addresses, and locations along with some tags that are relevant for searching. We don't have the resources of a company like Google, but we have a lot of students. So we will use a crowdsourcing approach by having each student provide information for at least 10 places of interest in the general Tucson area. You can include any place that you think a U of A student would be likely to go including, for example, restaurants near the airport.

For each location, we want the following information:

- Name of the place of interest
- Street address (or similar description) for the place (not a complete address, just enough information to find it if you were near that location)
- One or more search tags for this place
- The latitude of this place
- The longitude of this place

For example, suppose you want to include the Starbucks near campus in our data file. You know its name and address and can come up with some tags. To find its latitude and longitude, you can go to this web page:

http://universimmedia.pagesperso-orange.fr/geo/loc.htm

Or you can get the information directly from Google Maps by following the instructions here:

http://www.wikihow.com/Get-Longitude-and-Latitude-from-Google-Maps

You can add whatever search tags you think are appropriate, but Google provides a list of standard tags that would be useful to apply to your entries:

https://developers.google.com/places/documentation/supported_types

You should put the information together on separate lines, as in:

```
Starbucks
802-804 E University Blvd
restaurant, coffee
32.23146
-110.95907
```

You are to choose at least 10 locations and to come up with a similar 5-line entry for each location. The locations should not be private residences and should not include offensive descriptions. Put it all together in a file called `myplaces.txt`. You can use any text editor you like or you can use IDLE, but if you use an editor like Microsoft Word, be sure to save it as plain text. You are allowed to include a blank line between entries to make it easier to distinguish them while editing.

Part B: geolocation_client.py

In this part of the assignment you will write client code to manipulate some `GeoLocation` objects. The `GeoLocation` class is being provided to you, so you don't have to write it. You will instead be writing code that constructs and manipulates three `GeoLocation` objects.

The popular TV series Breaking Bad made use of geographic location information. The Walter White character buried millions of dollars at a particular location in the desert outside of Albuquerque, New Mexico. He then bought a lottery ticket to help him remember that his stash was buried at a latitude of 34 degrees, 59 minutes, 20 seconds and a longitude of -106 degrees, 36 minutes, 52 seconds. Your client program will compute the distance between Walter's stash and the local FBI building and a local studio known as ABQ Studios (where Breaking Bad was filmed). Walter's stash was supposedly buried in the desert, but from this client program, you'll see that the coordinates they gave on the TV show are really the coordinates of the studio.

Your program is required to produce the following output:

```
the stash is at latitude: 34.988889, longitude: -106.614444
ABQ studio is at latitude: 34.989978, longitude: -106.614357
FBI building is at latitude: 35.131281, longitude: -106.61263
distance in miles between:
    stash/studio = 0.07548768123801672
    stash/fbi    = 9.849836190409732
```

You should use the latitude and longitude values in this log to construct the three `GeoLocation` objects. You should then print the three objects and call the `distance_from` method twice to get the desired output. Please note that the latitude/longitude information in the first three lines of output has to be produced by calls on the `__str__` method of the `GeoLocation` class and the values in the final two lines of output have to be produced by calls on the `distance_from` method of `GeoLocation`.

Part C: place_information.py (8 points)

For this part of the assignment, you will write a class called `PlaceInformation` that stores information about a place of interest. It should have the following public methods:

```
__init__(name, address, tag, latitude, longitude)
get_name()
get_address()
get_tag()
__str__()
get_location()
distance_from(spot) # spot is a GeoLocation object
```

The first three "get" methods simply return the values that were provided when the object was constructed. The `__str__` method should return the name followed by the location details inside parentheses (location details should formatted the same as in the `__str__` method for `GeoLocation`). Although the constructor takes a latitude and longitude, you should store this information inside the `PlaceInformation` object using a `GeoLocation` object. The `get_location` method should return a reference to this `GeoLocation` object. Remember that in writing your class, you don't want to include code that appears elsewhere. For example, your `GeoLocation` object knows how to compute a distance, so you should not be repeating the code for computing distances in your `PlaceInformation` class. You should instead be asking the `GeoLocation` object to perform the computation. Be sure to properly encapsulate your object with private fields.

We are providing a client program called `place_information_client.py` that can be used to test your class.

Style Guidelines and Grading:

`PlaceInformation` is similar to the `GeoLocation` class, so you can use it as a model for how to write your own class. The `GeoLocation` class will also provide a useful example of how to comment your class. Each method should be commented and the class itself should have a general comment.

Follow good general style guidelines such as: appropriately using control structures like loops and `if/else` statements; avoiding redundancy using techniques such as functions, loops, and `if/else` factoring; good variable names, and naming conventions; and not having any lines of code longer than 100 characters. All fields should be private.

Comment descriptively at the top of your programs, each function, and on complex sections of your code. Comments should explain each function's behavior, parameters and returns.