Geography 378: Introduction to Geocomputing
Lab 2: Python Functions and Containers

Assigned: 9/27
Due: 10/11
20 points

Hand-in

- Please collect your answers for each task in a separate .py file. Compress your Python files into a single .zip file and name your zip file as **lab2_yourname.zip.**
- Submit the file to the assignment folder called "Lab 2".
- Include appropriate comments to explain what each line or block of code accomplishes. **You must comment your code for full credit**.

Notes

In this lab, you will extend what you already know about Python from Lab 1 and lecture to include the use of functions and containers in performing operations on spatial coordinates. There are three requirements you must meet for full credit on each script:

1) The script must work as directed.
2) The script must not crash if given valid inputs.
3) You must include enough explanatory comments.

Although you may assume the inputs are in the proper format, you should think about how the program could fail if user inputs are not valid. Future labs will ask you to trap errors, and it isn't too soon to start thinking about that. Although not required, you could insert comments in your programs indicating where testing and trapping should take place. Note that there are two kinds of errors a user might make: 1) invalid characters, such as supplying input that can't be converted to a number, 2) numbers that are themselves invalid, such as latitudes greater than 90^{o}.

Lab Assignments

For the following tasks, you can assume all coordinates are given in valid formats.

1. (5 pts) Write a function that converts an input latitude or longitude value in degrees, minutes, and seconds (DMS; no decimals) into decimal degrees (DD). Directions:
   a) The function should take three numbers representing degrees, minutes and seconds and returns the corresponding decimal degree equivalent.

b) You can assume if the degree value is negative all 3 fields are negative. Thus "-30 30 00" is -30.5 degrees, not -29.5 degrees.

c) Test your function with some known values before continue to task 2.

2. (5 pts) In the same script, add a function that converts from DD to DMS. Directions:
   a) It should take one DD value and return 3 numbers representing D, M and S.
   b) If DD is negative, the returned degree value should also be negative. The other two fields should be positive

3. (5 pts) In the same script, add a main program that collects DD or DMS input from the user, then convert it to the other form and report the converted value. Directions:
   a) The main program needs to gather a string from the keyboard using raw_input() and detect whether it is DD or DMS (Hint: use the split function). Then pass proper DD or DMS values to the functions above to convert.
   b) The DMS input is separated by comma and contains no space. The degree field could be negative to represent a negative value. (E.g. 43,4,23 or -43,4,23)
   c) An example of input/output is shown below:

```
>>> ================================ RESTART ================================
>>>
Please enter a latitude or longitude value in DMS or DD format.
-40,26,46
The input value is in DMS form.
Its DD form is -40.4461 .
>>> ================================ RESTART ================================
>>>
Please enter a latitude or longitude value in DMS or DD format.
-79.982
The input value is in DD form.
Its DMS form is -79 58 55 .
>>>
```

4. (5 pts) In "lab2_conversion.py" you are provided with two lists. "city" has five city names and "population" has their corresponding population. In the same script, create a dictionary called "cityPop" to store both city names and population. Use city names as keys and population as values. Print the dictionary (use command "print cityPop") in the end to test your result.