CS 100 Programming I
Project 1

Revision Date: October 2, 2015

Preamble

You may develop your code anywhere, but you must ensure it runs correctly under a Linux distribution before submission.

They Occur Everyday

> *"It's snowing still," said Eeyore gloomily. "And freezing." "However," he said, brightening up a little, "we haven't had an earthquake lately."*
>
> – A. A. Milne, creator of Winnie-the-Pooh

The US Geological Survey collects real-time data on earthquakes. You have been given the task of providing a report that can be used by scientists to analyze earthquake data. Surprisingly, there are earthquakes everyday.

To generate the report, you have decided to write a Python program that will help you in your task. The data you will use to generate your report will be stored in a file. The data will contain information about each earthquake recorded in the last 7 days. Specifically, the information recorded is: datetime, latitude, longitude, magnitude, depth and region.

Here is an example of some earthquake data that was recorded by the USGS:

```
"September 28, 2011 16:12:20 UTC" 60.0673 -141.3557 3.0 10.90 "Southern Alaska"
"September 28, 2011 14:59:10 UTC" 60.8420 -151.1492 2.6 5.00 "Kenai Peninsula, Alaska"
"September 28, 2011 14:06:24 UTC" 39.9150 -120.5632 2.9 9.70 "Northern California"
"September 28, 2011 13:29:07 UTC" 38.8180 -122.8055 2.6 2.20 "Northern California"
"September 28, 2011 13:04:51 UTC" 63.6456 -149.0473 2.6 0.20 "Central Alaska"
"September 28, 2011 11:54:07 UTC" 39.8499 -118.0310 3.9 3.00 "Nevada"
"September 28, 2011 11:34:02 UTC" 59.9138 -152.7750 2.8 89.60 "Southern Alaska"
```

Your task is to compute the following summary information about the earthquakes: the total number of earthquakes, the maximum magnitude, the minimum depth and the average magnitude. You are to also print the datetime and location of the earthquake with the maximum magnitude and the datetime and location of the earthquake with the minimum depth. For example, using the data above, you would compute and print the following:

```
Total number of earthquakes: 7
Maximum magnitude: 3.9 datetime: "September 28, 2011 11:54:07 UTC" location: "Nevada"
Minimum depth: 0.20 datetime: "September 28, 2011 13:04:51 UTC" location: "Central Alaska"
Average magnitude: 2.914285714285714
```

Getting the Data

Your main program should be placed in a file named *quake.py*. The program should accept one command-line argument that represents the file containing the earthquake data for the last 7 days.

Here is an example call:

```
python3 quake.py eq7.txt
```

The file name is just an example; you can invent your own file name and data. Some additional data from the USGS you can use to test your program is available in the file test1 which you can save in your project directory.

You must use the Scanner module to read in the data. Note that the Scanner module has a routine for reading in a double-quoted string, make your life easy. You can find documentation on how to use the scanner at the top of scanner.py. Retrieve the scanner with:

```
wget troll.cs.ua.edu/cs100/python/projects/scanner.py
```

Command-line arguments

Here is a short python program that checks to make sure there is one command-line argument and, if so, prints out the argument:

```
import sys

def main():
     # check to see if there is one argument
     # look at the length of sys.argv
     # sys.argv holds the name of the program plus the arguments

     if len(sys.argv) != 2:
        print("incorrect number of arguments, should be 1")
        sys.exit(1)

    print("the name of the program is",sys.argv[0])
    print("argument 1 is", sys.argv[1])
main()
```

Strategy

Here is the strategy you should use to solve this problem:

```
Scan the first record q from the earthquakes file after creating a scanner
While the record q in the earthquakes file is != ""
    Add the earthquake to the count of earthquakes
    See if this earthquake is > the current maximum magnitude
    See if this earthquake is < the current minimum depth
    Do what is needed for the computation of the average magnitude
    Scan the next record into q from the earthquakes file
Close the scanner for the earthquakes file when all records are read
Print your results
```

As shown above, after you read each record you will have to update your various counts and values so you can compute what is needed for your report.

The USGS has lots of data available for analysis. However, for this project, we are only using some of the columns of data available. You can look at the data at the USGS website:

http://earthquake.usgs.gov/earthquakes/recenteqsww/.

Raw data is available by clicking on the the "KML/RSS Feeds Data" and then choosing one of the CSV files. Again, note that for this project, we are only using a subset of the data items they collect.

Scanner

For your program you will need to do some file I/O (Input/Output). In order to scan each record in a file, you will use a common computer science tool called a *Scanner*. This will make your IO process much simpler.

To get the Scanner, enter the following command.

```
wget troll.cs.ua.edu/cs100/python/projects/scanner.py
```

The file *scanner.py* must be in the same directory as the program that calls it, *quake.py*. Typically, a scanner is used with a loop. Suppose we wish to read each token (a token is a series of characters surrounded by empty space) in a file. Here is a loop that does that:

```
s = Scanner(fileName)              #create the scanner
token = s.readtoken()              #read the first token
while token != "":                 #check if the read was good
    token = s.readtoken()          #read the next token
s.close()                          #always close the scanner when done
```

Using a scanner always means performing the five steps as given in the comments.

Of course, the code above doesn't have any output; Let's modify the code to print each token, one per line:

```
s = Scanner(fileName)
token = s.readtoken()
while token != "":
    print(token)
    token = s.readtoken()
s.close()
```

Often, adjacent tokens in a file are related. Below is a loop that presumes each three tokens in the file are related.

```
s = Scanner(fileName)
token1 = s.readtoken()
token2 = s.readstring()
token3 = s.readtoken()
while token1 != "":
    print(token1,token2,token3)
    token1 = s.readtoken()
    token2 = s.readstring()
    token3 = s.readtoken()
s.close()
```

The input from the file might look like this:

```
10 "John Doe" 3.5
20 "Sally Smith" 2.4
```

Note that the second token is a string that contains spaces, so we must use readstring() instead of readtoken(). The readtoken() assumes the token ends when a space is encountered. However, in the USGS data there are spaces in the datetime and location. Both datetime and location are surrounded by quotes. In readstring() a token begins with a quote and ends when the next quote is encountered. This means it will include any spaces, e.g. "Central Alaksa". What do you think would happen if we read "Central Alaska" with readtoken()?

Typically, we define a function to read one collection of tokens (a collection of such tokens is known as a *record*) at a time:

```
def readRecord(s):                  # we pass the scanner in
    token1 = s.readtoken()
    if token1 == "":                # EOF
        return "","",""             # Return 3 empty strings
    token2 = s.readstring()
    token3 = s.readtoken()
    return token1,token2,token3
```

Now our reading loop becomes:

```
s = Scanner(fileName)
token1,token2,token3 = readRecord(s)
while token1 != "":
    print(token1,token2,token3)
    token1,token2,token3 = readRecord(s)
s.close()
```

Note that it is the job of the caller of *readRecord* to create the scanner, repeatedly call *readRecord*, and close the scanner when done.

Note also, that the scanner read functions always return strings (just like input). This means that both readstring() and readtoken() return strings. You will need to convert a token to a number if required.

The *readRecord* above assumes 3 values to be read per record. Note that your file has 6 values per record. This means you will have to use 6 tokens, one for each value. Remember, for the datetime and location you will have to use readstring() instead of readtoken(). Also, note that sys.argv[1] points to the filename of your file.

To use a scanner, you will need to import it into your program:

```
from scanner import *
```

You should never need to modify the scanner code. However, the scanner file contains comments that explain its use and would be worth reading. You should write comments similar to this in all your programs.

Challenges

Try to format the output so that the maximum magnitude and minimum depth start in column 0 and the datetime starts in column 25 and the location starts in column 73.

Print a title to your report that includes the time period for the data. The time period is delimited by the last and first datetimes.

Submission Instructions

Change to the directory containing your assignment. Do an *ls* command to make sure you are in the correct place. You should see, at least, your *quake.py* and *scanner.py* files. Extra files are OK. Submit your program like this:

```
submit cs100 xxxxx project1
```

Remember to replace `xxxxx` with your instructor's name.