Project 5

Due: Monday, August 30th at 11:59 p.m. PDT

Please read the **entire document** before writing any code.

Contents

1 Introduction

Please read the **entire document** before writing any code.

1.1 Attribution

This project has been adapted (by Julie Workman) from an iOS programming lab developed by Dr. John Bellardo. Many thanks to both!

1.2 Objectives

- More practice working with and searching lists
- Introduction to file input and output in Python
- Introduction to objects in Python
- Introduction to sorting in Python
- Introduction to dictionaries in Python
- Exposure to JSON data and reading data from a webpage

1.3 Collaboration

You **may not collaborate** in any way on your project. See the syllabus for more details.

2 Problem Description

In this project, you will implement a program to store and display real earthquake data. Your program will begin by reading earthquake data from a file and displaying the data in a nicely formatted table. Then you will give your user the option to sort the earthquakes in different ways and re-display the data. As a last step, you will implement a feature to read new earthquake data from the usgs.gov website to update your program with live earthquake data!

A sample run of the program is shown here. User input is in bold for clarity:

```
Earthquakes:
------------
(1.05)           9km E of Running Springs, CA at 2017-02-27 00:51:05 (-117.008, 34.204)
(2.30)           16km SSW of Big Lake, Alaska at 2017-02-27 18:23:02 (-150.074, 61.381)
(2.22)        28km SW of Rio Dell, California at 2017-02-27 00:31:18 (-124.335, 40.307)
(2.19)            5km S of Gilroy, California at 2017-02-26 21:32:18 (-121.580, 36.958)
(2.06)              6km SW of Volcano, Hawaii at 2017-02-27 10:55:59 (-155.280, 19.383)
(4.50)      122km SSE of Chignik Lake, Alaska at 2017-02-27 08:07:28 (-157.822, 55.289)
(1.00)            4km NNW of Lake Henshaw, CA at 2017-02-27 07:43:04 (-116.775, 33.281)
(2.58)       62km WSW of Ferndale, California at 2017-02-27 19:04:52 (-124.934, 40.349)
(0.98)       2km N of The Geysers, California at 2017-02-27 01:03:53 (-122.757, 38.797)
(2.80)               12km W of Harper, Kansas at 2017-02-27 00:36:40 ( -98.161, 37.287)
(2.67)             25km SSE of Waimea, Hawaii at 2017-02-27 18:34:27 (-155.544, 19.832)

Options:
  (s)ort
  (f)ilter
  (n)ew quakes
  (q)uit

Choice: s
```

```
Sort by (m)agnitude, (t)ime, (l)ongitude, or l(a)titude? m

Earthquakes:
------------
(4.50)       122km SSE of Chignik Lake, Alaska at 2017-02-27 08:07:28 (-157.822, 55.289)
(2.80)               12km W of Harper, Kansas at 2017-02-27 00:36:40 ( -98.161, 37.287)
(2.67)             25km SSE of Waimea, Hawaii at 2017-02-27 18:34:27 (-155.544, 19.832)
(2.58)        62km WSW of Ferndale, California at 2017-02-27 19:04:52 (-124.934, 40.349)
(2.30)            16km SSW of Big Lake, Alaska at 2017-02-27 18:23:02 (-150.074, 61.381)
(2.22)         28km SW of Rio Dell, California at 2017-02-27 00:31:18 (-124.335, 40.307)
(2.19)             5km S of Gilroy, California at 2017-02-26 21:32:18 (-121.580, 36.958)
(2.06)               6km SW of Volcano, Hawaii at 2017-02-27 10:55:59 (-155.280, 19.383)
(1.05)            9km E of Running Springs, CA at 2017-02-27 00:51:05 (-117.008, 34.204)
(1.00)             4km NNW of Lake Henshaw, CA at 2017-02-27 07:43:04 (-116.775, 33.281)
(0.98)        2km N of The Geysers, California at 2017-02-27 01:03:53 (-122.757, 38.797)

Options:
  (s)ort
  (f)ilter
  (n)ew quakes
  (q)uit

Choice: f
Filter by (m)agnitude or (p)lace? p
Search for what string? ca

Earthquakes:
------------
(2.58)        62km WSW of Ferndale, California at 2017-02-27 19:04:52 (-124.934, 40.349)
(2.22)         28km SW of Rio Dell, California at 2017-02-27 00:31:18 (-124.335, 40.307)
(2.19)             5km S of Gilroy, California at 2017-02-26 21:32:18 (-121.580, 36.958)
(2.06)               6km SW of Volcano, Hawaii at 2017-02-27 10:55:59 (-155.280, 19.383)
(1.05)            9km E of Running Springs, CA at 2017-02-27 00:51:05 (-117.008, 34.204)
(1.00)             4km NNW of Lake Henshaw, CA at 2017-02-27 07:43:04 (-116.775, 33.281)
(0.98)        2km N of The Geysers, California at 2017-02-27 01:03:53 (-122.757, 38.797)

Options:
  (s)ort
  (f)ilter
  (n)ew quakes
  (q)uit

Choice: n

New quakes found!!!

Earthquakes:
------------
(4.50)       122km SSE of Chignik Lake, Alaska at 2017-02-27 08:07:28 (-157.822, 55.289)
(2.80)               12km W of Harper, Kansas at 2017-02-27 00:36:40 ( -98.161, 37.287)
(2.67)             25km SSE of Waimea, Hawaii at 2017-02-27 18:34:27 (-155.544, 19.832)
(2.58)        62km WSW of Ferndale, California at 2017-02-27 19:04:52 (-124.934, 40.349)
(2.30)            16km SSW of Big Lake, Alaska at 2017-02-27 18:23:02 (-150.074, 61.381)
(2.22)         28km SW of Rio Dell, California at 2017-02-27 00:31:18 (-124.335, 40.307)
(2.19)             5km S of Gilroy, California at 2017-02-26 21:32:18 (-121.580, 36.958)
(2.06)               6km SW of Volcano, Hawaii at 2017-02-27 10:55:59 (-155.280, 19.383)
(1.05)            9km E of Running Springs, CA at 2017-02-27 00:51:05 (-117.008, 34.204)
(1.00)             4km NNW of Lake Henshaw, CA at 2017-02-27 07:43:04 (-116.775, 33.281)
(0.98)        2km N of The Geysers, California at 2017-02-27 01:03:53 (-122.757, 38.797)
```

```
(1.38)        0km NE of The Geysers, California at 2017-02-28 00:14:38 (-122.754, 38.779)
(1.18)       2km ENE of The Geysers, California at 2017-02-27 23:53:09 (-122.731, 38.788)
(5.60)                 34km ENE of Namie, Japan at 2017-02-27 23:49:02 ( 141.374, 37.584)

Options:
  (s)ort
  (f)ilter
  (n)ew quakes
  (q)uit

Choice: f
Filter by (m)agnitude or (p)lace? m
Lower bound: 2.5
Upper bound: 10

Earthquakes:
------------
(4.50)        122km SSE of Chignik Lake, Alaska at 2017-02-27 08:07:28 (-157.822, 55.289)
(2.80)                 12km W of Harper, Kansas at 2017-02-27 00:36:40 ( -98.161, 37.287)
(2.67)               25km SSE of Waimea, Hawaii at 2017-02-27 18:34:27 (-155.544, 19.832)
(2.58)         62km WSW of Ferndale, California at 2017-02-27 19:04:52 (-124.934, 40.349)
(5.60)                 34km ENE of Namie, Japan at 2017-02-27 23:49:02 ( 141.374, 37.584)

Options:
  (s)ort
  (f)ilter
  (n)ew quakes
  (q)uit

Choice: q
```

3 Specification

3.1 Required Files

You must have a total of three files in your submission:

- `quake_funcs.py`: This file must include the actual function definitions/implementations, as well as the definition of your `Earthquake` class (see below). You are given this file to start with and you must add to it. Do ***not*** change existing code (with the exception of comments, you may remove the `TODO` comments).
- `funcs_tests.py`: This file must include unit test functions testing the functions defined in `quake_funcs.py`. You are given this file to start with (see above) and you must add to it. Do **`not`** change existing tests (with the exception of comments and triple quote surrounding the last test).
- `quakes.py`: This file must include the `main` function (and will probably call all the functions developed within `quake_funcs.py`).

3.2 `Earthquake` Object

You must store each line of data in an object whose type is a class called `Earthquake`. The `Earthquake` class should have the following attributes: `place` (a string), `mag` (a float), `longitude` (a float), `latitude` (a float), and `time` (an int). Put the definition for this class in your `quake_funcs.py` module.

Add an `__eq__` function to your `Earthquake` class so that you can test two `Earthquake` objects for equality. Two earthquakes are equal if all of their attributes are equivalent. *You need to add this function in order to pass my test cases. I give you tests in `funcs_tests.py` to test creating `Earthquake` objects and to test them for equality.*

3.3 Input File

Your program will read in initial earthquake data from a file.

You are required to write a function `read_quakes_from_file(filename)` that takes a string filename as input and returns a list of `Earthquake` objects as output. Make sure this function works with any filename you give it. In your main, you will take the earthquake filename as a command line argument.

Example quake files, e.g. "`quakes0.txt`" are given to you in the following format:

```
1.05 -117.0085 34.2036667 1488185465 9km E of Running Springs, CA
2.19 -121.5801697 36.9580002 1488173538 5km S of Gilroy, California
```

The magnitude, longitude, and latitude for each earthquake are given first. You should read these as floats. The time of the earthquake (in seconds since the Unix Epoch (January 1, 1970)) comes next. Read this as an integer. The rest of the line is the place of the earthquake. Store this as a string.

Hint: Recall this `split` and `join` methods of strings.

I give you a test in `funcs_tests.py` to test reading earthquakes in from a file.

Your function may assume any file given to it exists and contains valid earthquake data. I will not test with non-existent or invalid files.

3.4 Output

Your program should format and output all the earthquake data to the screen in a table. (See above for sample output.) The magnitude should be printed to 2 decimal places, the "place" of the earthquake in a column of 40 spaces, then the date and time of the earthquake, followed by the longitude and latitude printed to 3 decimal places each.

I give you a function to convert from the Unix timestamp to a formatted string displaying the time. Use this function to display the time in your table. The function is called `time_to_str` and is located in the given `quake_funcs.py` module.

I suggest writing a function to display the earthquake data. (A good `__str__` method may help.)

3.5 Program Options

After displaying the earthquake data to the user, display a list of program options. (See above for formatting.)

- Should the user select any of the sorts, sort the data according to the format chosen and re-display the data. If the user chooses magnitude or time, the earthquakes should be sorted in descending order. If the user chooses longitude or latitude, the earthquakes should be sorted in ascending order.
- If the user chooses to filter the quakes, get additional information regarding how the quakes should be filtered (see sample above). Display the filtered list of quakes to the user, but *do not* alter the original list of quakes.

  - Do not alter the order of the quakes. For example, if the original list of quakes is sorted by time, then the filtered list should also be ordered by time. Note that this shouldn't be any extra work on your behalf. Just don't reorder the quakes.
  - You must write and use two additional functions to help with this task:

    `filter_by_mag(quakes, low, high)` takes a list of quakes, a low value, and a high value. The function returns a new list of quakes consisting of all the quakes with a magnitude between the low and high values (inclusive). Note that this function does no printing.

    `filter_by_place(quakes, word)` takes a list of quakes and a string to search for as input. The function returns a new list of quakes consisting of all the quakes with the search word contained in the place string. The search should be case insensitive. Note that this function does no printing.

- If the user chooses to find new earthquakes, you must parse JSON data from the usgs.gov website to add the latest earthquakes to your program (see below for more details). This should be the last thing you implement in your program ans is worth 20 points of your program.
- If the user chooses to quit, then write the earthquake data back out to the input file to assure that any new earthquake data is saved. Note that you must write the data to the file in the *same format* that it was read in.

Additional menu details

- Your menu must work regardless of whether the user enters upper or lowercase characters. For example, if the user presses '`M`', then your program should sort the earthquakes by magnitude.
- Your program will not be tested with invalid input.

3.6 Reading New Quakes

Should the user choose to find new earthquake data, your program needs to read JSON (**J**ava**S**cript **O**bject **N**otation) data from the usgs.gov website. I provide a function for you to get and return this data as a Python dictionary, `get_json(url)`. See the last page of this document for a sample of what the data looks like (or go to the website and see it for yourself live!). You'll call my function with the url:

`http://earthquake.usgs.gov/earthquakes/feed/v1.0/summary/1.0_hour.geojson`

Then take the returned dictionary and get the "features" array.

For each feature in the features array, create a new `Earthquake` from the feature. You must write and use a function `quake_from_feature(feature)` in your solution. The function takes a feature dictionary as input and returns an `Earthquake` object as output. Simply read each piece of data from the appropriate place in the given feature dictionary and pass it to the appropriate part of the `Earthquake` constructor. For example, to read the "place" of the earthquake, you would access the `feature['properties']['place']`

Important: the time in the JSON data is stored as integer *milliseconds* since the epoch. You must convert from milliseconds to seconds when you construct a new earthquake object.

Finally, once you have a new `Earthquake` constructed and returned from your `quake_from_feature` function, check to see if it is already *in* your list of earthquakes. IF not, append the new quake to the list. ***Hint***: If your `__eq__` method if written properly, you may use the `in` operator. Additionally, keep track of whether or not your program found any new quakes.

When finished adding new quakes to the list, print "New quakes found!!!" if one or more quakes were added to the list. Then (regardless of whether or not new quakes were found) print the list of quakes and menu options again.

I provide a test in `funcs_tests.py` for your `quake_from_feature` function.

4 Testing

4.1 Unit Testing

You are given a sample `funcs_tests.py` with which to work. Make sure your code passes all these tests and add more of your own. You do not need to test the functions I've written.

4.2 Diff Testing

Your `main` code is required to produce the same output as mine. Like previous projects, you can check by diffing your output with mine givien certain user inputs. The difficulty in this project is that your code doesn't just read user input and print output. It:

- Reads data from a file;
- Reads user input;
- Prints output; and
- Writes data back to the same file.

So, to `diff` test your code, you can follow these steps carefully (and ask me if you don't understand any of the steps).

Each test case has four files associated with it. For example, test 0 has: `quakes0.txt`, `quakes_final0.txt`, `in0.txt`, and `out0.txt`.

The following steps are for doing test 0. Replace the 0 in each file name with a 1 for test 1, etc. (*Note*: Your program is destructive in that it modifies the starting quakes file. For this reason, there are backups of all the test files available on Canvas.)

1. Make a backup copy of the quakes file `quakes0.txt`.

   ```
   cp -i test_files/quakes0.txt test_files/quakes0.txt.bak
   ```

2. Run your program with the starting quakes file `quakes0.txt` and with the user input provided in `in0.txt`, saving the output to `my_out0.txt`.

   ```
   python3 quakes.py test_files/quakes0.txt < test_files/in0.txt > test_files/my_out0.txt
   ```

3. Diff your program output in `my_out0.txt` with the expected output in `out0.txt`.

   ```
   diff -wB test_files/my_out0.txt test_files/out0.txt
   ```

   The `-wB` flags ignore whitespace differences. Getting the whitespace to match mine exactly will be difficult (without me being a lot more specific in places) and wholly unnecessary.

4. Diff the quakes file `quakes0.txt` (which should have been altered by your program) with the expected `quakes_final0.txt`.

   ```
   diff test_files/quakes0.txt test_files/quakes_final0.txt
   ```

5. Copy your backup of the quakes file back. This is to reset your files to run the diff test again. Note that this should ask your permission (if you include the `-i`) because it will be deleting a file. Confirm you're deleting the correct file and type `y`.

   ```
   cp -i test_files/quakes0.txt.bak test_files/quakes0.txt
   ```

   When you run the tests for a second time, you can skip to step 2 as you already have the backup file created.

If you find differences (or your program seems to be behaving wildly), I suggest looking at the user input file (`in0.txt`) and running your code by hand to see what it does when you type in that input.

IMPORTANT: If you want to re-run a test, you must start with a "clean" version of the quakes file. If you did the copying and restoring of a backup correctly, you should be ready to start back with step 1. If you accidentally overwrote a test file, you can find all the tests on Canvas.

IMPORTANT 2: Please be careful copying the various quake files around so that you don't accidentally overwrite your `quakes.py` file (or any of your other Python files). If you have a version of any file saved in a git commit, you can restore it, but otherwise it's gone forever.

Note that not of the diff tests test the New Quakes feature of your program. The diff tests can help assure you that you are reading, sorting, filtering, displaying, and saving properly. If you confirm that your program, is able to get new quakes from the website, then you are probably good to go!

5 Grading

Your program grade will be based on:

- thoroughness of your `funcs_tests.py`,
- the number of *my* unit tests passed,
- the number of `diff` test cases passed,
- adherence to the specification,
- your program design, and
- program style.

Your program will be tested with test cases that you have not seen. Be sure to test your program thoroughly!!! To pass a test case, your output must match mine EXACTLY. Use `diff` to make sure your output matches mine in the test cases given and in test cases that you make up. If there are any differences whatsoever, you will not pass the test case.

6 GitHub Submission

Push your finished code back to GitHub. Refer to Lab 1, as needed, to remember how to push your local code.

7 Sample JSON Earthquake Data

```
{
    "type": "FeatureCollection",
    "metadata": {
        "generated": 1488185761000,
        "url": "http://earthquake.usgs.gov/earthquakes/feed/v1.0/summary/1.0_hour.geojson",
        "title": "USGS Magnitude 1.0+ Earthquakes, Past Hour",
        "status": 200,
        "api": "1.5.4",
        "count": 3
    },
    "features": [
        {
            "type": "Feature",
            "properties": {
                "mag": 1.05,
                "place": "9km E of Running Springs, CA",
                "time": 1488185465040,
                "updated": 1488185688121,
                "tz": -480,
                "url": "http://earthquake.usgs.gov/earthquakes/eventpage/ci37814024",
                "detail": "http://earthquake.usgs.gov/earthquakes/feed/v1.0/detail/ci37814024.geojson",
                "felt": null,
                "cdi": null,
                "mmi": null,
                "alert": null,
                "status": "automatic",
                "tsunami": 0,
                "sig": 17,
                "net": "ci",
                "code": "37814024",
                "ids": ",ci37814024,",
                "sources": ",ci,",
                "types": ",geoserve,nearby-cities,origin,phase-data,scitech-link,",
                "nst": 27,
                "dmin": 0.09331,
                "rms": 0.23,
                "gap": 83,
                "magType": "ml",
                "type": "earthquake",
                "title": "M 1.1 - 9km E of Running Springs, CA"
            },
            "geometry": {
                "type": "Point",
                "coordinates": [
                    -117.0085,
                    34.2036667,
                    6.79
                ]
            },
            "id": "ci37814024"
        },
        {
            "type": "Feature",
            "properties": {
                "mag": 2.22,
                "place": "28km SW of Rio Dell, California",
                "time": 1488184278000,
                "updated": 1488184563372,
                "tz": -480,
                "url": "http://earthquake.usgs.gov/earthquakes/eventpage/nc72768481",
                "detail": "http://earthquake.usgs.gov/earthquakes/feed/v1.0/detail/nc72768481.geojson",
                "felt": null,
                "cdi": null,
                "mmi": null,
                "alert": null,
                "status": "automatic",
                "tsunami": 0,
                "sig": 76,
                "net": "nc",
                "code": "72768481",
                "ids": ",nc72768481,",
                "sources": ",nc,",
                "types": ",geoserve,nearby-cities,origin,phase-data,scitech-link,",
                "nst": 5,
                "dmin": 0.04125,
                "rms": 0.01,
                "gap": 283,
                "magType": "md",
                "type": "earthquake",
                "title": "M 2.2 - 28km SW of Rio Dell, California"
            },
            "geometry": {
                "type": "Point",
                "coordinates": [
                    -124.3346634,
                    40.3071671,
                    7.13
                ]
            },
            "id": "nc72768481"
        },
        {
            "type": "Feature",
            "properties": {
                "mag": 1.6,
                "place": "36km W of Kenai, Alaska",
                "time": 1488182473565,
                "updated": 1488183500820,
                "tz": -540,
                "url": "http://earthquake.usgs.gov/earthquakes/eventpage/ak15412705",
                "detail": "http://earthquake.usgs.gov/earthquakes/feed/v1.0/detail/ak15412705.geojson",
                "felt": null,
                "cdi": null,
                "mmi": null,
                "alert": null,
                "status": "automatic",
                "tsunami": 0,
                "sig": 39,
                "net": "ak",
                "code": "15412705",
                "ids": ",ak15412705,",
                "sources": ",ak,",
                "types": ",geoserve,origin,",
                "nst": null,
                "dmin": null,
                "rms": 0.52,
                "gap": null,
                "magType": "ml",
                "type": "earthquake",
                "title": "M 1.6 - 36km W of Kenai, Alaska"
            },
            "geometry": {
                "type": "Point",
                "coordinates": [
                    -151.923,
                    60.5679,
                    64.7
                ]
            },
            "id": "ak15412705"
        }
    ],
    "bbox": [
        -151.923,
        34.2036667,
        6.79,
        -117.0085,
        60.5679,
        64.7
    ]
}
```