CSE 337: Homework Assignment 1

Instructions

Please read the the following instructions carefully before coding. You may lose points if you fail to follow these instructions.

- Deadline for this assignment is **September 30, 11:59PM. No late submissions accepted**.
- Stick with built-in Python libraries unless otherwise specified
- Keep your answers for each question in a separate file, and specify the file name clearly (*e.g.,* q1.py for question 1). For the questions with multiple parts, you can use a filename like q1_p1.py
- Make sure your programs work in other machines. One way to test this is to test your program in our UNIX servers, and see if it works there. You will lose marks if your program fails to run in our machines.
- Put all of your python files in a single folder (Do not make a sub-folder for each question).

  Name and zip your folder like this: Ying_Lu_111234567.zip

1 Commodities Trading [20 pts]

You decided to make money by trading commodities. To make meaningful amounts of money, you need other people's money. However, it is hard to convince people with money unless you can quantitatively demonstrate you are not going to lose money. Hence, you will need to calculate various statistics on your prices data.

Part 1. You are provided with a file(prices_sample.csv) where each line is time and the corresponding commodity price. First, you will need to convert the time to a more desirable format. The timestamps are expressed in UNIX epoch, which means it denotes the number of seconds passed since Jan 1, 1970(*e.g.,* Jan 1, 1970 means 0). You need to convert this to a date time format(*i.e.,* YYYY-MM-DD HH:MM:SS). Fortunately, Python already provides functionality for this conversion. Find that out, and use it to group together the prices that belong to same day with the following format:

YYYY-MM-DD: $P(HH_1 : MM_1 : SS_1), P(HH_2 : MM_2 : SS_2), .., P(HH_N : MM_N : SS_N)$

where $P(HH_1 : MM_1 : SS_1)$ is the price corresponding to the first timestamp of the day. For example, the output for April 1, 2019 should be something like following:

2019-04-01: $513, 512, 511, 510, ..., 523$

Then write the result to a file in this format, one line for each day. **[8 pts]**

Part 2. Go over the prices file, and calculate the 25th, 50th and 75th percentiles and the variance ($\left(\sigma = \frac{\sum(x-\mu)^2}{N}\right)$) for the prices, and print them to the screen. Also, find the five days with farthest price to the mean, and print those dates and the prices to the screen. **[12 pts]** Note: The days you print should be five different(unique) days.

2 Daily Data [15 pts]

It turns out looking at the prices in week-resolution could provide a higher-order summary of the price trends.

Question. Write a program that computes the max, min, and the average prices for each and every week, and save it to a new file in a comma separated format. (YYYY-MM-DD followed by the values asked in the question, where YYYY-MM-DD is the first day of the week). You need to take into account incomplete weeks as well. **[15 pts]**

3 One Step Ahead [15 pts]

Your model should work well in theory, but it has a problem: Too many people know what you know, and you can't profit using this strategy. You need to step up your game. What if you were able throw in more data to your model?

Question. Write a program to parse and get all the Symbol, Last Price, Market Time, and Change fields from the "commodities futures" table from `https://finance.yahoo.com/commodities`. The program returns the a list of lists, where each list contains the Symbol, Name, Last Price and Market Time of a commodity. Write all these to a file named commodities.txt, one line for each commodity, and with comma separated values(*e.g.,* symbol, last price, market time, change)**[15 pts]**. Hint: You do not have to use BeautifulSoup for this, but it may make things much easier.

4 Word Analytics [20 pts]

You decided to add NLP(Natural Language Processing) capabilities to your model. One good way of deriving information from text is to count words in the text.

Part 1. To this end, you need to write a program that gets a filename and n as input, retrieves the text, and returns a dictionary where keys are n-grams(*i.e.,* groups of n consecutive words, for example, 3-grams in "to be or not to be" are "to be or", "be or not", "or not to" and "not to be"), and the values are the number of occurrences of n-grams that appear more than once in the text**[15 pts]**. Note: For the purposes of this question, assume $1 \leq n \leq 10$.

Part 2. Using the results of the previous part, find the 10 n-grams with the highest total length(*i.e.,* len(w1) + len(w2) + len(w3)) **[5 pts]**.

You can test your program on this file after downloading it: `http://www.gutenberg.org/files/100/100-0.txt`

5 Password Check [15 pts]

Your hedge fund started to make good money, and some of your friends have become investors. You decided to provide information to your investors using a website. To ensure your users have strong passwords, you will check for certain conditions. Here are the rules for a strong password:

- The password should contain at least 6 characters and at most 20 characters in total.
- The password should contain at least one numerical character, one upper-case character and one special character(*i.e.,* neither numerical, nor alphabetical)
- A substring of length 3 cannot appear more than once in the password. For instance, "3rtA%1rtA" has two substrings "rtA", hence it fails this rule.
- The password should not be a palindrome. And given memory constrains, you need to be able to check this condition using constant memory.
- The number of unique characters in the password, should be higher than half of the length of the password. For example, "abAa5b&cabc" has six unique characters, "A", "5", "a", "b", "&" and "c", but the length of the password is eleven, hence it passes this test.
- Username and the reverse of the username should not appear in the password.

Part 1. Write a Python program with two inputs (username and password) as strings and it should return True if they fulfills the rules for strong passwords, otherwise return False. **[10 pts]**.

Part 2. Write a program that prompts the user to enter a username and password, and does not terminate until a strong password entered(*i.e.,* keeps asking for the password)**[5 pts]**.

6 Coding for Fun [15 pts]

Your hedge fund made you a billionaire, and now you are coding only for fun.

Part 1. Find and print all 3-narcissistic numbers between 100 and 999 (inclusive) by using at least one lambda function

An n-digit number that is the sum of nth powers of its digits is called an n-narcissistic number. For example, 153 is a 3-narcissistic number for $153 = 1^3 + 5^3 + 3^3$.**[5 pts]**

Part 2. Write a program which uses map() and filter() to use a list as exponents and create a new list with the exponentiated values that are less than 1000 where base is 2.

For example, if your input is [1, 4, 20], you should print out the list [2, 16], because $2^1 = 2, 2^4 = 16, 2^{20} = 1048576$, but only 2 and 16 are less than 1000. **[5 pts]**.

Part 3. Use at least two of map, filter or reduce functions to count words in a string. Assume that the words are provided as a list of strings(*i.e.,* [word1, word2, word3, ... ,]), and return a list of tuples with the format [(word1,count1), (word3, count3),...,]. **[5 pts]**.