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 **March 1, 11:59PM. No late submissions accepted**.
- The Python version used in this assignment **must be 2.7**
- 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: Trung_Nguyen_111234567.zip

1 Crypto-coin Hedge Fund [20 pts]

With the rise of crypto-currencies, you decided to make money by trading Bitcoins. 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 Bitcoin value. 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 and HH:MM:SS). Fortunately, Python already provides functionality for this conversion. Find that out, convert each timestamp in the file to actual date and time, and write it back to another file named datetimes.txt. **[7 pts]**

Part 2. Go over the prices file, and calculate the max, min, mean and the standard deviation for the prices. Find the corresponding dates for max, min and mean, and print them to screen. **[13 pts]**

2 Correlations Everywhere [15 pts]

It turns out Bitcoin has a high standard deviation, which means it is a risky trading instrument. However, your investor friends decide to invest in it anyways. Now what you need is a trading strategy. It turns out one way to do this is to find two correlated instruments, and use this correlation to exploit market inefficiencies. Hence, you will need to calculate Pearson Correlation between two variables. It turns out the two columns of numbers you have in the original file is a good means of testing your program.

Question. Write a program that computes the Pearson Correlation between two variables(Unix timestamp and bitcoin price in this example.). Print the result to the screen. **[15 pts]**

3 One Step Ahead [20 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 to read and analyze finance news automatically and use it to improve your trading models? This will be faster than a human reading the news and it may give you the edge.

Question. Write a program to parse and get the titles, source(*e.g.,* Wall Street Journal) and the date(*e.g.,* 1 hour ago) of all articles from `https://finance.google.com/finance/market_news`. The program returns

the a list of lists, where each list contains the title, source and the date of an article. Write all these to a file named top10articles.txt, one line for each article, and with comma separated values(*e.g.,* title,source,date)**[20 pts]**. Hint: You do not have to use BeautifulSoup for this, but it may make things much easier.

4 Word Analytics [15 pts]

Now you are able to grab finance articles and use them for your analytics. One good way of deriving information from text is to count words in the text. To this end, you need to write a program that gets a filename as an input, retrieves the text, and returns a dictionary where keys are words, and the values are the number of occurrence of these words.

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

Part 1. Write a Python program that parses the given file path and returns a dictionary where keys are the words in the text, and the values are the word counts. A word is defined by the space characters around it. You should ignore the empty lines [10 pts].

Part 2. Using the dictionary you created in the previous question, find the most 10 common words, and write them into top10words.txt file [5 pts].

5 Password Check [15 pts]

Your hedge fund started to make good money, and you have many investors. You decided to provide information to your investors using a website. You need to make sure your users have strong passwords. You will need to check the strength of their password. Here are the rules for a strong password:

- The password should contain at least 8 characters
- The password should contain at least one numerical, one alphabetical, and one special character(neither numerical, nor alphabetical)
- The password should not contain the same character more than two times consecutively
- There should not be patterns with increasing values of length 3 or more(*e.g.,* "abc" or "789"). Hint: "ord()" function may come handy here.
- Number of distinct characters in the password should not be less than half of the length of the password. For example, "ababcabc" has only three distinct characters, "a", "b" and "c", but the length of the password is 8, hence it fails this test.

Part 1. Write a Python program that takes a string as input and checks for these criteria, and returns True if the password is strong enough, and False if the password is not strong enough**[10 pts]**.

Part 2. Write a program that prompts the user to enter a 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 prime numbers between 2 and 100 (inclusive) by using at least one lambda function**[4 pts]**.

Part 2. Write a program which uses map() and filter() to make a list whose elements are square of even number in [1,2,3,4,5,6,7,8,9,10]. Print the result. **[3 pts]**.

Part 3. Write a recursive function to calculate and print all interleavings of two lists. The main property of interleavings is that the order of elements in the individual lists are preserved. For example, for inputs [1,2] and [3,4], your program should result in [1,2,3,4],[1,3,4,2],[1,3,2,4],[3,4,1,2],[3,1,2,4],[3,1,4,2]. **[3 pts]**.

Part 4.

```
def count_pattern(str, pattern, replace_str):

    assert len(str) >= len(pattern)

    # complete the function
    ...
```

Complete the above function which takes three string variables as input, and replace all substrings in the first string that matches a given pattern specified by the second input with a given substitute string specified by the last input, and return this updated string as output. For example, count_pattern('shjhfdddedaaba','xyx', '123') returns 's123fdd123a123' as there are three substrings in the first string that match the given pattern: 'hjh', 'ded', 'aba'. **[5 pts]**