**Data Focused Python**

**Homework 5**

**Mini 1 2024**

***Due at 11:59 pm on Monday, Sept. 30***

***You will lose 1 point every 5 minutes after that time***

1. **(100 points) Regular Expressions**

The **expenses.txt** file provides good examples for regular expression matching.

1. Create a Python script file named **expense\_regex.py**. In this script, define an empty list named **records**, then read the lines from **expenses.txt** and **append** each line (*excluding* its terminating newline character) to the **records** list. (As an alternative, you can use a **list** *comprehension* to create the **records** **list**.)

Add this code for displaying lines from **records** that match a trivial regular expression:

**import re**

**# 1a**

**pat = r'D'**

**for line in records:**

**if re.search(pat, line) != None:**

**print(line)**

Confirm that the output is lines from **records** that contain a **D** somewhere.

1. Comment out the pattern for part **1a**, and add a new pattern for part **1b**, like this:

**# 1a**

**# pat = r'D'**

**# 1b**

**pat = r'\''**

Confirm that the output is lines from **records** that contain a single quote (**'**) character

somewhere.

1. Comment out the pattern for part **1b**, and add a new pattern for part **1c** that will display lines from records that contain a double quote (**"**) character somewhere. Run the module to test.
2. Continue in this manner of commenting out the previous part’s pattern and adding a new pattern. Add a pattern that will display lines that begin with **7**; run the module to test.
3. Continue in this manner…. Add a pattern that will display lines that end with an **r** *or* a **t**; test.
4. Add a pattern that will display lines that contain a literal period (**.**) character; test.
5. Display lines that contain an **r** followed later by a **g**. (The **r** and the **g** do not need to be consecutive characters.)
6. Display lines that contain two *consecutive* uppercase letters (for example, **AA**, **DF**, **LM**, **YW**, …).
7. Display lines that contain a comma (**,**) character.
8. Display lines that contain *three or more* comma characters (not necessarily consecutive).
9. Display lines that *do not* contain any **v**, **w**, **x**, **y**, or **z** characters.
10. Display lines that contain money amounts between 10.00 and 99.99. ***Hint:*** What must the first character be? What must the second character be? What must the third character be? And so forth. The pattern **'[10.00-99.99]'** *will not work*, because one pair of square brackets only matches a single character, not a range of multiple characters. For example, **'[aaaaaaa]'** matches one **a**, not a sequence of seven **a** characters. The pattern **'[abcabcabcaaabbbaaaccccccc]'** matches the same thing as the pattern **'[abc]'** or the pattern **'[acb]'** or **'[bac]'** and so forth: that is, *either* an **a** *or* a **b** *or* a **c**. The regular expression pattern **'[AA-ZZ]'** matches *either* an **A**, *or* a character in the range from **A** through **Z**, *or* a **Z**; that is, the pattern is equivalent to **'[A-Z]'**.
11. Display lines that contain *exactly three* commas.
12. Display lines that contain a **(** character (that is, an *open parenthesis* character).
13. Display lines that describe meals costing at least 100.00.
14. Display lines that have an expense category that is *exactly four characters wide* (your pattern should work even if more lines are added to the file, with new categories that have not yet been defined).
15. Display lines for expenses that occurred in March.
16. Display lines that contain an **a**, followed by a **b**, followed by a **c** (perhaps with other characters between the **a** and the **b** and the **c**).
17. Display lines that contain some sequence of two characters, followed later by that same sequence of two characters, followed later by that same sequence of two characters again. That is, each matched line should contain at least one sequence of two characters at least three times.
18. Display lines in which the expense description contains a both a lowercase **a** and a decimal digit character between **0** and **9**, *in either order*. That is, the **a** might appear *before* the digit, or the **a** might appear *after* the digit.
19. Display lines that contain no uppercase letters.
20. Display lines that contain a **d** character, possibly followed by one *optional* character, followed by an **i** character. (Matches would include words like **diver**, **doily**, **drip**, **diplomat**, etc.)

***When finished, put your expense\_regex.py source code file into a zip archive named* Team\_***N***\_HW5.zip *file, where*** *N* ***is your team number, and upload your .zip file to Canvas.***