Homework 1: Regular Expressions

Due Sept. 11

1 RegEx Practice (30 points)

Practice regular expressions by playing RegEx Golf.

You **must** complete the following (28 points, autograded):

- Warmup (4 pts)
- Anchors (4 pts)
- It never ends (4 pts)
- Ranges (4 pts)
- Backrefs (4 pts)
- One additional puzzle of your choosing (8 pts)

1.1 Turn In

In the file **golf.py**, fill in the empty strings with your best (that is, shortest) regular expression for each of the puzzles that you solved.

You'll get 2 points for turning in a nontrivial file, and full points for each solution as long as your regular expression didn't avoid the challenge of writing rules that generalize properties of the words (e.g., r"^(word1|word2|word3...)$" would be ridiculously long and a little silly).

That said, if your length seems multiple times longer than what's on the leaderboards, challenge yourself to get one that's shorter!

1.2 Integrity Note

Since these puzzles are online, there are (undoubtedly) solutions posted somewhere. I trust you not to look for those solutions, but to submit the best solutions you can come up with on your own.

2 Chatbot (60 points)

For this part of lab, you will extend the poetry chatbot that we started in class. You can add new functionality, or improve how the chatbot handles parts of the conversation that we worked through in class.

2.1 Using regular expressions (15 points)

You should use each of the following at least once (5 points each):

- Regular expression groups ((...)). **Note**: these should be used to capture content that your chatbot processes later; don't just put parentheses around a regular expression and then ignore the group.
- Character classes ([...], \d, \D, \w, \W, \s, \S, etc.)
- Regular expression quantifiers (+, *, and/or {...})

One weakness of our existing chatbot is how lightly it validates input. You could work on improving this with regular expressions. Or you could add additional search criteria to the chatbot, such as excluding particular poets.

2.2 Sentiment analysis (15 points)

Sentiment analysis is the task of identifying the emotion associated with a piece of text. The simplest way of doing this is to build a list of words associated with positive emotions, and a list of words associated with negative emotions, and assign the text a sentiment score based on how many of these words occur in it.

Write a function called **sentiment** that takes a poem as input and returns a sentiment classification ("positive", "neutral", "negative").

Extend the chatbot so that it asks the user what kind of poetry it is in the mood for, and filters poems by sentiment using a function called **filter_sentiment**. As with the functions we wrote in class, your filtering function should always return at least one poem, even if this means ignoring the filtering criteria.

You may not use a prebuilt sentiment analysis system from a package.

2.3 Filtering by poem length (15 points)

Extend our chatbot so that it asks the user whether they prefer long or short poems.

Write a function called **prioritize_poem_length**, which takes a length preference as a string and a set of poems. Your function should calculate the average poem length in the dataset and filter out poems with above average or below average lengths, depending on the user's length preference.

You may find it useful to write a helper function to validate the user's length response.

Once you have **prioritize_poem_length**, write a function called **length** to handle the dialogue eliciting the user's length preference. This should return the user's length preference.

Modify **poem_search** to obtain and apply your user's line length preference by calling **length** and **prioritize_poem_length**.

2.4 Report (15 points)

Write a short report that describes your bot. It should have the following sections:

- **Regex Description**: Clearly describe how you make use of each of the required regular expression features in your chatbot. For at least one regular expression, discuss a false

negative or false positive that arises. (5 points for completeness and clarity)

- **Analysis**: Describe at least one interaction with your bot that worked well, and at least one interaction with your bot that works poorly (or not as one might expect). You should include a screenshot or transcript of your conversations in your writeup. This should include explanation not just of what happened, but why it happened, and why that was good or bad. (5 points for completeness and clarity)
- **Future Directions**: Thoughtfully describe how you would address the existing shortcomings of your bot if you had more time. For example, what would you do to make your bot better if you had another week to work on it? If you were going to use this code as the starting point for a final class project? (5 points for completeness and clarity)

2.5 Curiosity (10 points)

For each homework assignment, I will assign 10 points based on intellectual curiosity. By following the instructions above, you will receive a 90 on the assignment. To receive a higher score, you will need to go beyond these instructions.

For this assignment, you could exercise curiosity by including a particularly in-depth analysis in your report. You could add an additional feature for your chatbot. Or you could show it in some other way relating to your chatbot or to regular expressions.

Please include a section in your report detailing what (if anything) you are submitting for these points.

2.6 Submission

On Gradescope, you should submit your **golf.py** file, your extended **poetry_bot.py** file (5 points for working, non-trivial file) and your report PDF.