EE361: Fundamentals of Software Engineering

Instructors: Faraz Hussain || Aratrika Ray-Dowling

Spring 2025

Homework on CH8

"Homework on CH8" Question framing and posting: Aratrika Ray-Dowling

Due on: April 15, 2025 || 10:00 PM

Total Points: 30

The supporting dictionary '`words.txt`' and the ciphertext '`ciphertext.txt`' can be found in the zipped file '`Supporting_files.zip`'.

Question-1: Railfence Cipher *(10 points)*

Here is a target plain text:

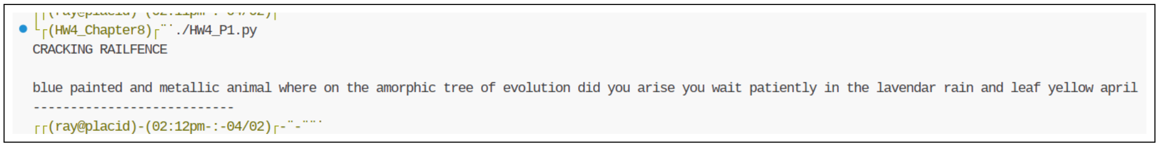
> Blue-painted and metallic animal, Where on the amorphic tree of evolution Did you arise? You wait patiently in the lavendar rain And leaf-yellow April

The above plain text is encrypted to cipher text through railfence/transposition cipher using 4 rails.
Please perform the following and write a Python script.

- The corresponding cipher text of the above plain text will be provided to you in a textfile "`ciphertext.txt`". Please load the textfile to your Python script.

- Utilize the `railDecrypt()` and `createWordDict()` functions and call them inside `railBreak()` function to break the ciphertext to the appropriate plaintext.

- Your resultant `railBreak()` must produce the below expected output. The dictionary to work with `railBreak()` is provided to you as a textfile `words.txt`



```
● └┌(HW4_Chapter8)┌¨˙./HW4_P1.py
  CRACKING RAILFENCE

  blue painted and metallic animal where on the amorphic tree of evolution did you arise you wait patiently in the lavendar rain and leaf yellow april
  --------------------------
  ┌┌(ray@placid)-(02:12pm-:-04/02)┌-¨-¨¨˙
```

Question-2: Calculate letter frequency of ciphertext *(10 points)*

Write a Python script and perform the following over the text provided in `ciphertext.txt`.

- Please load the textfile `ciphertext.txt` to your Python script.

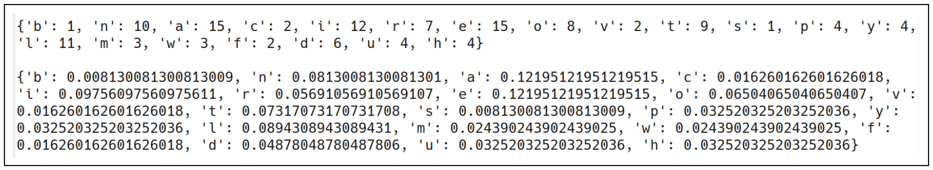
- Use the function `removeMatches()` within the modified versions of `letterFrequency()` function.

- Make two versions of the `letterFrequency()` function where one should calculate the frequency of unique letters in the cipher text in terms of count/number. The second version of `letterFrequency()` function should calculate the frequency of unique letters in the cipher text in terms of percentage.

- While making the two versions of `letterFrequency()`, if a character is not found in the ciphertext then assign 0.

- Call both the versions and print both results.

The expected output is provided below.



```
{'b': 1, 'n': 10, 'a': 15, 'c': 2, 'i': 12, 'r': 7, 'e': 15, 'o': 8, 'v': 2, 't': 9, 's': 1, 'p': 4, 'y': 4,
'l': 11, 'm': 3, 'w': 3, 'f': 2, 'd': 6, 'u': 4, 'h': 4}

{'b': 0.008130081300813009, 'n': 0.0813008130081301, 'a': 0.12195121951219515, 'c': 0.016260162601626018,
'i': 0.0975609756097561l, 'r': 0.056910569105691O7, 'e': 0.12195121951219515, 'o': 0.06504065040650407, 'v':
0.016260162601626018, 't': 0.07317073170731708, 's': 0.008130081300813009, 'p': 0.032520325203252036, 'y':
0.032520325203252036, 'l': 0.0894308943089431, 'm': 0.024390243902439025, 'w': 0.024390243902439025, 'f':
0.016260162601626018, 'd': 0.04878048780487806, 'u': 0.032520325203252036, 'h': 0.032520325203252036}
```

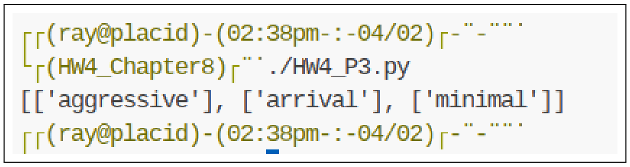
Question-3: Using regular expressions to crack partially encoded strings *(10 points)*

Here is a list of strings called `rep_str`
`'gbson', 'rvuxyz', 'mopqi'`
Here is another list of partially encoded strings called `target`
`'AzzRExxIVE', 'AiiInAL', 'spNpsAL'`

Write a Python script and perform the following.

- Load the `rep_str` and `target` in separate lists and let's call them `scramble` and `partial` respectively.
- Use the function `checkWord2()` and make use of each `scramble` element in the form of a regular expression and break each element in `partial`.
- Use `words.txt` as the supporting word-dictionary for this problem.
- A unique element in `scramble` will be able to break a unique element in `partial`. Make an empty list `final_results` and write a nested for loop to loop over `scramble` and `partial`. Discard the empty lists returned by `checkWord2()`. Otherwise, when the function `checkWord2()` returns a result, append that to the `final_results` list.

The expected output is shown below:



```
┌┌(ray@placid)-(02:38pm-:-04/02)┌-¨-¨¨˙
└┌(HW4_Chapter8)┌¨˙./HW4_P3.py
[['aggressive'], ['arrival'], ['minimal']]
┌┌(ray@placid)-(02:38pm-:-04/02)┌-¨-¨¨˙
```