We will implement a very simple encryption scheme that closely resembles the one-time-pad. You have probably seen this method used in movies like Unknown. The idea is that you and your counterparty share a book whose words you will use as the raw material for a codebook. In this case, you need Metamorphosis, by Franz Kafka.

Your job is to create a codebook of 2-tuples that map to specific words in the given text based on the line and position the words appears in the text. The text is very long so there will be duplicated words. Strip out all of the punctuation and make everything lowercase.

For example, the word **let** appears on line `1683` in the text as the fifth word (reading from left-to-right). Similarly, the word **us** appears in the text on line `1761` as the fifth word.

Thus, if the message you want to send is the following:

```
let us not say we met late at the night about the secret
```

Then, one possible valid sequence for that message is the following:

```
 [(1394, 2), (1773, 11), (894, 10), (840, 1), (541, 2), (1192, 5), (1984, 7), (2112, 6), (1557, 2), (959, 8), (53, 10),
(2232, 8), (552, 5)]
```

Your counterparty receives the above sequence of tuples, and, because she has the same text, she is able to look up the line and word numbers of each of the tuples to retrieve the encoded message. Notice that the word **the** appears twice in the above message but is encoded differently each time. This is because re-using codewords (i.e., 2-tuples) destroys the encryption strength. In case of repeated words, you should have a randomized scheme to ensure that no message contains the same 2-tuple, even if the same word appears multiple times in the message. If there is only one occurrence of a word in the text and the message uses that word repeatedly so that each occurrence of the word cannot have a unique 2-tuple, then the message should be rejected (i.e., assert against this).

Your assignment is to create an encryption function and the corresponding decryption function to implement this scheme. Note that your downloaded text should have 2362 lines and 25186 words in it.

Here are the function signatures:

```
 1 def encrypt_message(message,fname):
 2     '''
 3     Given `message`, which is a lowercase string without any punctuation, and `fname` which is the
 4     name of a text file source for the codebook, generate a sequence of 2-tuples that
 5     represents the `(line number, word number)` of each word in the message. The output is a list
 6     of 2-tuples for the entire message. Repeated words in the message should not have the same 2-tuple.
 7
 8     :param message: message to encrypt
 9     :type message: str
10     :param fname: filename for source text
11     :type fname: str
12     :returns: list of 2-tuples
13     '''
33 def decrypt_message(inlist,fname):
34     '''
35     Given `inlist`, which is a list of 2-tuples`fname` which is the
36     name of a text file source for the codebook, return the encrypted message.
37
38     :param message: inlist to decrypt
39     :type message: list
40     :param fname: filename for source text
41     :type fname: str
42     :returns: string decrypted message
```

Please put your Python code in a Python script file and upload it. Please retain your submitted source files! Remember to use all the best practices we discussed in class. You can use any module in the Python standard library, but third-party modules (e.g., Numpy, Pandas) are restricted to those **explicitly** mentioned in the problem description.

Tips:

- After you have submitted your file, do **not** use the browser back or reload buttons to navigate or open the page in multiple browser tabs, as this may cause your `attempts` to decrease unexpectedly. It may take up to thirty seconds for your code to be processed, so please be **patient**.
- If you find yourself back at the main page without any feedback or change in your `attempts` then it means that your code timed out or crashed in some unexpected way.
- Ensure that your development environment does not presume the existence of certain packages for the autograder. The

autograder does not have anything other than the standard library and those third-party libraries **explicitly** named in the problem description.

- Do not leave extraneous statements in your code like test cases, print statements, or anything else besides what is needed to evaluate your submission because the the autograder will spend its limited time executing those lines, which may result in unexpected crashes or timeouts.

浏览... 0602.py Upload Python source code file

Correct! Back to assignments. functional points = 20 /20 and validation points = 10/10