Assignment 5

due 5/3/2020

We here at CREEPI (Cork Research Establishment for the Exploitation of Personal Information) have a client who has hired us for a spot of data interrogation. For once, we are on the side of the angels and are trying to assist the investigation of dodgy corporate shenanigans.

The target of our investigation is a trove of email from company named Enron and specifically those emails sent by two senior executive named Kenneth Lay and Jeffery Skilling.[1] The email corpus is huge: about 3GB in all with over 150 users and well over half a million individual emails.

Write a Python program that prints out a summary of all of the emails sent by Lay or Skilling. Limit the output to the first 30 such emails found (just to keep down the volume of output). The result should look something like:

```
[11 May 2001] jeff.skilling@enron.com -> sgiridha@prismintl.com
    Subject: Re: Guidance
[12 Jan 2001] jeff.skilling@enron.com -> sherri.sera@enron.com
    Subject: Beard Vote
[12 Jan 2001] jeff.skilling@enron.com -> sherri.sera@enron.com
    Subject: Beard Vote
```

Each email summary should include the date the email was sent, the email address of the sender (either Lay or Skilling), the address of the recipient (or the first recipient in the case of multiple recipients) and the subject line.

Write a Python function `emails_from(suspects)` that lists (i.e. prints) all emails sent by one of these individuals in the format given above. The parameter 'suspects' is dictionary containing the email details of the two individuals as shown.

```
red_flags = {
    "lay-k": ["kenneth.lay@enron.com", "klay@enron.com"],
    "skilling-j": ["skilling@enron.com", "jeff.skilling@enron.com"]
    }
```

In each case the key gives the account/directory name and the value gives a list of the email addresses associated with that account.

If you are feeling adventurous, you may wish to consider a refinement of the above named `emails_between(suspects)` that lists summaries of all emails between the two individuals i.e all emails sent by one suspect and received by another. Don't worry about duplicates (emails lisited twice) or emails send from one of the indivuals to himself. Extend the notion of "recipient" to include those list on the CC line and the BCC line. *This is a bit trickier and is optional.*

[1] Enron was a corporation that was a big player in the US energy sector in the 1980s and 1990s. Its collapse and bankruptcy in 2001 revealed that the company's finances had been a complete sham, deliberatly and cunningly camouflaged by the company's executives over many years. During the trial that followed, the largest fraud case in US corporate history, the email trove was introduced as evidence and so entered the public domain. Lay and Skilling were both convicted and received lengthy prison sentences and so we need not feel too squeamish about poring over their emails.

1. Python has a package that facilitates the parsing of emails. To use this include the following import in your program.

   ```
   from email.parser import Parser
   ```

   The documentation for this package is available here:

   ```
   /docs.python.org/3/library/email.parser.html
   ```

   Use the "plain" parser (not the FeedParser). This allows you to extract various details from an email file such as the sender, date, subject line and so on.

2. The Enron email corpus is available at `/users/shared/enron`. This directory contains the individual email directories of 153 employees. Each employee directory in turn contains a number of subdirectories (e.g. "inbox", "sent" and so on). These subdirectories contain a collection of numbered files each of which contains the text of a single email. We are concerned only with employee directories `lay-k` and `skilling-j`.

3. Experiment with the parsing package before tackling the whole assignment. Pluck a single email out of the collection, use the parser to read and interpret it and make sure you can extract all the relevant information (date, sender and so on).

4. For working with files and directories review the material of Lecture 6. This contains guidance on how to access a particular directory or list the contents of a directory. Again it makes sense to experiment with stepping trough the directories and individual email files before tacking the whole assignment.

5. Some of the emails may contain oddball characters that the email parser baulks at (i.e. throws an exception). Skip those emails. To prevent such emails derailing the program, use a try-except statement to wrap around the the email-parsing statement. See the slides on Exceptions in the "Bits and Pieces" section of the lectures page of the module webpage, if you need to refresh your memory of how the try statement works.