There are two sides to an email message. For the recipient, the burden of email is never-ending. For the sender, the wait is unbearable; "did they get see my email?" to "could it be in their spam folder?" then "when will they reply?" and eventually "what does no answer mean?"

For this assignment, you will make a program to help you handle email responses using your email history data. Your program will tell the sender what to expect based on recent email activity, and help draft a reply message.

1 Email Responder

When an email arrives in your inbox, it can trigger a callback (a function that is called when there is a new email) in a program that you have running. Then this program can do anything you need, like write a response to the sender. Your email responder should:

1. Predict how long your reply will be based on: a) past response rates to that sender, b) how "busy" you are based on your recent email reply times, unread messages, c) your email habits like what time of day you usually reply to emails.
2. Consider incorporating other features of the email message into your prediction (length of the email, what time of date it was sent, whether it's a weekend, etc.).
3. Give some context about how long it takes you to geerally respond to emails, what happened in previous email exchanges with that person (like summary statistics of emails sent). This might be a nice thing for the sender to know.

To compute the prediction, use a regression (e.g., linear regression). Regression is similar to classification in A3, but for predicting a number instead of a category. sklearn can also do regression. So you have some features (items 1 and 2 above), you have some known observations (the actual time it took for you to respond in past cases), and some unknown values you are trying to predict (your expected response time).

Here's a starting point to access to your email history through IMAP (which all email services support). You probably want to get the headers of past emails between you and the sender, along with the timestamps. MailBot might be a good library to trigger an action when you get an email. You can combine this with the IMAP processing code.

Using these libraries together, your email responder should compose an instant reply to the sender about what to expect. Maybe since it's the weekend, and this sender and you take a few days between emails, they might not expect a response until Monday. Or maybe it's Wednesday afternoon, and you responded to a different email 10 minutes ago, so it's likely you'll get to this one right away.

Now you don't have to actually send the response (but you are encouraged to, see the Bonus section below), especially when testing, but write it to a file. Later, check by hand how close your email assistant's predictions were, and whether you think the message as a whole would have been helpful to the sender.

1.1 Email Assistant for Drafting a Response

A second helpful thing is an email assistant that helps you write email responses and save them in your Drafts folder. That way, when you reply, you can just open those drafts, make some edits, and send them off. Be creative with this part!

For example, your email assistant might try to find the most similar email that has been sent to you before, copy your response to that and replace the names so you can use that as a starting point. Alternatively, you could build off an existing reply bot or chat bot.

Hopefully you end up with something you can actually use at the end of this assignment!

Note that this part is only worth 2 points, so if you're short on time, you might want to just get the responder working.

1.2 Tips

Your email assistant should judiciously filter which emails should be responded to and which should not. For example, emails sent to a wide audience (e.g., mailing list) or where you are not in the 'To' or 'CC' fields usually don't need a response. You might also want to look in the email message for words like "action" or "response required".

You probably want to use the content in the email headers to parse the metadata (timestamps and sender/recipient information) rather than download the full messages, especially because email messages can be quite large due to attachments.

Start early! This assignment is purposefully underspecified, so there is room for you to do something creative. There are a few components which might take longer than you expect: setting up the python libraries (or choosing alternate ones), figuring out what to extract from the emails, doing the regression which is similar to part of A3, and coming up with ways to generate email text that you would actually reuse. You are encouraged to work with other students to get the email processing libraries working, but do the rest on your own.

2 Grading

This assignment is worth 15 points: 13 points for the email responder, and 2 points for the email assistant. Check in: by April 21, you should be able to dredge up past email details with that sender and be comfortable using the email-handling libraries above. We'll sort out any issues with email processing in class.

If you are brave enough to actually enable your email responder for a week, you will get two bonus points. You probably want the response messages to make a note that it's part of a class project, that it's an automatically generated email, and be careful not to send emails to the entire department.