# Solution Design

Andrew Winterbotham

This project is a reddit bot that parses the "Today I Learned" (/r/todayilearned/) subreddit and scans for facts containing keywords typed in by the user. If it detects a fact containing one or more of these keywords, an email containing the fact and a link to its source is sent to the user's Gmail address.

After a fact is sent to the user, the id of that submission is appended to a list so that it does not get sent again. In addition, it is also written to a file called already_sent.txt so that when the program is started up again, it knows what facts have already been sent. Each time the program starts up, the ids from the file are put into the list for the program to check each time before an email is sent.

The bot scans reddit every 30 minutes, so that the user is not spammed by too many emails. Rather, whenever they check their email, there will simply be a few emails containing these interesting facts. This project uses PRAW (python api reddit wrapper) to crawl reddit.

I first had to decide on which modules I needed. I knew that my project essentially needed to do two things: first, to crawl reddit scanning for facts containing certain keywords, and second, to then email these facts to the user. For crawling reddit I decided that the PRAW wrapper offered a nice level of abstraction. To send emails, I decided that the smtplib module was the best option.

Thus, to a design a solution to this problem, I realized that I would essentially need two classes: one that crawls reddit for the facts to be emailed to the user, which I called 'RedditBot', and another that sends these emails, which I called 'EmailBot'.

I first set about designing the EmailBot class. For this, I mainly used the smtplib library to set up a connection with a mail server and login to it. I decided to use Gmail for this, as most people have a Gmail account. In order to hide the password, I needed the getpass module. When an instance of this class is created, a connection is automatically set up. The user is asked for their Gmail address and the password for this account. The password is not echoed to the screen and is then destroyed from memory upon entering. The EmailBot class inherits from 'object' and contains one method: send_message. This method is called with two arguments: 'from' and 'subject'.

In the RedditBot class, the send_message method is called with the string 'New fact!!!' for 'from' and the fact extracted from reddit as 'subject'. When the RedditBot class is instantiated, the following is initialized: the user is asked for

the keywords which they want the bot to search for, the reddit api connection is automatically set up, and the list and file containing the ids of the facts already sent are created.

In order to check for profanity in facts before they are sent, the check_profanity method is used. This is essentially the same as that used in the lectures. In addition, the url of the source of the fact is shortened using the shorten method, which uses the online goo.gl url shortener. I then decided to use these two methods in the run method; if the fact contains a keyword, has no profanity and has not already been sent, it is emailed to the user, along with a shortened version of the url of its source.

The main problem I encountered was related to facts containing non-ascii characters. After much searching, I found the smart_str method which is a part of django. Before the non-ascii characters were converted, facts containing these raised errors on the online profanity checker, the goo.gl url shortener and in sending the email with smtplib.

After experimenting with and testing the code, it soon became apparent that I needed some way to deal with the various exceptions that could occur, rather than have the program simply shut down with an obfuscated error message. I thus added various try and exception blocks to each class to deal with the various types of errors that could occur.

Once I had my classes designed, I needed a way to run them only once every thirty minutes. I thus wrote a function to instantiate and run them, in a separate file called fact_sender.py, which also contained a way to deal with a sudden crash of the program; if the program suddenly shuts down, a message is printed and the file containing the ids of the facts already sent is closed and saved, so that the next time the program is run, it knows which facts have already been sent; these facts are loaded into memory using a list. To make the program sleep for thirty minutes, I used the sleep method from the time module, and to deal sudden crashes, I used the atexit module.

Only a certain number of emails will be sent each time the program runs; this is dictated by a while loop in the main function in the file fact_sender.py.

The pseudocode for each class is as foillows:

**EmailBot:**

- The following is done automatically (when a class instance is created):
  - Ask for the password and the username, and set up the connection with the Gmail server. If that doesn't work, print an error message and exit
  - If that works, login to the server with those details
  - If the wrong password or username are entered, let the user know, and then exit
- send_message method:
  - The headers for the email are specified (who the email is to and from and its subject etc: it is sent to and from the user themselves)

- The email is then sent to the account
- If an error occurs, an error message is printed and the program will gracefully exit

**RedditBot:**

- Automatically done:
  - Asks for keywords
  - Puts each one into a list
  - Creates a file and list containg the ids of facts already emailed
  - Sets up a connection with our subreddit
- check_profanity method:
  - Adds the whole fact to the url for whick we check profanity for
  - If there is profanity: return the value 'True'
  - Otherwise, return the value 'False'
  - If there is an error in checking, return the value 'True' (so that the fact is not emailed)
- shorten method:
  - Specify the information and url to be sent to the onliune shortener
  - If we get no error, return the shortened url
  - Otherwise, just return the original, unshortened url
- run method:
  - goes through each fact, checking if for the keyword(s)
  - if a fact contains one of them or more, check for profanity (using the check_profanity method) and if it has been sent already
  - if it has no profanity and hasn't been sent already, email it along with the url of its source to the user, andf end the mwthod
  - Otherwise, continue looking
  - If it cannot send, it is treated as already sent, so we do not try to send it again, getting the same error

The class diagrams for the two classes are contained in the files EmailBot.pdf and RedditBot.pdf.