CS2910 Lab 5: Sending Email with SMTP

Lab Assignment

This is a team assignment; each team should be two members unless a different size is approved by the instructor.

Installing pytz

1. Go to https://pypi.python.org/pypi/setuptools#windows-simplified
2. Download ez_setup.py to your computer.
3. Add C:\Python27\Scripts to your system path (Ask me how if you have questions)
4. Open a command prompt **as an administrator**. (Ask me how if you have questions)
5. Change into the directory where you downloaded ez_setup.py
6. Run `python ez_setup.py` (**as administrator**)
7. Run (from any directory) `easy_install --upgrade pytz` (**as an administrator**).

You can then install future packages by repeating just the steps 4. and 7.

Introduction

The goal of this lab is to write a short **Python** program, to send an email message to an email server, using the SMTP protocol, STARTTLS security, and AUTH LOGIN authentication.

Procedure

1. Download the skeleton Python template: smtp.py
2. Edit the header of the file to include your team members' names, in the format provided.
3. Complete the **smtp_send** method to send an email message with at least the following content:
    - Email header
        - From
        - To (one recipient; optionally, you may add more by using a list of strings for this value)
        - Subject
        - Date
    - Email text (may be multiple lines)

   Optionally, you may add other capabilities, such as sending an attachment, but make sure that all the required functionality is completely working first.

You will want to add other helper methods, but do not change the any other code provided in the template.

4. Add comments *at the end* of your Python file, with the following information:
    - A description of the functionality you implemented and the results of your testing.
    - Comments on your experience in completing the lab, including any problems you encountered. Briefly explain what you learned.
    - Questions and suggestions.

You **may not** use a prebuilt library like **Lib/smtplib**; the point of this lab is for you to understand the low-level implementation of the SMTP protocol.

Additional implementation details will be discussed in class. If you have questions about these requirements, ask in class or lab.

If this base functionality turns out to be too easy, you may experiment with adding additional functions, but be sure the basic requirements are still met. Make sure you always have a working backup copy before adding features.

Try to divide up the primary responsibility for parts of the program in an equitable way.

Hints and Notes

- As in the HTTP labs, you will have to both send and receive on the TCP connection to the SMTP client. By now, you should probably have a "read line" function, and it may be helpful to have a function that can send an SMTP command and return the response (status code and accompanying data) in a form that is easy to search for a particular response line's "value".

- By now, you should know how to read from a network stream; if you still have questions, ask the instructor for assistance.

- Getting the proper SMTP "Date" value can be a little tricky. For now, just use the provided get_formatted_date() function.

- To convert a non-secure TCP connection into an SSL/TLS secure connection, you can "wrap" the existing TCP socket:

  ```
  skt = ssl.wrap_socket(skt_nonssl, ssl_version=ssl.PROTOCOL_SSLv3)
  ```

  In this example, after wrapping the original skt_nonssl socket, you would switch over to using the skt socket for the subsequent secure operations.

  For more information, including different protocol versions (which may be needed for some

SMTP servers), see docs.python.org/2/library/ssl.html. The available protocols vary by Python version; for Python 2.7.5, the available options are:

  - PROTOCOL_SSLv2
  - PROTOCOL_SSLv23
  - PROTOCOL_SSLv3
  - PROTOCOL_TLSv1

  You can see which protocol versions are available by typing the following command in an interactive Python window:

```
>>> import ssl
>>> dir (ssl)
```

  Look for the symbols that start with "PROTOCOL_".

- You can find documentation of email header elements in RFC 5322, titled Internet Message Format.
- When you have questions you can't resolve, consult the instructor as soon as possible, in person or by email.

Submission (Due Friday, Week 7, 11PM)

One team member should submit your Python file by uploading it to the upload page (which is also linked from the Schedule).

(Acknowledgements: The original version of this lab written by Dr. Sebern.)