Overview of Received Emails

Now that we understand how to send emails progammatically with Python, let's explore how we can read and search recieved emails. To do we will use the built-in imaplib library. We will also use the built in email library for parsing through the recieved emails.

In [2]:
```
import imaplib
```

In [3]:
```
M = imaplib.IMAP4_SSL('imap.gmail.com')
```

In [4]:
```
import getpass
```

In [ ]:
```
user = input("Enter your email: ")
```

In [9]:
```
# Remember , you may need an app password if you are a gmail user
#
password = getpass.getpass("Enter your password: ")
```

```
Enter your password: ········
```

In [ ]:
```
M.login(user,password)
```

In [11]:
```
M.list()
```

Out[11]:
```
('OK',
 [b'(\\HasNoChildren) "/" "INBOX"',
  b'(\\HasNoChildren) "/" "Personal"',
  b'(\\HasNoChildren) "/" "Receipts"',
  b'(\\HasNoChildren) "/" "Sent"',
  b'(\\HasNoChildren) "/" "Trash"',
  b'(\\HasNoChildren) "/" "Travel"',
  b'(\\HasNoChildren) "/" "Work"',
  b'(\\HasChildren \\Noselect) "/" "[Gmail]"',
  b'(\\All \\HasNoChildren) "/" "[Gmail]/All Mail"',
  b'(\\Drafts \\HasNoChildren) "/" "[Gmail]/Drafts"',
  b'(\\HasNoChildren \\Important) "/" "[Gmail]/Important"',
  b'(\\HasNoChildren \\Sent) "/" "[Gmail]/Sent Mail"',
  b'(\\HasNoChildren \\Junk) "/" "[Gmail]/Spam"',
  b'(\\Flagged \\HasNoChildren) "/" "[Gmail]/Starred"',
  b'(\\HasNoChildren \\Trash) "/" "[Gmail]/Trash"'])
```

In [12]:
```
# Connect to your inbox
M.select("inbox")
```

Out[12]:
```
('OK', [b'28297'])
```

Searching Mail

Now that we have connected to our mail, we should be able to search for it using the specialized syntax of IMAP. Here are the different search keys you can use:

```
<tr>
    <td>'BEFORE date'</td>
    <td>
    Returns all messages before the date. Date must be formatted as 01-Nov-
2000.
    </td>
</tr>

 <tr>
    <td>'ON date'</td>
    <td>
    Returns all messages on the date. Date must be formatted as 01-Nov-2000.
    </td>
</tr>

 <tr>
    <td>'SINCE date'</td>
    <td>
    Returns all messages after the date. Date must be formatted as 01-Nov-
2000.
    </td>
</tr>

<tr>
    <td>'FROM some_string '</td>
    <td>
    Returns all from the sender in the string. String can be an email, for
example 'FROM               user@example.com' or just a string that may
appear in the email, "FROM example"
    </td>
</tr>

<tr>
    <td>'TO some_string'</td>
    <td>
    Returns all outgoing email to the email in the string. String can be an
email, for example 'FROM user@example.com' or just a string that may appear
in the email, "FROM example"
    </td>
</tr>

<tr>
    <td>'CC some_string' and/or 'BCC some_string'</td>
    <td>
    Returns all messages in your email folder. Often there are size limits
from imaplib.
    To change these use imaplib._MAXLINE = 100 , where 100 is whatever you
want the limit to be.
    </td>
</tr>
```

```
    <tr>
        <td>'SUBJECT string','BODY string','TEXT "string with spaces"'</td>
        <td>
        Returns all messages with the subject string or the string in the body
    of the email. If the string you are searching for has spaces in it, wrap it
    in double quotes.
        </td>
    </tr>

    <tr>
        <td>'SEEN', 'UNSEEN'</td>
        <td>
        Returns all messages that have been seen or unseen. (Also known as read
    or unread)
        </td>
    </tr>


        <tr>
        <td>'ANSWERED', 'UNANSWERED'</td>
        <td>
        Returns all messages that have been replied to or unreplied to.
        </td>
    </tr>


        <tr>
        <td>'DELETED', 'UNDELETED'</td>
        <td>
        Returns all messages that have been deleted or that have not been
    deleted.
        </td>
    </tr>
```

</table>

You can also use the logical operators AND and OR to combine the above statements. Check out the full list of search keys here: http://www.4d.com/docs/CMU/CMU88864.HTM.

Please note that some IMAP server providers for different email services will have slightly different syntax. You may need to experiment to get the results you want.

Now we can search our mail for any term we want.

In [15]:
```python
# Use if you get an error saying limit was reached
imaplib._MAXLINE = 10000000
```

Send yourself a test email with the subject line:

```
this is a test email for python
```

Or some other uniquely identifying string.

We will now need to reconnect to our imap server. You will probably need to restart your kernel for this step if you are using jupyter notebook.

In [ ]:
```python
# Restart your kernel and run the following:
import imaplib
import getpass
M = imaplib.IMAP4_SSL('imap.gmail.com')
user = input("Enter your email: ")
password = getpass.getpass("Enter your password: ")
M.login(user,password)
```

In [2]:
```python
# Connect to your inbox
M.select("inbox")
```

Out[2]:
```
('OK', [b'28299'])
```

Let's now search and confirm if it is there:

In [105…
```python
typ ,data = M.search(None,'SUBJECT "this is a test email for python"')
```

We can now save what it has returned:

In [106…
```python
typ
```

Out[106]:
```
'OK'
```

In [107…
```python
data
```

Out[107]:
```
[b'28298']
```

The data will be a list of unique ids.

In [108…
```python
# typ, data = M.fetch(data[0],"(RFC822)")
```

In [112…
```python
result, email_data = M.fetch(data[0],"(RFC822)")
```

In [113…
```python
raw_email = email_data[0][1]
```

In [116…
```python
raw_email_string = raw_email.decode('utf-8')
```

We can use the built in email library to help parse this raw string.

In [120…
```python
import email
```

In [121…
```python
email_message = email.message_from_string(raw_email_string)
```

In [125…
```python
for part in email_message.walk():
    if part.get_content_type() == "text/plain":
        body = part.get_payload(decode=True)
        print(body)
```

```
b'This is a test to see if the python search worked.\r\n'
```

Excellent! We've successfully have been able to check our email's inbox , filter by some condition, and read the body of the text that was there. This will come in handy in the near future!

Keyword	**Definition**
'ALL'	Returns all messages in your email folder. Often there are size limits from imaplib. To change these use imaplib._MAXLINE = 100 , where 100 is whatever you want the limit to be.