HIGH-LEVEL NETWORKING CLASSES

Python includes several specialized networking modules for application-layer protocols built on the standard socket module. The available modules provide implementations of HTTP, SMTP, IMAP(Internet message access protocol) and POP3, NNTP(network news transfer protocol), XML-RPC(remote procedure call) & FTP.

Useful application-layer protocol modules include:

-The httplib (HTTP Client)

The HTTP client interface is useful when developing Web robots or other Internet scraping agents. The Web protocol is request/response in nature over stream sockets. The httplib module can be demonstrated by creating a new HTTP client instance with HTTPConnection which provides the Website to connect to.

With the new object request are made using request method and specification of the HTTP GET method which requests files from the server. The get response method parses the HTTP response header to understand if an error has occurred. Once successful the read method on new response object returns & prints the text. Building a simple (non-rendering) HTTP client with httplib:

```
import httplib

httpconn = httplib.HTTPConnection("www.ibm.com")

httpconn.request("GET", "/delopmentworks/index.html")

resp = httpconn.getresponse()

if resp.reason == "ok":
     resp_data = resp.read()
     print resp_data
httpconn.close()
```

-The smtplib (SMTP Client)

SMTP allows one to send email messages to a mail server, this is useful in networking systems to relay status about the operation of a device. In Python, it's simple and it consists of creating an SMTP object, sending and email message using the sendemail method then closing with the quit method:

```
import smtplib

fromAdrs = 'me@mail.com'
toAdrs = 'yours@mail.com'
msg = 'From: me@mail.com\r\nTO: yours@mail.com\r\nSubject:Hello'

mailClient = smtplib.SMTP('192.168.1.1')
mailClient.sendmail( fromAdrs, toAdrs, msg )
mailClient.quit
```

-The poplib (POP3 Client)

The POP3 protocol allows one to connect to a mail server & download new email which is useful for remote commanding by embedding commands within the body of an email. After the command smtplib can be used to return a response email to the source.

The demonstration will shows a simple application that connects to a mail server & emits the subject lines for all pending email for the user. The poplib offers several methods for gathering & managing email at the server. A new POP3 object is created with the POP3 method by specifying the mail server. The user and pass_ methods authenticate the application to the server , the stat method returns the number of messages waiting for the user & the total number of bytes taken up by all messages.

Each available message will be looped through using the retr method to grab the next email message, this methods returns a list in the form (response, ['line, ……], octets). Where response is the POP3 response for the particular message, the line list represents the individual lines of the email message and the octets is the number of bytes for the email message. The inner loop simply iterates over the second element([1]) of the list which is the list of the email message body. The Subject is checked if present and then it's printed.

After all email messages have been checked, a call to the quit method ends the POP3 session. The top method can be used instead of retr method to extract the header for the email messages. This method is faster & minimizes the amount of data transferred to the client:

```
import poplib
import re

popClient = poplib.POP3('192.168.1.1')

popClient.user('user')
popClient.pass_('password')

numMsgs, mboxSize = popClient.stat()

print "Number of messages ", numMsgs
print "Mailbox size", mboxSize
print

for id in range (numMsgs):
     for mail in popClient.retr(id+1)[1]:
          if re.search( 'Subject:', mail ):
               print mail
     print

popClient.quit()
```