套接字编程作业 2-3：邮件客户

本作业要求了解 SMTP 协议，并使用 Python 实现 SMTP 标准协议。Python 提供了一个名为 smtplib 的模块，它内置了使用 SMTP 协议发送邮件的方法。但是我们不会在本实验中使用此模块，因为它隐藏了 SMTP 和套接字编程的细节。

在这个编程作业中，你要开发一个简单的邮件客户端软件，将邮件发送给任意收件人。你的邮件客户端将需要通过 TCP 连接到邮件服务器，使用 SMTP 协议与邮件服务器进行交互，经该邮件服务器向某接收方发送一封电子邮件，最后关闭与该邮件服务器的 TCP 连接。

为了限制垃圾邮件，一些邮件服务器不接受来源随意的 TCP 连接，建议使用你的大学邮件服务器和流行的 Webmail 服务器（如 QQ 或 163 邮件服务器）。

任务 1、编写邮件客户端代码

请在“SMTP 客户端框架代码.py”中标有#Fill-in-start 和#Fill-in-end 的地方填写代码，每个地方都可能需要不止一行代码。

框架代码中的“mailserver”变量值是发件人邮箱的 SMTP 服务器域名（学校邮箱为 mail.std.uestc.edu.cn，QQ 邮箱为 smtp.qq.com，163 邮箱为 smtp.163.com），SMTP 默认端口号是 25。

QQ 邮箱和 163 邮箱默认关闭 SMTP 服务，需要在设置中打开 SMTP 服务。另外，QQ 邮箱和 163 邮箱在打开 SMTP 服务后，会设置一个授权码，在程序使用这个授权码作为密码（代码中的“password”变量）登录，而不是平时使用的密码。

任务 2、运行邮件客户端代码

在命令行窗口运行你编写的邮件客户端程序代码，查看服务器返回的每条消息，其中包含每次操作后返回的状态码。

同时，登录发件人邮箱和收件人邮箱，在发件人的已发送文件夹中和收件人的收件箱中查看这封被发送的邮件。

在某些情况下，收件人邮件服务器可能会将你的电子邮件分类为垃圾邮件。如果你在收件人的收件箱中没有看到这封电子邮件，请检查垃圾邮件文件夹。

任务 3*、提供附件发送支持（选做）

任务 1 实现的 SMTP 邮件客户端只能在电子邮件正文中发送文本消息。修改你的邮件客户端代码，使其可以发送包含文本文件和图像文件的电子邮件。

作业提交要求

任务 1 和 2 必做，任务 3 选做。

将以下内容压缩打包后提交，压缩文件的命名规则为“作业 2-3-学号-姓名”：

1）任务 1 的完整邮件客户端代码文件，提交时请用***替换发件人邮件账户口令

2）运行结果报告文件，包含：

- Python 版本
- 根据任务 2 中邮件客户端程序运行中服务器返回的消息，画出本作业实现的邮件客户端与发件人邮箱的 SMTP 邮件服务器之间的交互时序图
- 任务 2 的邮件客户端程序运行窗口截图
- 任务 2 中发件人邮箱的已发送文件夹中这封被发送的邮件截图
- 任务 2 中收件人邮箱的收件箱中这封被发送的邮件截图。

3）可选提交：任务 3 的邮件客户端代码文件 1 份，以及运行结果报告 1 份（包含客户端和服务器程序运行窗口截图）

SMTP 客户端框架代码

```
from socket import *
import base64

# Mail content
subject = "I love computer networks!"
contenttype = "text/plain"
msg = "I love computer networks!"
endmsg = "\r\n.\r\n"

# Choose a mail server (e.g. Google mail server) and call it mailserver
mailserver =    #Fill-in-start    #Fill-in-end

# Sender and reciever
fromaddress =    #Fill-in-start    #Fill-in-end
toaddress =    #Fill-in-start    #Fill-in-end

# Auth information (Encode with base64)
username =      #Fill-in-start    #Fill-in-end
password =    #Fill-in-start    #Fill-in-end

# Create socket called clientSocket and establish a TCP connection with mailserver
#Fill-in-start
#Fill-in-end

recv = clientSocket.recv(1024) .decode()
print(recv)

# Send HELO command and print server response.
#Fill-in-start
#Fill-in-end

# Send AUTH LOGIN command and print server response.
clientSocket.sendall('AUTH LOGIN\r\n'.encode())
recv = clientSocket.recv(1024).decode()
print(recv)

clientSocket.sendall((username + '\r\n').encode())
recv = clientSocket.recv(1024).decode()
print(recv)

clientSocket.sendall((password + '\r\n').encode())
recv = clientSocket.recv(1024).decode()
print(recv)

# Send MAIL FROM command and print server response.
#Fill-in-start
#Fill-in-end

# Send RCPT TO command and print server response.
#Fill-in-start
#Fill-in-end

# Send DATA command and print server response.
#Fill-in-start
#Fill-in-end
```

```
# Send message data.
#Fill-in-start
#Fill-in-end

# Message ends with a single period and print server response.
#Fill-in-start
#Fill-in-end

# Send QUIT command and print server response.
#Fill-in-start
#Fill-in-end

# Close connection
clientSocket.close()
```