****

**课程实验报告**

|  |  |  |  |
| --- | --- | --- | --- |
| **课程编号** |  | **课程名称** | **计算机网络** |
| **实验学分** |  | **任课教师** | **叶松涛** |
| **学生学号** | **190320020** | **行政班级** | **信科1901** |
| **学生姓名** | **田章** | **联系方式** | **17680314099** |

**20 21 -2022学年度2学期**

**教务处制**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **实验** | | | | |
| **实验编号** | **实验3** | **实验名称** | **邮件客户** | |
| **指导教师** | **叶松涛** | **实验日期** | **2022.5.26** | |
| **实验同组人** | **未分组** |  |  | |
| 1. **实验目的和要求**   实验目的：更加了解SMTP协议。您还将学到使用Python实现标准协议的经验。  实验要求：1.开发一个简单的邮件客户端，将邮件发送给任意收件人。2.客户端将需要连接到邮件服务器，使用SMTP协议与邮件服务器进行对话，并向邮件服务器发送电子邮件。3.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。 | | | | |
| 1. **实验设备、软件**   编辑器VScode  Python环境：3.10.1  谷歌浏览器：开启QQ邮箱的SMTP服务，登录谷歌邮箱查看发送端发出的邮件信息 | | | | |
| 1. **实验内容（步骤、计划、方案）**  * **准备工作**：   **SMTP**：简单邮递传送协议（英语：Simple Mail Transfer Protocol，缩写：SMTP）可用在传送和接收电子邮件的信息，但SMTP通常用作传送电子邮件信息，而不是接收。  **IMAP**：即Internet Message Access Protocol（互联网邮件访问协议），可以通过这种协议从邮件服务器上获取邮件的信息、下载邮件等。IMAP与POP类似，都是一种邮件获取协议。  **IMAP和POP的区别**：POP允许电子邮件客户端下载服务器上的邮件，但是在电子邮件客户端的操作（如：移动邮件、标记已读等），这是不会反馈到服务器上的。但是IMAP电子邮件客户端的操作都会反馈到服务器上，对邮件进行的操作，服务器上的邮件也会做相应的动作。同时，IMAP可以只下载邮件的主题，只有当真正需要的时候，才会下载邮件的所有内容。  **因此，我们开启的是IMAP/SMTP服务**  **说明**：原本打算使用谷歌邮箱作为发送端，QQ邮箱作为接收端，这样就可以用VScode运行发送，然后手机登录QQ邮箱来查看邮件内容。但是由于网络原因，邮件客户端设置成谷歌邮箱一直发送失败，因此我们使用QQ邮箱作为发送端，谷歌邮箱作为接收端。  我们是开发一个邮件客户端用于发送，因此我们需要开启发送端QQ邮箱的SMTP服务。由于IMAP和SMTP是一起同时开启与关闭的，因此我们都开启就行。  **开启QQ邮箱的SMTP**  用谷歌浏览器打开QQ邮箱官网，进入设置，账户，下划开启IMAP/SMTP服务，通过绑定的手机验证来获取授权码。     * **发送端与接收端的一些设置**   由于我们选择的是基于QQ邮箱发送的，因此需要指定邮件服务器为**"smtp.qq.com"，**另外还需要指定发送端和接收端的邮箱地址名。作为一个邮件客户端。发送邮件之前我们需要登录操作，因而还需要配指明用户名与密码，用户名即为我们发送端的邮箱地址（因为已经指定了邮箱服务器为QQ邮箱，因此后缀可以省略），需要特别注意的是，基于SMTP登录的客户端密码为IMAP/SMTP服务的授权码。    本题没有要求使用SSL安全连接，因此端口号设置为25。   * 仿照题目给出的部分代码，通过谷歌浏览器查找发送相应命令返回的响应状态码，我们可以将其余命令的代码补上。另外，我们在向服务器验证身份成功后，需要发送用户名以及密码，这里为了安全，我们对用户名密码进行编码后再使用base64进行加密处理。 * **电子邮件基本格式**   总的来说，参照rfc4021标准，MIME格式分为邮件头和邮件体两部分  **邮件头**  邮件头包含了发件人、收件人、主题、时间、MIME版本、内容的类型、内容的传输编码方式等重要信息。每条信息称为一个域。  MIME邮件的基本信息、格式信息、编码方式等重要内容都记录在邮件内的各种域中，域的基本格式：{域名}：{内容}，域由域名后面跟“：”再加上域的信息内容构成，一条域在邮件中占一行或者多行，域的首行左侧不能有空白字符，比如空格或者制表符，占用多行的域其后续行则必须以空白字符开头。域的信息内容中还可以包含属性，属性之间以“;”分隔，属性的格式如下：{属性名称}=”{属性值}”。  **邮件体**  邮件内容有各种各样的（即纯文本，超文本，内嵌资源（比如内嵌在超文本中的图片），附件的组合），服务器通过第一个content-type判断邮件内容，如果包含了其他内容，邮件体被分为多个段，段中可包含段，每个段又包含段头和段体两部分。  **代码MailClient.py见附录** | | | | |
| 1. **实验原始记录、结果分析（可以附页）**   **运行MailClient.py文件，输出如下**    **没有类似220 reply not received from server.的输出语句，因此程序是正常运行的，这时候我们查看谷歌邮箱的收件箱。**    **可以看到我们是正常收到了相应的发送信息。** | | | | |
| 1. **实验思考题及其解答**   **练习1.** 类似Google邮件的服务器（如地址：smtp.gmail.com，端口：587））要求您的客户端在发送MAIL FROM命令之前，需要为了身份验证和安全原因添加传输层安全（TLS）或安全套接字层（SSL）。将TLS / SSL命令添加到现有的命令中，并使用上述地址和端口为Google邮件服务器实现客户端。  **思考**：使用SSL安全连接，我们在实验3的代码上稍作修改即可，首先需要导入ssl模块。因为使用的SSL连接,而QQ邮箱的SSL协议端口号是465,因此我们需要将实验3中的端口号修改为465。使用默认参数获取一个SSLContext实例contex = ssl.create\_default\_context()，然后对套接字进行封装clientSocketSSL = context.wrap\_socket(clientSocket, server\_hostname=mailserver)，封装后返回clientSocketSSL，然后将实验3中的clientSocket全部替换成clientSocketSSL，从而就完成了练习1的全部代码。  **练习1代码见附录，运行截图见下**：      **练习2.** 您当前的SMTP邮件客户端只能在电子邮件正文中发送文本消息。修改您的客户端，使其可以发送包含文本和图像的电子邮件。  思考：先准备文本和图像，这里我们使用超文本文件hello.html  <!DOCTYPE html>  <html lang="zh-CN">  <head>      <meta charset="UTF-8">      <meta http-equiv="X-UA-Compatible" content="IE=edge">      <meta name="viewport" content="width=device-width, initial-scale=1.0">      <title>首页</title>  </head>  <body>      <h1>刘亦菲</h1>      网上随便搜的一张刘亦菲的照片，发给你咯。  </body>  </html>  图像名为 刘亦菲.jpg    **我们将两个文件与代码放在同一文件夹下面。**  我们在练习1的基础上对代码中的msg进行修改。  通过实验3，我们已经知道了右箭头的一些域，这里我们需要发送的并不是普通文本，因此需要指定Content-Type域的值。  Content-Type域定义了邮件中所含各种内容的类型以及相关属性。邮件所含的文本、超文本、附件等信息都按照对应Content-Type域所指定的媒体类型、存储位置、编码方式等信息存储在邮件中。Content-Type域基本格式：Content-Type：{主类型}/{子类型}。  要将超文本和图像一起通过邮箱发送，我们需要设置Content-Type 的主类型为multipart。因为我们使用了超文本和图片，因此需要设置 Content-Type:multipart/related    请求头的Content-Type属性指定为multipart/related，就还需要定义boundary属性。   * boundary的值必须以英文中间双横杠--开头，这个--称为前导连字符;boundary的值除了前导连字符以外的部分不能超过70个字符;boundary的值不能包含HTTP协议或者URL禁用的特殊意义的字符。 * 请求体中的请求行数据是由多部分组成，boundary属性的值模式--${boundary}用于分隔每个独立的分部。 * 每个部分可以单独定义Content-Type和该部分的数据体 * 请求体以boundary参数的值模式--${boundary}--作为结束标志 * 每个--${boundary}之前默认强制必须为CRLF，如果某一个部分的文本类型请求体以CRLF结尾，那么在请求体的二级制格式上，必须显式存在两个CRLF，如果某一个部分的请求体不以CRLF结尾，可以只存在一个CRLF，这两种情况分别称为分隔符的显式类型和隐式类型。   这里我们设置boundary=“--tianzhang”即可。后面两个分部分别是发送html超文本内容和jpg图像，因此分别设置Content-Type为text/html和image/jpeg。  由于我们并不是发送的普通文本，因此还需要指定编码方式。  MIME邮件可以传送图像、声音、视频以及附件，这些非ASCII码的数据都是通过一定的编码规则进行转换后附着在邮件中进行传递的。编码方式存储在邮件的Content-Transfer-Encoding域中，一封邮件中可能有多个Content-Transfer-Encoding域，分别对应邮件不同部分内容的编码方式。目前MIME邮件中的数据编码普遍采用Base64编码或Quoted-printable编码来实现。这里我们使用base64编码。  **完整代码见附录，运行截图如下：**    通过输出的结果可以看出，邮件是正常发送，因此我们打开谷歌浏览器登录谷歌邮箱查看收件箱 | | | |
| 1. **实验评分**   **教师签名：**    **年 月 日** | | | |

**附录：**

**实验3完整代码**

from socket import \*

import base64

# Choose a mail server (e.g. Google mail server) and call it mailserver

mailserver = "smtp.qq.com"

# 发送方地址和接收方地址，from 和 to

mailFromAddress = 'tianzhang.chn@qq.com'

mailToAddress = 'tianzhang.chn@gmail.com'

# 用户名,由于我们指定了邮箱服务器为qq,因此是可以省略后缀的,当然也可以用mailFromAddress

username = mailFromAddress

# 此处不是自己的密码，而是开启SMTP服务时对应的授权码

password = 'ybtlpzfewslqdcdi'

# Create socket called clientSocket and establish a TCP connection with mailserver

*'''*

*(class) socket(family: AddressFamily | int = ..., type: SocketKind | int = ..., proto: int = ..., fileno: int | None = ...)*

*family(给定的套接族)一般有两种重要参数：*

*socket.AF\_INET  用于服务器与服务器之间的网络通信*

*socket.AF\_INET6 基于IPV6方式的服务器与服务器之间的网络通信*

*type(套接字类型)，也是一般两个类型：*

*socket.SOCK\_STREAM  基于TCP的流式socket通信*

*socket.SOCK\_DGRAM   基于UDP的数据报式socket通信*

*'''*

clientSocket = socket(AF\_INET, SOCK\_STREAM)

# 常用邮箱的服务器(SMTP/POP3)地址和端口整理 https://blog.51cto.com/u\_15300443/3091999

# 或者由于我们直接进QQ邮箱官网 https://service.mail.qq.com/cgi-bin/help?subtype=1&id=20010&no=1000557

# 因此端口号我们选择25

serverPort = 25

# connect只能接收一个参数

clientSocket.connect((mailserver, serverPort))

# 从客户套接字中接收信息

# 220 服务就绪，可以执行新用户的请求。

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

print(recv)

if recv[:3] != '220':

    print('220 reply not received from server.')

# Send HELO command and print server response.

# 开始与服务器的交互，服务器将返回状态码250,说明请求动作正确完成

heloCommand = 'HELO mailserver\r\n'

clientSocket.send(heloCommand.encode())

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

print(recv)

if recv[:3] != '250':

    print('250 reply not received from server.')

# 发送"AUTH LOGIN"命令，验证身份.服务器将返回状态码334（服务器等待用户输入验证信息）

loginCommand = 'AUTH LOGIN\r\n'

clientSocket.send(loginCommand.encode())

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

print(recv)

if recv[:3] != '334':

    print('334 reply not received from server.')

# 邮箱账户经过base64编码

# base64编码后是一些bytes字符串,因此我们加换行符也需要+ b'\r\n'

userCommand = base64.b64encode(username.encode()) + b'\r\n'

clientSocket.send(userCommand)

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

print(recv)

if recv[:3] != '334':

    print('334 reply not received from server')

# 邮箱密码经过base64编码

passCommand = base64.b64encode(password.encode()) + b'\r\n'

clientSocket.send(passCommand)

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

print(recv)

# 如果用户验证成功，服务器将返回状态码235

if recv[:3] != '235':

    print('235 reply not received from server')

# Send MAIL FROM command and print server response.

# TCP连接建立好之后，通过用户验证就可以开始发送邮件。邮件的传送从MAIL命令开始，MAIL命令后面附上发件人的地址。

# 发送 MAIL FROM 命令，并包含发件人邮箱地址

MFCommand = 'MAIL FROM: <'+ mailFromAddress + '>\r\n'

clientSocket.send(MFCommand.encode())

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

print(recv)

if recv[:3] != '250':

    print('250 reply not received from server.')

# Send RCPT TO command and print server response.

# 接着SMTP客户端发送一个或多个RCPT (收件人recipient的缩写)命令，格式为RCPT TO: <收件人地址>。

# 发送 RCPT TO 命令，并包含收件人邮箱地址，返回状态码 250

RTCommand = 'RCPT TO: <'+ mailToAddress + '>\r\n'

clientSocket.send(RTCommand.encode())

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

print(recv)

if recv[:3] != '250':

    print('250 reply not received from server.')

# Send DATA command and print server response.

# 发送 DATA 命令，表示即将发送邮件内容。服务器将返回状态码354（开始邮件输入，以"."结束）

DATACommand = 'DATA\r\n'

clientSocket.send(DATACommand.encode())

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

print(recv)

if recv[:3] != '354':

    print('354 reply not received from server.')

# Send message data.

msg = 'From: ' + mailFromAddress + '\r\n'

msg += 'To: ' + mailToAddress +  '\r\n'

msg += 'Subject: ' + '测试' +  '\r\n'

msg += "\r\n I love computer networks!"

clientSocket.send(msg.encode())

# Message ends with a single period.

# 以"."结束。请求成功返回 250

endmsg = "\r\n.\r\n"

clientSocket.send(endmsg.encode())

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

print(recv)

if recv[:3] != '250':

    print('250 reply not received from server.')

# Send QUIT command and get server response.

QUITCommand = 'QUIT\r\n'

clientSocket.send(QUITCommand.encode())

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

print(recv)

# 221 服务关闭控制连接。

if recv[:3] != '221':

    print('221 reply not received from server.')

clientSocket.close()

**练习1完整代码**

from socket import \*

import base64

import ssl

# Choose a mail server (e.g. Google mail server) and call it mailserver

mailserver = "smtp.qq.com"

mailFromAddress = 'tianzhang.chn@qq.com'

mailToAddress = 'tianzhang.chn@gmail.com'

username = mailFromAddress

password = 'ybtlpzfewslqdcdi'

# Create socket called clientSocket and establish a TCP connection with mailserver

# 这里使用的SSL连接,qq邮箱的SSL协议端口号是465,参考资料下方超链接

# https://service.mail.qq.com/cgi-bin/help?subtype=1&id=20010&no=1000557

serverPort=465

# 直接使用默认参数获取一个SSLContext实例

context = ssl.create\_default\_context()

clientSocket = socket(AF\_INET, SOCK\_STREAM)

clientSocket.connect((mailserver, serverPort))

# 封装套接字

clientSocketSSL = context.wrap\_socket(clientSocket, server\_hostname=mailserver)

recv = clientSocketSSL.recv(1024).decode()

print(recv)

if recv[:3] != '220':

    print('220 reply not received from server.')

# Send HELO command and print server response.

heloCommand = 'HELO mailserver\r\n'

clientSocketSSL.send(heloCommand.encode())

recv = clientSocketSSL.recv(1024).decode()

print(recv)

if recv[:3] != '250':

    print('250 reply not received from server.')

# 登录过程

loginCommand = 'AUTH LOGIN\r\n'

clientSocketSSL.send(loginCommand.encode())

recv = clientSocketSSL.recv(1024).decode()

print(recv)

if recv[:3] != '334':

    print('334 reply not received from server.')

# 邮箱账户经过base64编码

userCommand = base64.b64encode(username.encode()) + b'\r\n'

clientSocketSSL.send(userCommand)

recv = clientSocketSSL.recv(1024).decode()

print(recv)

if recv[:3] != '334':

    print('334 reply not received from server')

# 邮箱密码经过base64编码

passCommand = base64.b64encode(password.encode()) + b'\r\n'

clientSocketSSL.send(passCommand)

recv = clientSocketSSL.recv(1024).decode()

print(recv)

if recv[:3] != '235':

    print('235 reply not received from server')

# Send MAIL FROM command and print server response.

MFCommand = 'MAIL FROM: <'+ mailFromAddress + '>\r\n'

clientSocketSSL.send(MFCommand.encode())

recv = clientSocketSSL.recv(1024).decode()

print(recv)

if recv[:3] != '250':

    print('250 reply not received from server.')

# Send RCPT TO command and print server response.

RTCommand = 'RCPT TO: <'+ mailToAddress + '>\r\n'

clientSocketSSL.send(RTCommand.encode())

recv = clientSocketSSL.recv(1024).decode()

print(recv)

if recv[:3] != '250':

    print('250 reply not received from server.')

# Send DATA command and print server response.

DATACommand = 'DATA\r\n'

clientSocketSSL.send(DATACommand.encode())

recv = clientSocketSSL.recv(1024).decode()

print(recv)

if recv[:3] != '354':

    print('354 reply not received from server.')

# Send message data.

msg = 'From: ' + mailFromAddress + '\r\n'

msg += 'To: ' + mailToAddress +  '\r\n'

msg += 'Subject: ' + '测试' +  '\r\n'

msg += "\r\n I love computer networks!"

clientSocketSSL.send(msg.encode())

# Message ends with a single period.

endmsg = "\r\n.\r\n"

clientSocketSSL.send(endmsg.encode())

recv = clientSocketSSL.recv(1024).decode()

print(recv)

if recv[:3] != '250':

    print('250 reply not received from server.')

# Send QUIT command and get server response.

QUITCommand = 'QUIT\r\n'

clientSocketSSL.send(QUITCommand.encode())

recv = clientSocketSSL.recv(1024).decode()

print(recv)

if recv[:3] != '221':

    print('221 reply not received from server.')

clientSocketSSL.close()

**练习2完整代码**

from socket import \*

import base64

import ssl

# Choose a mail server (e.g. Google mail server) and call it mailserver

mailserver = "smtp.qq.com"

mailFromAddress = 'tianzhang.chn@qq.com'

mailToAddress = 'tianzhang.chn@gmail.com'

username = mailFromAddress

password = 'ybtlpzfewslqdcdi'

# Create socket called clientSocket and establish a TCP connection with mailserver

serverPort=465

context = ssl.create\_default\_context()

clientSocket = socket(AF\_INET, SOCK\_STREAM)

clientSocket.connect((mailserver, serverPort))

clientSocketSSL = context.wrap\_socket(clientSocket, server\_hostname=mailserver)

recv = clientSocketSSL.recv(1024).decode()

print(recv)

if recv[:3] != '220':

    print('220 reply not received from server.')

# Send HELO command and print server response.

heloCommand = 'HELO mailserver\r\n'

clientSocketSSL.send(heloCommand.encode())

recv = clientSocketSSL.recv(1024).decode()

print(recv)

if recv[:3] != '250':

    print('250 reply not received from server.')

# 登录过程

loginCommand = 'AUTH LOGIN\r\n'

clientSocketSSL.send(loginCommand.encode())

recv = clientSocketSSL.recv(1024).decode()

print(recv)

if recv[:3] != '334':

    print('334 reply not received from server.')

# 邮箱账户经过base64编码

userCommand = base64.b64encode(username.encode()) + b'\r\n'

clientSocketSSL.send(userCommand)

recv = clientSocketSSL.recv(1024).decode()

print(recv)

if recv[:3] != '334':

    print('334 reply not received from server')

# 邮箱密码经过base64编码

passCommand = base64.b64encode(password.encode()) + b'\r\n'

clientSocketSSL.send(passCommand)

recv = clientSocketSSL.recv(1024).decode()

print(recv)

if recv[:3] != '235':

    print('235 reply not received from server')

# Send MAIL FROM command and print server response.

MFCommand = 'MAIL FROM: <'+ mailFromAddress + '>\r\n'

clientSocketSSL.send(MFCommand.encode())

recv = clientSocketSSL.recv(1024).decode()

print(recv)

if recv[:3] != '250':

    print('250 reply not received from server.')

# Send RCPT TO command and print server response.

RTCommand = 'RCPT TO: <'+ mailToAddress + '>\r\n'

clientSocketSSL.send(RTCommand.encode())

recv = clientSocketSSL.recv(1024).decode()

print(recv)

if recv[:3] != '250':

    print('250 reply not received from server.')

# Send DATA command and print server response.

DATACommand = 'DATA\r\n'

clientSocketSSL.send(DATACommand.encode())

recv = clientSocketSSL.recv(1024).decode()

print(recv)

if recv[:3] != '354':

    print('354 reply not received from server.')

# Send message data.

# 转换图片和HTML文本

with open("刘亦菲.jpg","rb") as f:

    image\_data = base64.b64encode(f.read())

with open("hello.html","rb") as f:

    html\_data = base64.b64encode(f.read())

# 构造邮件正文 参考文档 http://niehan.blog.techweb.com.cn/archives/250.html

msg = 'FROM: ' + mailFromAddress + '\r\n'

msg += 'TO: ' + mailToAddress +  '\r\n'

msg += 'Subject:超文本和图片\r\n'

*'''*

*+----------------------------------------multipart/mixed----------------------------------------+*

*|                                                                                               |*

*|                                                                                               |*

*|   +---------------------------multipart/related---------------------------+                   |*

*|   |                                                                       |                   |*

*|   |                                                                       |                   |*

*|   |   +---------mutipart/alternative--------------+   +-------------+     |   +-------+       |*

*|   |   |                                           |   |   内嵌资源   |     |   |  附件  |       |*

*|   |   |                                           |   +-------------+     |   +-------+       |*

*|   |   |   +---------------+   +---------------+   |                       |                   |*

*|   |   |   |   纯文本正文   |   |   超文本正文    |   |                       |                   |*

*|   |   |   +---------------+   +---------------+   |   +-------------+     |   +-------+       |*

*|   |   |                                           |   |   内嵌资源   |     |   |  附件  |       |*

*|   |   |                                           |   +-------------+     |   +-------+       |*

*|   |   +-------------------------------------------+                       |                   |*

*|   |                                                                       |                   |*

*|   |                                                                       |                   |*

*|   +-----------------------------------------------------------------------+                   |*

*|                                                                                               |*

*|                                                                                               |*

*+-----------------------------------------------------------------------------------------------+*

*'''*

# 这里我们使用了超文本和图片，因此需要设置 Content-Type:multipart/related

# boundary参数取值 https://www.vlts.cn/posts/28d1c9d8.html

*'''*

*请求头的Content-Type属性除了指定为multipart/related, 还需要定义boundary属性。*

*boundary的值必须以英文中间双横杠--开头，这个--称为前导连字符;*

*boundary的值除了前导连字符以外的部分不能超过70个字符;*

*boundary的值不能包含HTTP协议或者URL禁用的特殊意义的字符。*

*请求体中的请求行数据是由多部分组成, boundary属性的值模式--${boundary}用于分隔每个独立的分部。*

*每个部分可以单独定义Content-Type和该部分的数据体*

*请求体以boundary参数的值模式--${boundary}--作为结束标志*

*每个--${boundary}之前默认强制必须为CRLF,*

*如果某一个部分的文本类型请求体以CRLF结尾,那么在请求体的二级制格式上,必须显式存在两个CRLF,*

*如果某一个部分的请求体不以CRLF结尾,可以只存在一个CRLF,这两种情况分别称为分隔符的显式类型和隐式类型。*

*'''*

msg += 'Content-Type:multipart/related; boundary="--tianzhang"\r\n'

msg += 'MIME-Version:1.0\r\n'

msg += '\r\n'

msg = msg.encode()

# html文件与图片此时已经是二进制字符串，因此我们每次添加的数据都需要进行编码发送，即不能直接添加后整体编码

# 添加html文件

msg += '----tianzhang\r\n'.encode()

# content-type的各种取值 https://www.runoob.com/http/http-content-type.html

msg += 'Content-Type:text/html; charset="UTF-8"\r\n'.encode()

*'''*

*MIME邮件可以传送图像、声音、视频以及附件,这些非ASCII码的数据都是通过一定的编码规则进行转换后附着在邮件中进行传递的。*

*编码方式存储在邮件的Content-Transfer-Encoding域中,*

*一封邮件中可能有多个Content-Transfer-Encoding域,分别对应邮件不同部分内容的编码方式。*

*目前MIME邮件中的数据编码普遍采用Base64编码或Quoted-printable编码来实现。*

*'''*

# 对二进制内容进行转换  name属性值有:7bit、8bit、binary、quoted-printable、base64

msg += 'Content-Transfer-Encoding:base64\r\n'.encode()

msg += '\r\n'.encode()

msg += html\_data

msg += '\r\n'.encode()

msg += '\r\n'.encode()

# 添加图片

msg += '----tianzhang\r\n'.encode()

msg += 'Content-Type:image/jpeg; name="刘亦菲.jpg"\r\n'.encode()

msg += 'Content-Transfer-Encoding:base64\r\n'.encode()

msg += '\r\n'.encode()

msg += image\_data

msg += "\r\n".encode()

msg += '\r\n'.encode()

msg += '----tianzhang--\r\n'.encode()

clientSocketSSL.send(msg)

# Message ends with a single period.

endmsg = "\r\n.\r\n"

clientSocketSSL.send(endmsg.encode())

recv = clientSocketSSL.recv(1024).decode()

print(recv)

if recv[:3] != '250':

    print('250 reply not received from server.')

# Send QUIT command and get server response.

QUITCommand = 'QUIT\r\n'

clientSocketSSL.send(QUITCommand.encode())

recv = clientSocketSSL.recv(1024).decode()

print(recv)

if recv[:3] != '221':

    print('221 reply not received from server.')

clientSocketSSL.close()