

# 实验二：python实现Email客户端

|  |  |
| --- | --- |
| **学号：** | 20231199 |
| **姓名：** | **尚文韬** |
| **专业：** | **保密技术** |
| **学院：** | **计算机与信息技术学院** |

目录

**[一、 准备实验](#_Toc1102559474_WPSOffice_Level1)** **[3](#_Toc1102559474_WPSOffice_Level1)**

[1.实验目的](#_Toc139674031_WPSOffice_Level2) [3](#_Toc139674031_WPSOffice_Level2)

[2. 实验环境](#_Toc301812846_WPSOffice_Level2) [3](#_Toc301812846_WPSOffice_Level2)

[3.实验内容](#_Toc212128508_WPSOffice_Level2) [4](#_Toc212128508_WPSOffice_Level2)

**[二、 实验操作](#_Toc139674031_WPSOffice_Level1)** **[4](#_Toc139674031_WPSOffice_Level1)**

[1. 实验过程](#_Toc420979936_WPSOffice_Level2) [4](#_Toc420979936_WPSOffice_Level2)

[2. 代码实现](#_Toc1598651134_WPSOffice_Level2) [4](#_Toc1598651134_WPSOffice_Level2)

[2. 实验结果展示](#_Toc1361701521_WPSOffice_Level2) [9](#_Toc1361701521_WPSOffice_Level2)

**[三、 实验总结](#_Toc301812846_WPSOffice_Level1)** **[10](#_Toc301812846_WPSOffice_Level1)**

[1.问题总结](#_Toc384237368_WPSOffice_Level2) [10](#_Toc384237368_WPSOffice_Level2)

[2. 改进](#_Toc394117447_WPSOffice_Level2) [11](#_Toc394117447_WPSOffice_Level2)

# 准备实验

## 1.实验目的

（1）运用各种编程语言实现基于smtp以及imap的Email 客户端软件（包括接收邮件和发送邮件）

本次实验选择使用python语言，是基于课本第二章的课后套接字编程作业，详情参考《计算机网络自顶向下方法》（第七版）第二章应用层P119套接字编程作业3

## 实验环境

1. 操作系统：macOS Montery 12.1
2. 使用软件：Visual Studio Code
3. 编程语言：python 3.9.1
4. 硬件环境：MacBook Pro M1芯片

注意：现在提供的工具存在版本与环境的问题，需要选择合适系统完成

## 3.实验内容

（1）实现smtp协议完成邮件发送功能

（2）实现可以完成收邮件功能的协议，完成邮件收发功能

（3）测试收发功能

式，设计了更便于分析的报文返回格式。

# 实验操作

## 实验过程

(主要是记录自己完成的过程情况和问题，包括实验平台的问题；给出代码运行结果)

1. 调用python内置库smtplib以及imaplib，用MIMEText来传送html
2. 搜索资料，通过了解163.com的网络，需要输入**授权码**，需要通过163.com认证（根据[RFC 2971协议](https://datatracker.ietf.org/doc/html/rfc2971)需要客户端首先上传身份信息），使用hello函数构造合法的身份信息上传后解决问题。
3. 分别使用smtp的功能实现，发送邮件及报文；使用imap的功能实现，邮箱客户端（可以接收文件，详细参考实验结果）
4. 运行client.py，实现Email客户端接收发邮件功能



图一二：为设置163.com设置SMTP的截图（来源于网络）

## 代码实现

client.py源代码：



imap\_client.py源代码：

(隐私信息此处用红色标记，具体操作请输入**邮箱地址、个人授权码、imap邮箱服务器ip**)

import email

import email.header

import imaplib

ROOT\_DIR = "resourses/"

class imap\_client:

def \_\_init\_\_(self):

#input your own hostname

self.user\_id = ['hostname'](mailto:'donaldshawn1199@163.com')

#input your own authorization code

self.password = 'authorization\_code'

#input your imap server

self.imap\_server = 'imap.example.com'

def login(self):

try:

serv = imaplib.IMAP4(self.imap\_server, 143)

print('imap----服务器连接成功')

except Exception as e:

print('imap----服务器连接失败:', e)

return

try:

serv.login(self.user\_id, self.password)

print('imap----登录成功')

return serv

except Exception as e:

print('imap----登录失败：', e)

return

def logout(self, conn):

conn.close

conn.logout()

def get\_content(self, conn):

conn.select()

op = input("默认显示未读邮件，输入A读取所有邮件")

if op == 'A':

ret, data = conn.search(None, 'ALL')

else:

ret, data = conn.search(None, 'UNSEEN')

# 邮件列表

email\_list = data[0].split()

emailnum = len(email\_list)

if (emailnum == 0):

print('收件箱为空，已退出')

return

print("查询到", emailnum, "封邮件")

while True:

try:

num = int(input("请输入需要获取的邮件序号"))

break

except ValueError:

print("请输入一个合法数字")

item = email\_list[len(email\_list) - num]

ret, data = conn.fetch(item, '(RFC822)')

msg = email.message\_from\_string(data[0][1].decode())

sub = msg.get('subject')

email\_from = msg.get('from')

email\_to = msg.get('to')

sub\_text = email.header.decode\_header(sub)

email\_from\_text = email.header.decode\_header(email\_from)

email\_to\_text = email.header.decode\_header(email\_to)

# 如果是特殊字符，元组的第二位会给出编码格式，需要转码

if sub\_text[0]:

sub\_detail = self.\_\_tuple2str(sub\_text[0])

email\_from\_detail = ''

for i in range(len(email\_from\_text)):

email\_from\_detail = email\_from\_detail + \

self.\_\_tuple2str(email\_from\_text[i])

email\_to\_detail = ''

for i in range(len(email\_to\_text)):

email\_to\_detail = email\_to\_detail + \

self.\_\_tuple2str(email\_to\_text[i])

print('主题：', sub\_detail)

print('发件人：', email\_from\_detail)

print('收件人：', email\_to\_detail)

# 通过walk可以遍历出所有的内容

for part in msg.walk():

# 这里要判断是否是multipart，如果是，数据没用丢弃

if not part.is\_multipart():

content\_type = part.get\_content\_type()

name = part.get\_filename()

if name:

trans\_name = email.header.decode\_header(name)

if trans\_name[0][1]:

file\_name = trans\_name[0][0].decode(trans\_name[0][1])

else:

file\_name = trans\_name[0][0]

print('开始下载附件:', file\_name)

attach\_data = part.get\_payload(

decode=True) # 解码存储数据

try:

f = open(ROOT\_DIR + file\_name, 'wb')

except Exception as e:

print(e)

f = open(ROOT\_DIR + 'tmp', 'wb')

# 遇到错误缓存文件

f.write(attach\_data)

f.close()

print('附件下载成功:', file\_name)

else:

# 文本内容

txt = part.get\_payload(decode=True)

if content\_type == 'text/html':

print("文件格式为html，请使用浏览器查看文件")

htmlfile = open("resourses/html\_content.html", 'w')

try:

txt = txt.decode()

except UnicodeDecodeError:

txt = txt.decode('gbk')

htmlfile.write(txt)

htmlfile.close()

elif content\_type == 'text/plain':

print('邮件正文：')

# 尝试解码

try:

txt = txt.decode()

except UnicodeDecodeError:

txt = txt.decode('gbk')

print(txt)

def hello(self, conn):

imaplib.Commands['ID'] = 'AUTH'

args = ("name", "lisi", "contact", "null",

"version", "0.0", "vendor", "myclient")

typ, dat = conn.\_simple\_command('ID', '("' + '" "'.join(args) + '")')

def \_\_tuple2str(self, tuple\_):

if tuple\_[1]:

out\_str = tuple\_[0].decode(tuple\_[1])

else:

if isinstance(tuple\_[0], bytes):

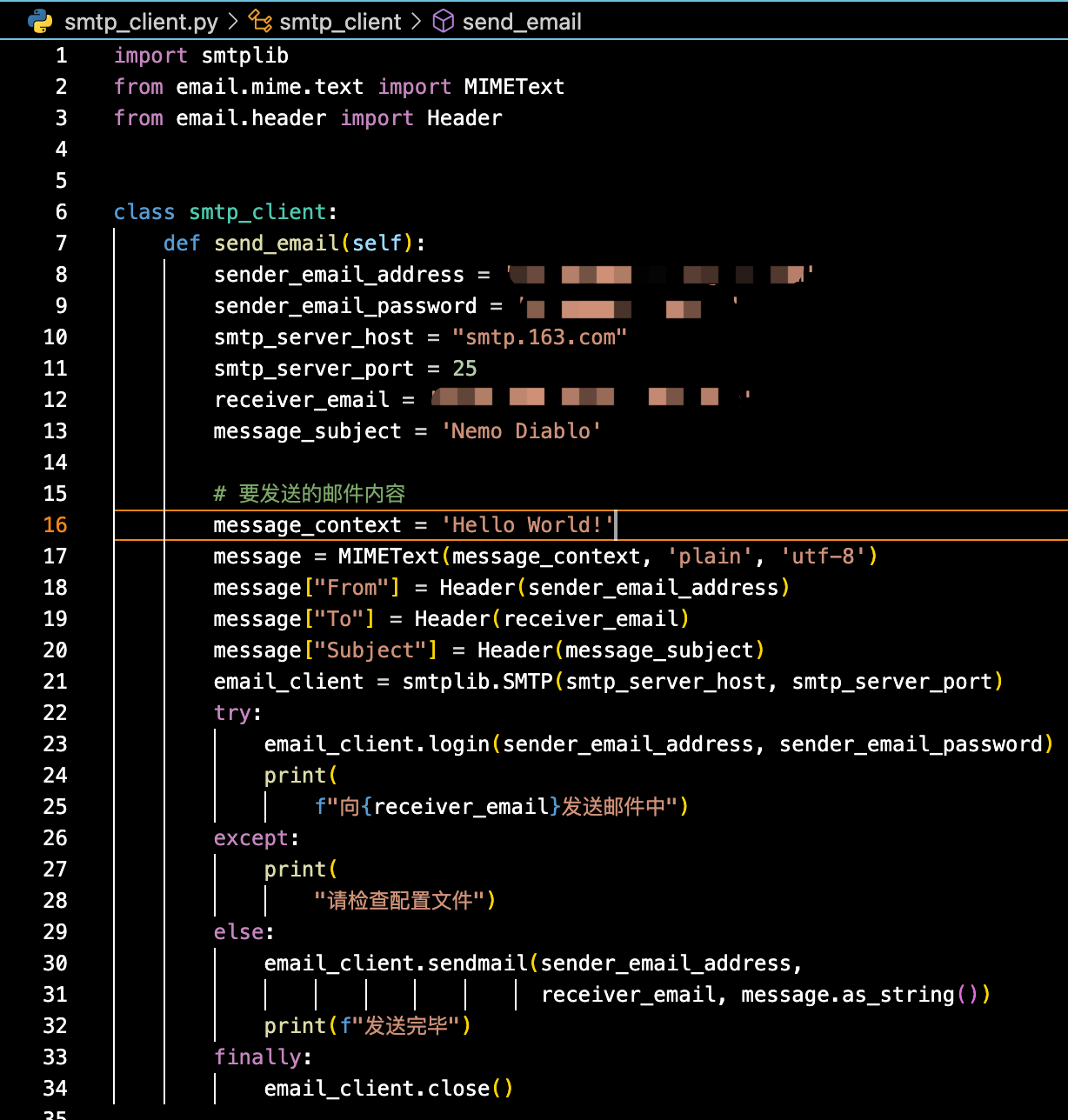
out\_str = tuple\_[0].decode('gbk')

else:

out\_str = tuple\_[0]

return out\_str

smtp\_client.py源代码：((隐私信息此处用马赛克屏蔽，具体操作请输入**邮箱地址、个人授权码、imap邮箱服务器ip**))



## 实验结果展示

（1）运行client.py

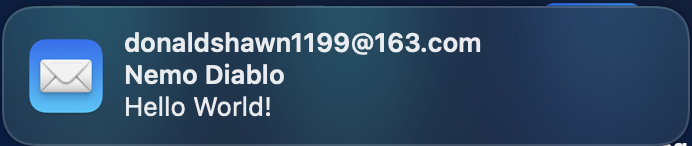
## 

图三：客户端连接成功

（2）实现email发送功能，通过smtp协议：即：在终端输入S

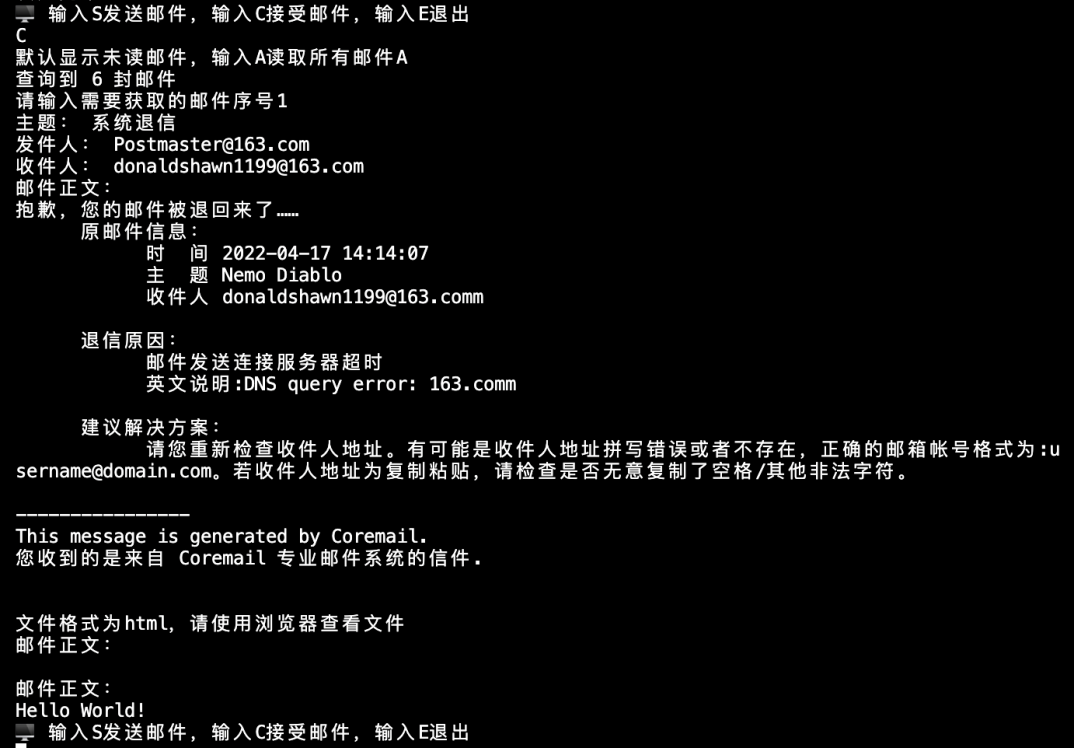


图四：发送邮件成功



图五：接收邮件成功

1. 实现email接收功能，通过imap协议：即：在终端输入C



图四：查看邮件成功

# 实验总结

## 1.问题总结

（1）这次实验耗时最多的是阅读库的文档和模板代码，smtplib以及imaplib都是python内置库，api的设置很规范，我在实现的时候常常遇到编码的问题，出现异常，判断一个字段的编码让我卡住很久，后来我仔细地阅读了文档之后才发现字段的编码都在(内容，编码)二元组中给出了，只要使用decode(编码)的形式便可以解决问题。

（2）是对于这次的实验环境不同根据所选的邮箱服务器不同也有不同。我选择的是163邮箱，当login之后准备执行search()函数的时候突然报错，错误如下：

imaplib.error: command SEARCH illegal in state AUTH, only allowed in states SELECTED

经过网上查询，发现这是由于网易邮箱根据[RFC 2971协议](https://datatracker.ietf.org/doc/html/rfc2971)需要客户端首先上传身份信息。这里使用hello函数构造合法的身份信息上传后解决问题。

## 改进

（1）这次的邮件收取的界面使用的是命令行界面，可以使用GUI来提高软件的易用性。Python的运行速度相对于java等语言较为缓慢，如果对速度有需求可以使用java实现对应逻辑。

（2）可以设计一个功能时时监听收件箱中邮件数量，缓存发来的邮件并提醒用户

（3）本次实验在利用smtp协议与imap协议完成了邮件收发的基本功能的基础上实现了附件接收与html格式邮件的收取。完成了实验要求的任务