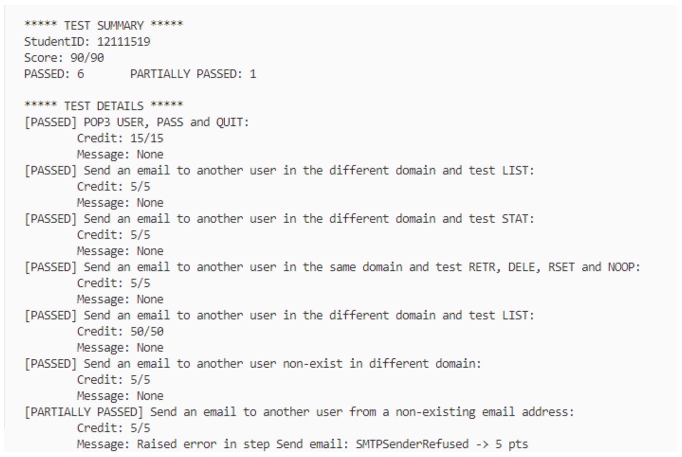
Assignment 1

Basic Part

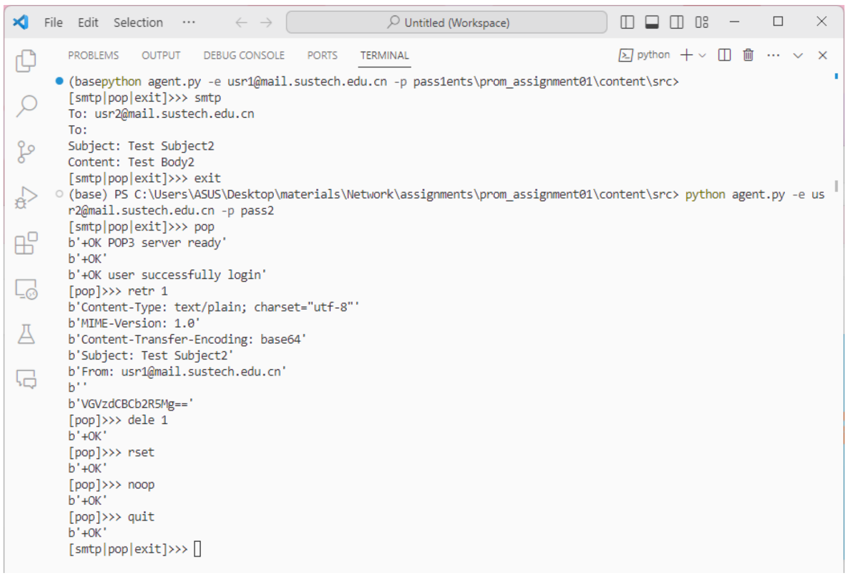
Passed the test in WSL:



```
***** TEST SUMMARY *****
StudentID: 12111519
Score: 90/90
PASSED: 6       PARTIALLY PASSED: 1

***** TEST DETAILS *****
[PASSED] POP3 USER, PASS and QUIT:
        Credit: 15/15
        Message: None
[PASSED] Send an email to another user in the different domain and test LIST:
        Credit: 5/5
        Message: None
[PASSED] Send an email to another user in the different domain and test STAT:
        Credit: 5/5
        Message: None
[PASSED] Send an email to another user in the same domain and test RETR, DELE, RSET and NOOP:
        Credit: 5/5
        Message: None
[PASSED] Send an email to another user in the different domain and test LIST:
        Credit: 50/50
        Message: None
[PASSED] Send an email to another user non-exist in different domain:
        Credit: 5/5
        Message: None
[PARTIALLY PASSED] Send an email to another user from a non-existing email address:
        Credit: 5/5
        Message: Raised error in step Send email: SMTPSenderRefused -> 5 pts
```

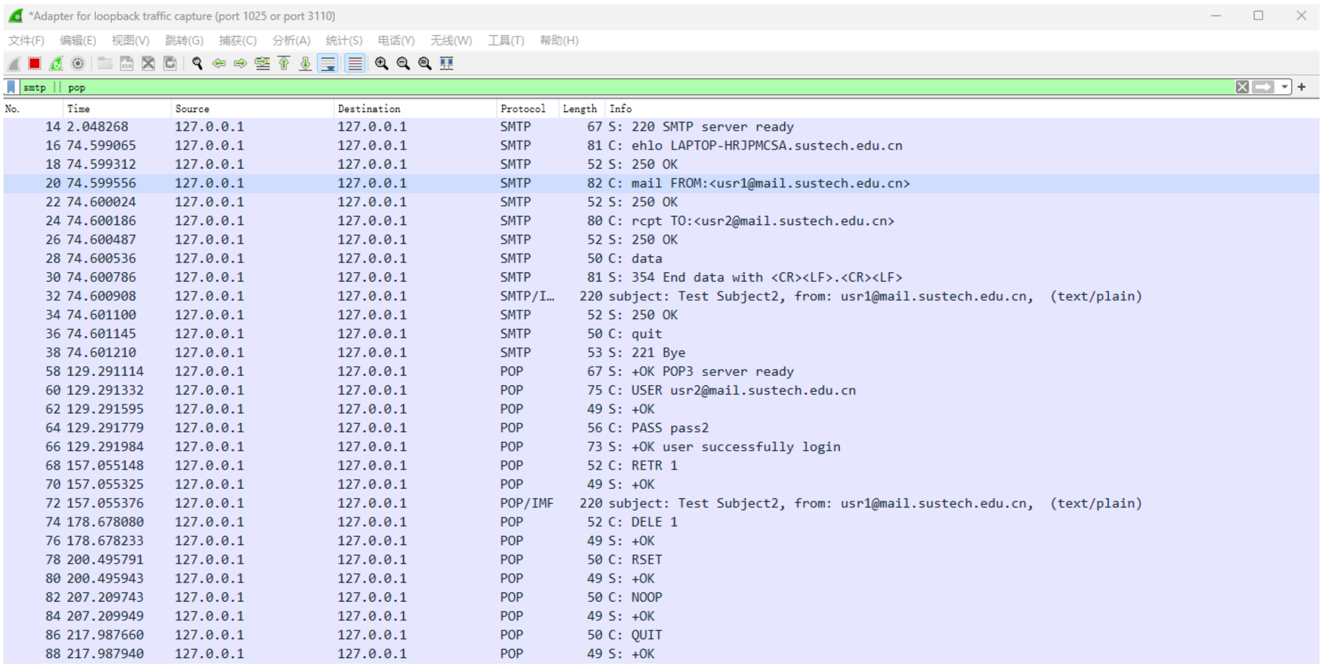
For packet capturing, I run the steps in 4.yml manually in Windows.

In agent.py:



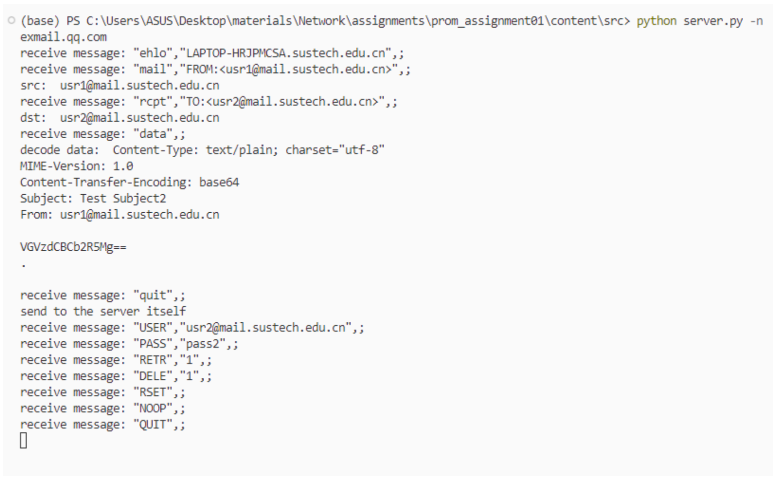
```
(basepython agent.py -e usr1@mail.sustech.edu.cn -p pass1ents\prom_assignment01\content\src>
[smtp|pop|exit]>>> smtp
To: usr2@mail.sustech.edu.cn
To:
Subject: Test Subject2
Content: Test Body2
[smtp|pop|exit]>>> exit
(base) PS C:\Users\ASUS\Desktop\materials\Network\assignments\prom_assignment01\content\src> python agent.py -e us
r2@mail.sustech.edu.cn -p pass2
[smtp|pop|exit]>>> pop
b'+OK POP3 server ready'
b'+OK'
b'+OK user successfully login'
[pop]>>> retr 1
b'Content-Type: text/plain; charset="utf-8"'
b'MIME-Version: 1.0'
b'Content-Transfer-Encoding: base64'
b'Subject: Test Subject2'
b'From: usr1@mail.sustech.edu.cn'
b''
b'VGVzdCBCb2R5Mg=='
[pop]>>> dele 1
b'+OK'
[pop]>>> rset
b'+OK'
[pop]>>> noop
b'+OK'
[pop]>>> quit
b'+OK'
[smtp|pop|exit]>>> 
```

In Wireshark:


*Adapter for loopback traffic capture (port 1025 or port 3110)
文件(F) 编辑(E) 视图(V) 跳转(G) 捕获(C) 分析(A) 统计(S) 电话(Y) 无线(W) 工具(T) 帮助(H)
smtp || pop

```
No.  Time        Source      Destination  Protocol  Length  Info
  14 2.048268    127.0.0.1   127.0.0.1    SMTP        67 S: 220 SMTP server ready
  16 74.599065   127.0.0.1   127.0.0.1    SMTP        81 C: ehlo LAPTOP-HRJPMCSA.sustech.edu.cn
  18 74.599312   127.0.0.1   127.0.0.1    SMTP        52 S: 250 OK
  20 74.599556   127.0.0.1   127.0.0.1    SMTP        82 C: mail FROM:<usr1@mail.sustech.edu.cn>
  22 74.600024   127.0.0.1   127.0.0.1    SMTP        52 S: 250 OK
  24 74.600186   127.0.0.1   127.0.0.1    SMTP        80 C: rcpt TO:<usr2@mail.sustech.edu.cn>
  26 74.600487   127.0.0.1   127.0.0.1    SMTP        52 S: 250 OK
  28 74.600536   127.0.0.1   127.0.0.1    SMTP        50 C: data
  30 74.600786   127.0.0.1   127.0.0.1    SMTP        81 S: 354 End data with <CR><LF>.<CR><LF>
  32 74.600908   127.0.0.1   127.0.0.1    SMTP/I…    220 subject: Test Subject2, from: usr1@mail.sustech.edu.cn,  (text/plain)
  34 74.601100   127.0.0.1   127.0.0.1    SMTP        52 S: 250 OK
  36 74.601145   127.0.0.1   127.0.0.1    SMTP        50 C: quit
  38 74.601210   127.0.0.1   127.0.0.1    SMTP        53 S: 221 Bye
  58 129.291114  127.0.0.1   127.0.0.1    POP         67 S: +OK POP3 server ready
  60 129.291332  127.0.0.1   127.0.0.1    POP         75 C: USER usr2@mail.sustech.edu.cn
  62 129.291595  127.0.0.1   127.0.0.1    POP         49 S: +OK
  64 129.291779  127.0.0.1   127.0.0.1    POP         56 C: PASS pass2
  66 129.291984  127.0.0.1   127.0.0.1    POP         73 S: +OK user successfully login
  68 157.055148  127.0.0.1   127.0.0.1    POP         52 C: RETR 1
  70 157.055325  127.0.0.1   127.0.0.1    POP         49 S: +OK
  72 157.055376  127.0.0.1   127.0.0.1    POP/IMF    220 subject: Test Subject2, from: usr1@mail.sustech.edu.cn,  (text/plain)
  74 178.678080  127.0.0.1   127.0.0.1    POP         52 C: DELE 1
  76 178.678233  127.0.0.1   127.0.0.1    POP         49 S: +OK
  78 200.495791  127.0.0.1   127.0.0.1    POP         50 C: RSET
  80 200.495943  127.0.0.1   127.0.0.1    POP         49 S: +OK
  82 207.209743  127.0.0.1   127.0.0.1    POP         50 C: NOOP
  84 207.209949  127.0.0.1   127.0.0.1    POP         49 S: +OK
  86 217.987660  127.0.0.1   127.0.0.1    POP         50 C: QUIT
  88 217.987940  127.0.0.1   127.0.0.1    POP         49 S: +OK
```

To make things more clear, I also put the screenshot of the output of my server here:



```
(base) PS C:\Users\ASUS\Desktop\materials\Network\assignments\prom_assignment01\content\src> python server.py -n
exmail.qq.com
receive message: "ehlo","LAPTOP-HRJPMCSA.sustech.edu.cn",;
receive message: "mail","FROM:<usr1@mail.sustech.edu.cn>",;
src:  usr1@mail.sustech.edu.cn
receive message: "rcpt","TO:<usr2@mail.sustech.edu.cn>",;
dst:  usr2@mail.sustech.edu.cn
receive message: "data",;
decode data:  Content-Type: text/plain; charset="utf-8"
MIME-Version: 1.0
Content-Transfer-Encoding: base64
Subject: Test Subject2
From: usr1@mail.sustech.edu.cn

VGVzdCBCb2R5Mg==
.

receive message: "quit",;
send to the server itself
receive message: "USER","usr2@mail.sustech.edu.cn",;
receive message: "PASS","pass2",;
receive message: "RETR","1",;
receive message: "DELE","1",;
receive message: "RSET",;
receive message: "NOOP",;
receive message: "QUIT",;
```

Bonus Part

Error report

In POP, I implement several types of error reporting, including CONN_REFUSED, AUTH_FAILED, INVALID_COMMAND, INVALID_ARGUMENT.

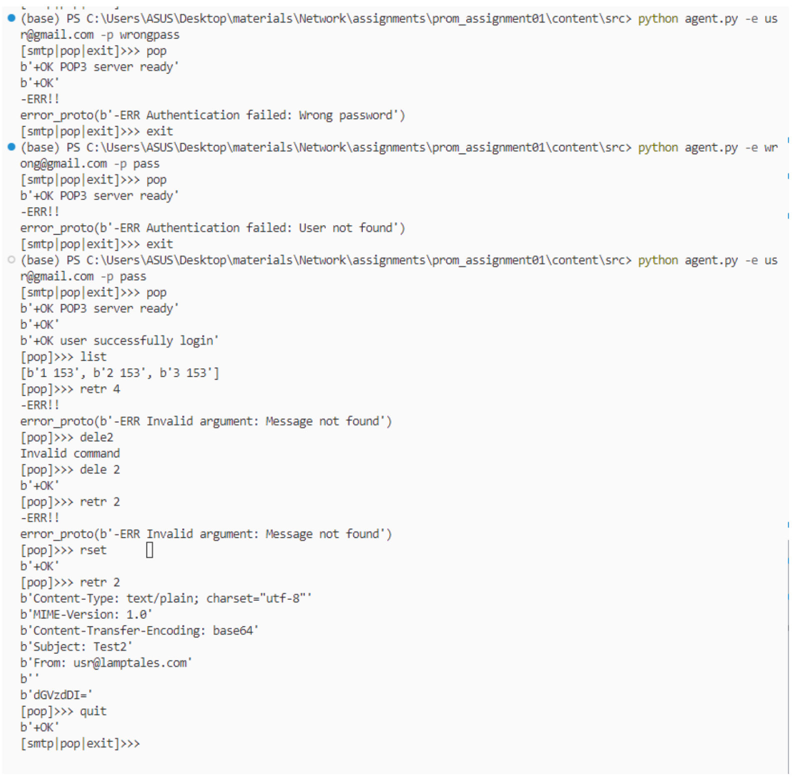
The core part of the code is as follow:

```python
def pop_error_report(error_code, msg=None):
    error_msg = ''
    if error_code == CONN_REFUSED:
        error_msg = '-ERR Connection refused'
    elif error_code == AUTH_FAILED:
        error_msg = '-ERR Authentication failed'
    elif error_code == INVALID_COMMAND:
        error_msg = '-ERR Invalid command'
    elif error_code == INVALID_ARGUMENT:
        error_msg = '-ERR Invalid argument'
    else:
        error_msg = '-ERR Unknown error'
    if msg:
        error_msg += f': {msg}\r\n'
    else:
        error_msg += '\r\n'
    return error_msg.encode()
```

And in the POP server, the error can be reported in this way:

```
if command == 'DELE':
    if len(message) < 2:
        conn.sendall(pop_error_report(INVALID_COMMAND))
        continue
    del_num = int(message[1]) - 1
    if del_num not in left_list:
        conn.sendall(pop_error_report(INVALID_ARGUMENT, 'Message not found'))
        continue
    else:
        delete_list.append(del_num)
        left_list.remove(del_num)
        conn.sendall(b'+OK\r\n')
```

Here is the test for some of the errors:



```
(base) PS C:\Users\ASUS\Desktop\materials\Network\assignments\prom_assignment01\content\src> python agent.py -e us
r@gmail.com -p wrongpass
[smtp|pop|exit]>>> pop
b'+OK POP3 server ready'
b'+OK'
-ERR!!
error_proto(b'-ERR Authentication failed: Wrong password')
[smtp|pop|exit]>>> exit
(base) PS C:\Users\ASUS\Desktop\materials\Network\assignments\prom_assignment01\content\src> python agent.py -e wr
ong@gmail.com -p pass
[smtp|pop|exit]>>> pop
b'+OK POP3 server ready'
-ERR!!
error_proto(b'-ERR Authentication failed: User not found')
[smtp|pop|exit]>>> exit
(base) PS C:\Users\ASUS\Desktop\materials\Network\assignments\prom_assignment01\content\src> python agent.py -e us
r@gmail.com -p pass
[smtp|pop|exit]>>> pop
b'+OK POP3 server ready'
b'+OK'
b'+OK user successfully login'
[pop]>>> list
[b'1 153', b'2 153', b'3 153']
[pop]>>> retr 4
-ERR!!
error_proto(b'-ERR Invalid argument: Message not found')
[pop]>>> dele2
Invalid command
[pop]>>> dele 2
b'+OK'
[pop]>>> retr 2
-ERR!!
error_proto(b'-ERR Invalid argument: Message not found')
[pop]>>> rset
b'+OK'
[pop]>>> retr 2
b'Content-Type: text/plain; charset="utf-8"'
b'MIME-Version: 1.0'
b'Content-Transfer-Encoding: base64'
b'Subject: Test2'
b'From: usr@lamptales.com'
b''
b'dGVzdDI='
[pop]>>> quit
b'+OK'
[smtp|pop|exit]>>>
```

In SMTP, I implement the types of error reporting including the syntax of the command, the existence of the user, and the existence of the receiver's domain server.

Here is some of the important code:

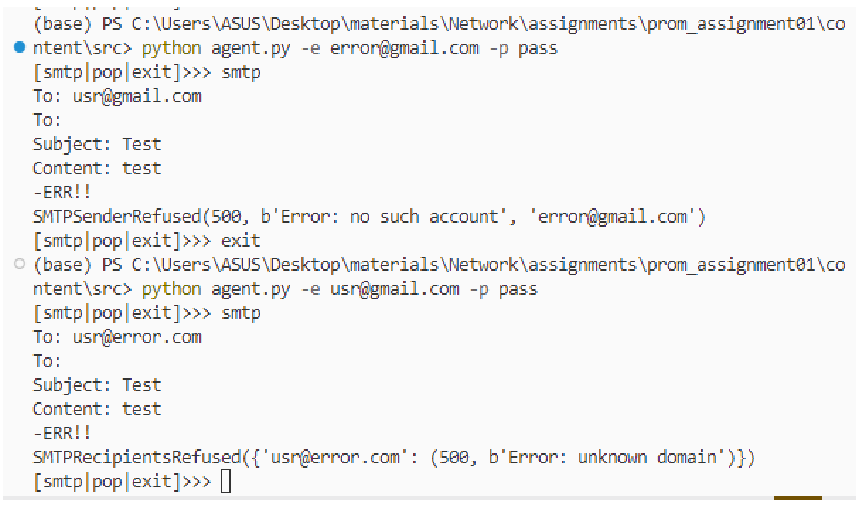
Sender validation:

```
            if state == WAITING_MAIL:
                if len(message) >= 2 and command == 'MAIL' and message[1].startswith('FROM:'):
                    src = message[1][6:len(message[1]) - 1]
                    if DEBUG:
                        print("src: ", src)
                    from_ip, from_port = analyze_addr(src)
                    if from_ip == 'localhost' and from_port == SMTP_PORT and src not in ACCOUNTS:
                        conn.sendall(b'500 Error: no such account\r\n')
                        continue

                    state = WAITING_RCPT
                    conn.sendall(b'250 OK\r\n')
                else:
                    conn.sendall(b'500 Error: you should send a legal MAIL command\r\n')
                    continue
```

Receiver validation:

```
            if state == WAITING_RCPT:
                if len(message) >= 2 and command == 'RCPT' and message[1].startswith('TO:'):
                    dst = message[1][4:len(message[1]) - 1]
                    if DEBUG:
                        print("dst: ", dst)

                    if src not in ACCOUNTS and dst not in ACCOUNTS:
                        conn.sendall(b'500 Error: wrong message\r\n')
                        state = WAITING_MAIL
                        continue
                    temp_domain = dst.split('@')[-1] + '.'
                    if temp_domain not in FDNS['MX']:
                        conn.sendall(b'500 Error: unknown domain\r\n')
                        state = WAITING_MAIL
                        continue

                    state = WAITING_DATA
                    conn.sendall(b'250 OK\r\n')
                else:
                    conn.sendall(b'500 Error: you should send a legal RCPT command\r\n')
                    continue
```

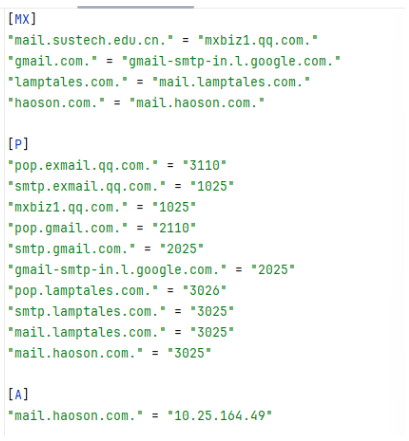
Here is the test for some errors:



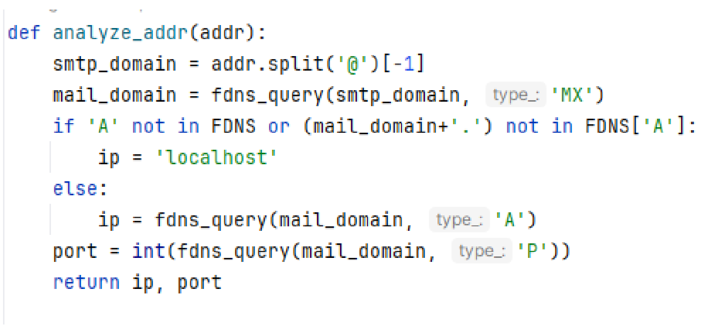
```
(base) PS C:\Users\ASUS\Desktop\materials\Network\assignments\prom_assignment01\co
ntent\src> python agent.py -e error@gmail.com -p pass
[smtp|pop|exit]>>> smtp
To: usr@gmail.com
To:
Subject: Test
Content: test
-ERR!!
SMTPSenderRefused(500, b'Error: no such account', 'error@gmail.com')
[smtp|pop|exit]>>> exit
(base) PS C:\Users\ASUS\Desktop\materials\Network\assignments\prom_assignment01\co
ntent\src> python agent.py -e usr@gmail.com -p pass
[smtp|pop|exit]>>> smtp
To: usr@error.com
To:
Subject: Test
Content: test
-ERR!!
SMTPRecipientsRefused({'usr@error.com': (500, b'Error: unknown domain')})
[smtp|pop|exit]>>> 
```

Peer Mailing

I finish this part with my classmate 秦颢轩(12111321). Assume that the mail server for ***lamptales.com*** is running on my computer, and the mail server for ***haoson.com*** is running on his computer. Now the users under the two servers want to communicate. We adjust the DNS config and the checkup method, so it becomes as follow:



```
[MX]
"mail.sustech.edu.cn." = "mxbiz1.qq.com."
"gmail.com." = "gmail-smtp-in.l.google.com."
"lamptales.com." = "mail.lamptales.com."
"haoson.com." = "mail.haoson.com."

[P]
"pop.exmail.qq.com." = "3110"
"smtp.exmail.qq.com." = "1025"
"mxbiz1.qq.com." = "1025"
"pop.gmail.com." = "2110"
"smtp.gmail.com." = "2025"
"gmail-smtp-in.l.google.com." = "2025"
"pop.lamptales.com." = "3026"
"smtp.lamptales.com." = "3025"
"mail.lamptales.com." = "3025"
"mail.haoson.com." = "3025"

[A]
"mail.haoson.com." = "10.25.164.49"
```

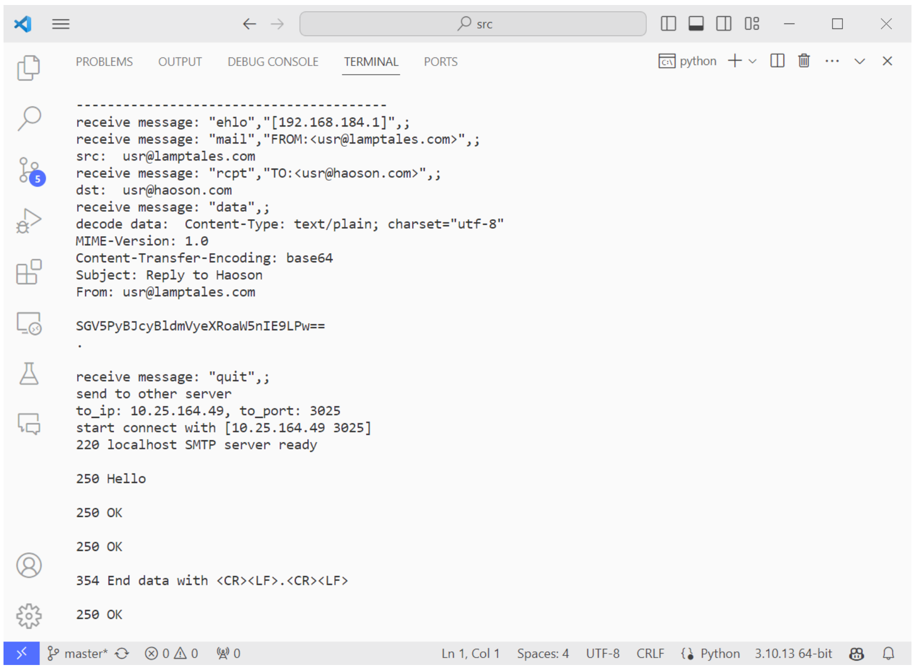


```python
def analyze_addr(addr):
    smtp_domain = addr.split('@')[-1]
    mail_domain = fdns_query(smtp_domain, 'MX')
    if 'A' not in FDNS or (mail_domain+'.') not in FDNS['A']:
        ip = 'localhost'
    else:
        ip = fdns_query(mail_domain, 'A')
    port = int(fdns_query(mail_domain, 'P'))
    return ip, port
```

And now we can send mails to each other, here is an example.

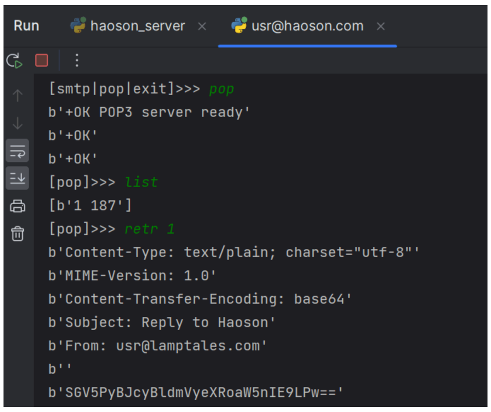
On my computer:

On my lamptales server:



```
----------------------------------------
receive message: "ehlo","[192.168.184.1]",;
receive message: "mail","FROM:<usr@lamptales.com>",;
src:  usr@lamptales.com
receive message: "rcpt","TO:<usr@haoson.com>",;
dst:  usr@haoson.com
receive message: "data",;
decode data:  Content-Type: text/plain; charset="utf-8"
MIME-Version: 1.0
Content-Transfer-Encoding: base64
Subject: Reply to Haoson
From: usr@lamptales.com

SGV5PyBJcyBldmVyeXRoaW5nIE9LPw==
.

receive message: "quit",;
send to other server
to_ip: 10.25.164.49, to_port: 3025
start connect with [10.25.164.49 3025]
220 localhost SMTP server ready

250 Hello

250 OK

250 OK

354 End data with <CR><LF>.<CR><LF>

250 OK
```

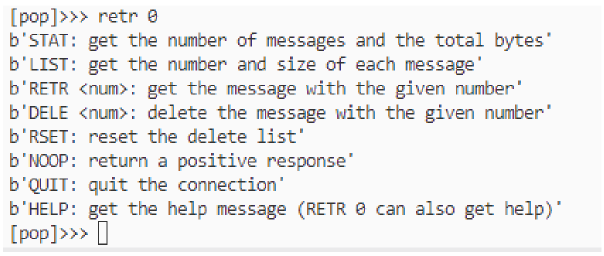
On Qin's computer:



```
[smtp|pop|exit]>>> pop
b'+OK POP3 server ready'
b'+OK'
b'+OK'
[pop]>>> list
[b'1 187']
[pop]>>> retr 1
b'Content-Type: text/plain; charset="utf-8"'
b'MIME-Version: 1.0'
b'Content-Transfer-Encoding: base64'
b'Subject: Reply to Haoson'
b'From: usr@lamptales.com'
b''
b'SGV5PyBJcyBldmVyeXRoaW5nIE9LPw=='
```

Extra Commands:

I implement the HELP command in POP. Send HELP to get the help message. In convenience, as RETR 0 will never be accepted, I will also return the help message to a RETR 0 command.



```
[pop]>>> retr 0
b'STAT: get the number of messages and the total bytes'
b'LIST: get the number and size of each message'
b'RETR <num>: get the message with the given number'
b'DELE <num>: delete the message with the given number'
b'RSET: reset the delete list'
b'NOOP: return a positive response'
b'QUIT: quit the connection'
b'HELP: get the help message (RETR 0 can also get help)'
[pop]>>> 
```