Documentation Contents

Simple Mail Transfer Protocol

Introduction

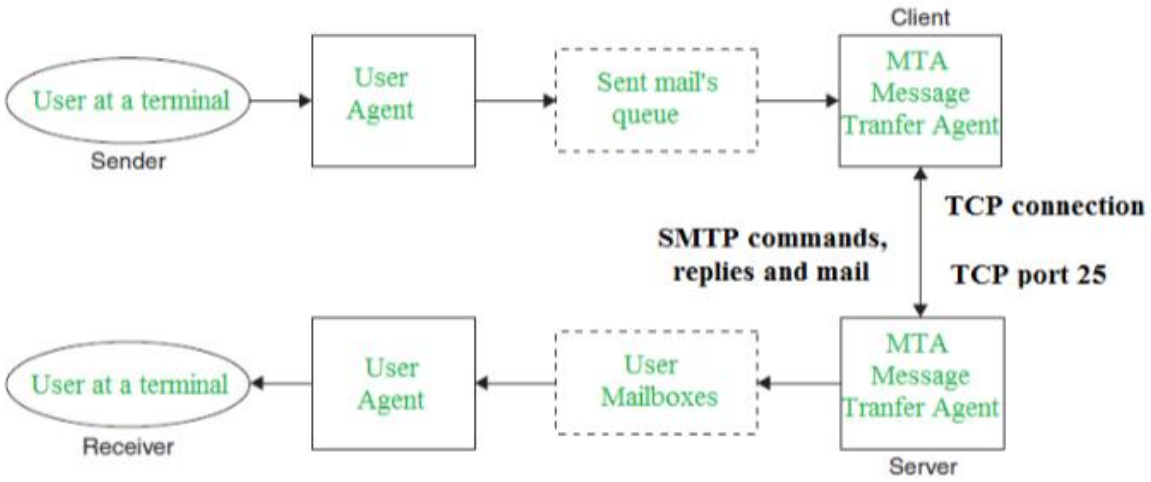
Simple Mail Transfer Protocol (SMTP) is a protocol that allows mail transfer from a user to another over the internet.

Types of SMTP

- *End-to-end*
  Used to communicate between different organizations

- *Store-and-forward method*
  Used to commincate within an organization

SMTP System Model



The Project

This is a simple program made using python to send an email, but without using python's smtplib library.

Github Link: https://github.com/yaraamrsalah/Simple-Mail-Transfer-Protocol-SMTP-.git

The program asks for the sender's email and password, the receiver's email, as well as the email's subject and body (user inputs). It also checks the email connection and prints a confirmation or error message based on whether the connection was successful or not.

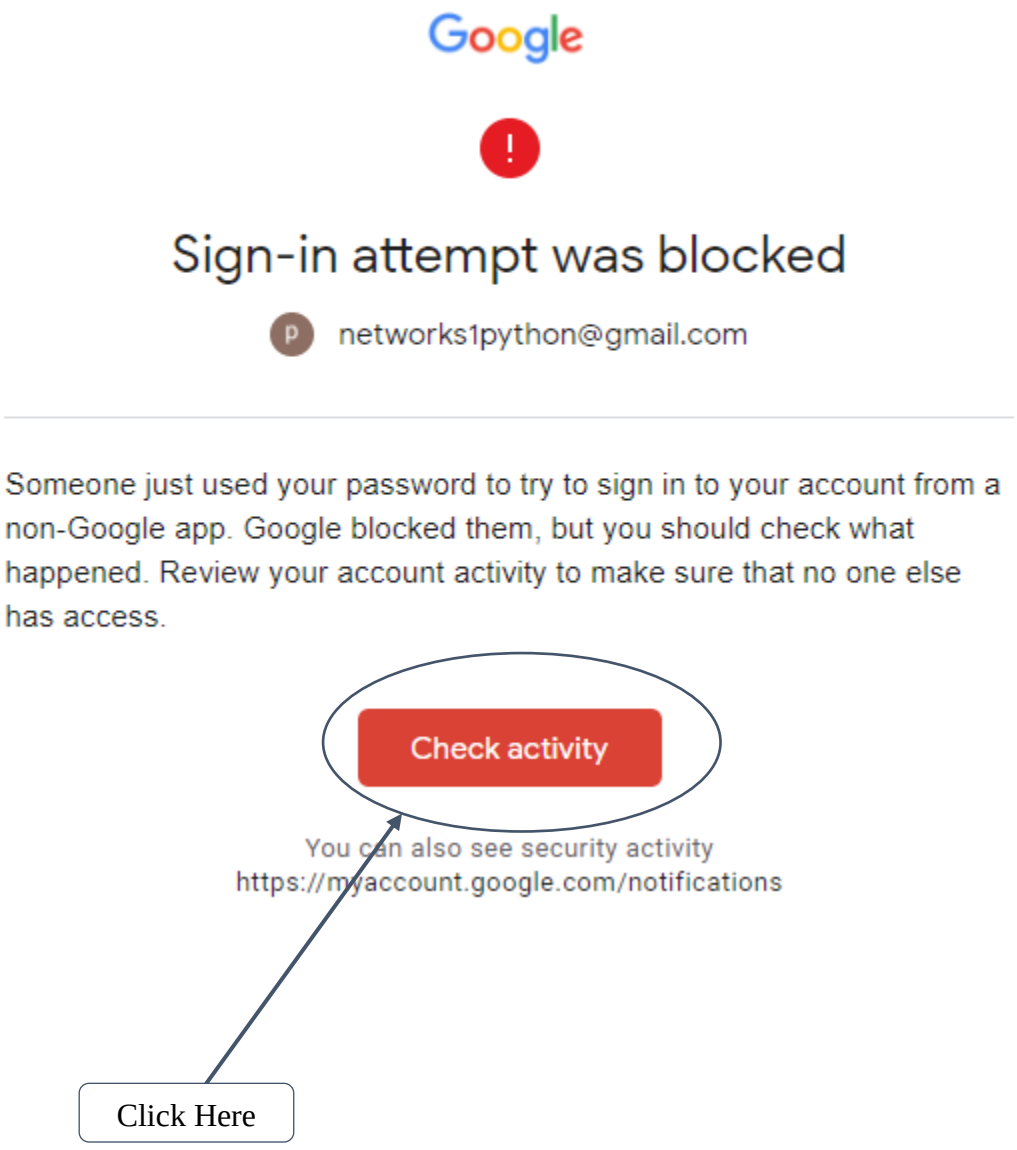
Case I: Unsucessful Connection

```
>>> runfile('C:/Users/yaraa/Desktop/Project/main.py', wdir='C:/Users/yaraa/Desktop/Project')
Enter Your Email Address: >? networks1python@gmail.com
Warning: Password input may be echoed.
Enter Your Password: >? networks1_SMTP
Enter Email Destination: >? networks1python@gmail.com
Enter Email Subject: >? Test
Enter Email Body Message: >? Testing Python
b'220 smtp.gmail.com ESMTP k18sm72457999wrd.45 - gsmtp\r\n'
220 reply not received from server.
250 smtp.gmail.com at your service

b'334 VXNlcm5hbWU6\r\n'
334 UGFzc3dvcmQ6

535-5.7.8 Username and Password not accepted. Learn more at
535 5.7.8  https://support.google.com/mail/?p=BadCredentials k18sm72457999wrd.45 - gsmtp

530-5.7.0 Authentication Required. Learn more at
530 5.7.0  https://support.google.com/mail/?p=WantAuthError k18sm72457999wrd.45 - gsmtp

530-5.7.0 Authentication Required. Learn more at
530 5.7.0  https://support.google.com/mail/?p=WantAuthError k18sm72457999wrd.45 - gsmtp

502 5.5.1 Unrecognized command. k18sm72457999wrd.45 - gsmtp

502 5.5.1 Unrecognized command. k18sm72457999wrd.45 - gsmtp
>>>
```

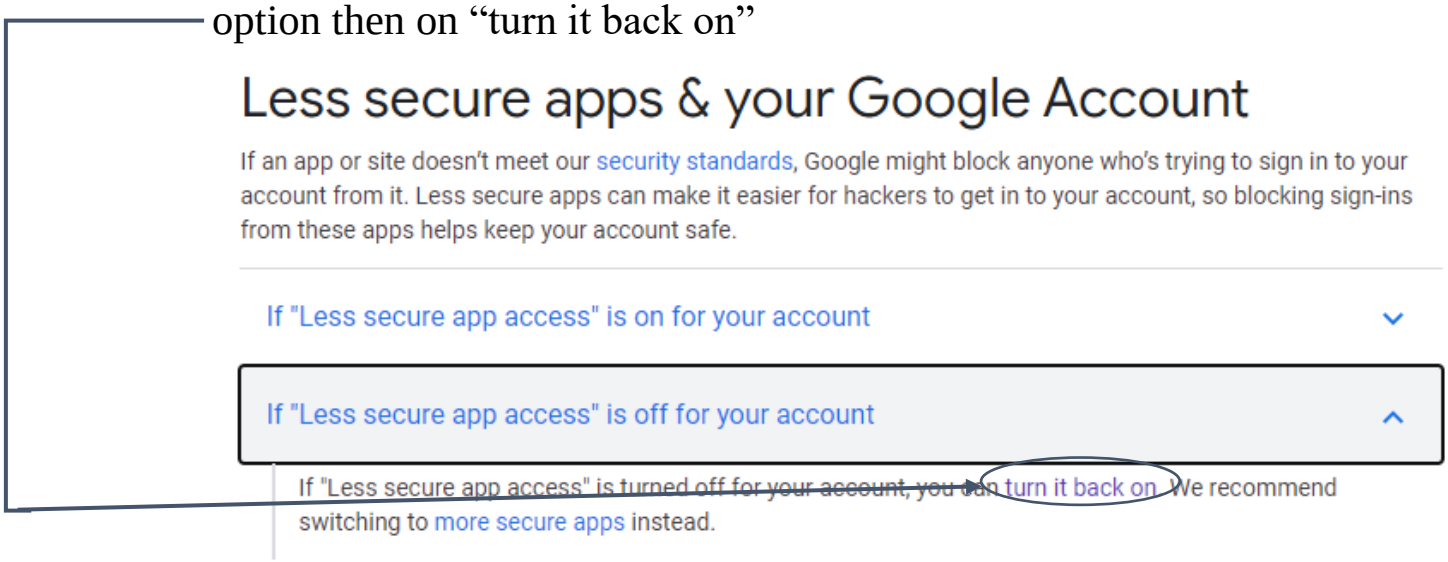
How to Fix Unsuccessful Connection

To fix unsuccessful connection issue, it is important to follow these steps:

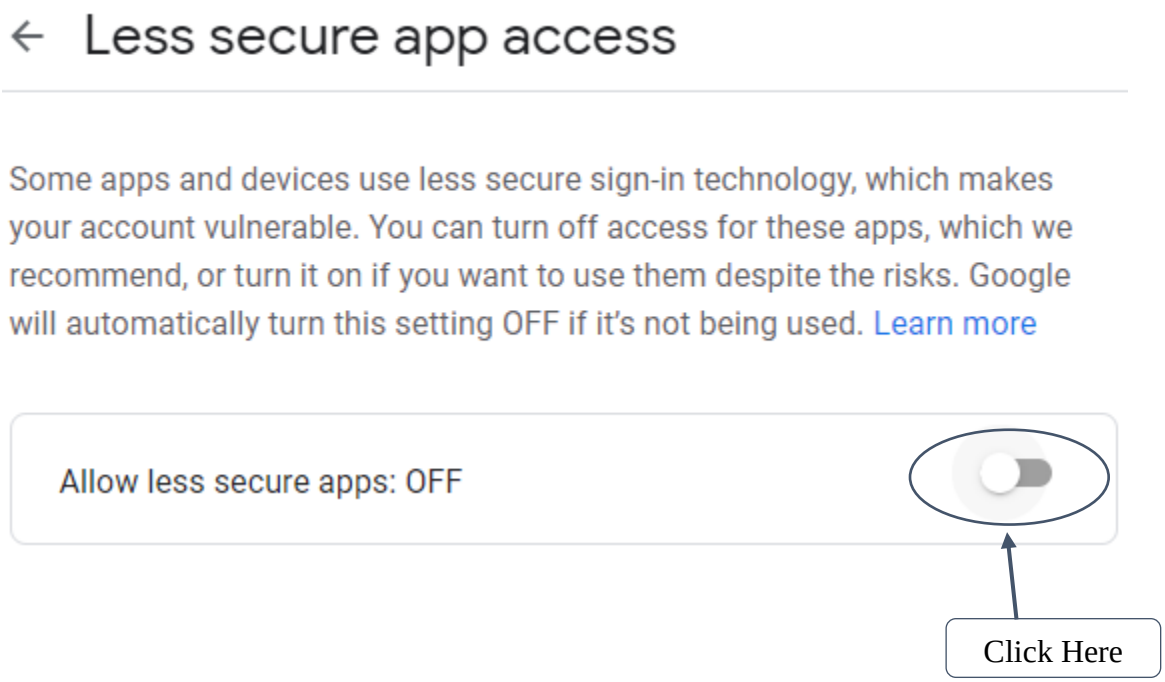
1. After trying to login to the email, a security alert will be emailed to it



2. After clicking, you will be taken to this page. Click on the second option then on “turn it back on”

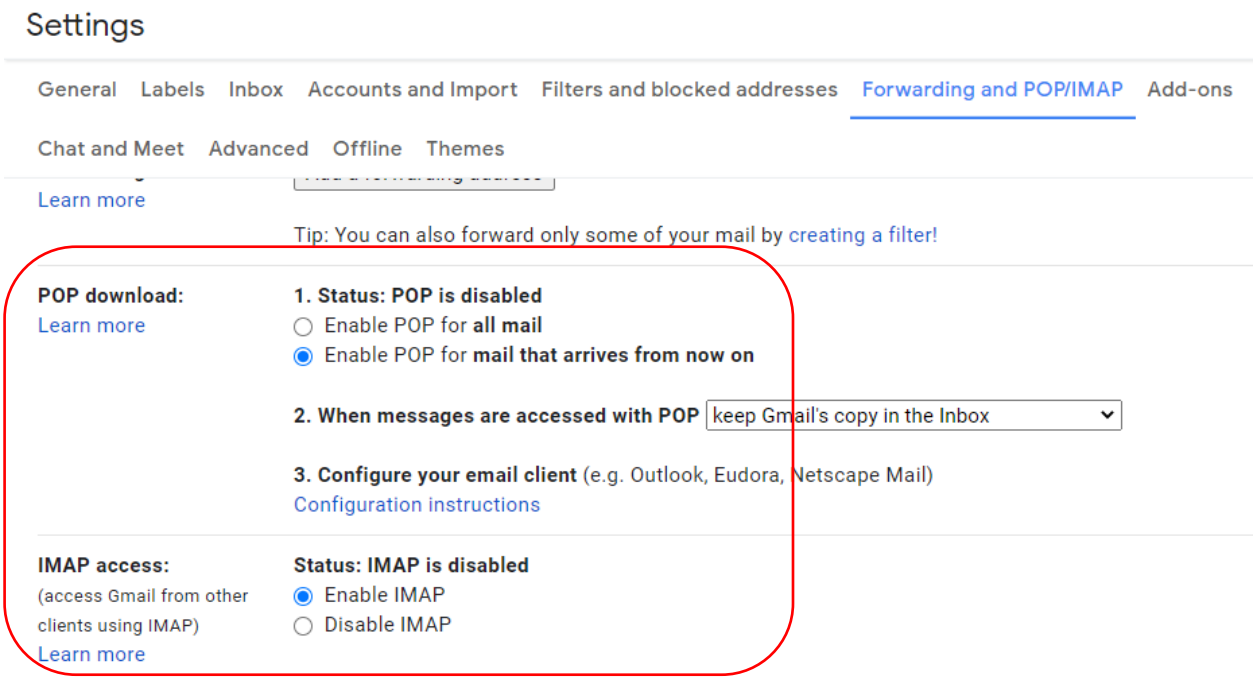


3. Turn “Allow less secure apps” on.



4. From your Gmail account:

- Click on Settings
- Click on Forwarding and POP/IMAP
- Enable POP (either options)
- Enable IMAP

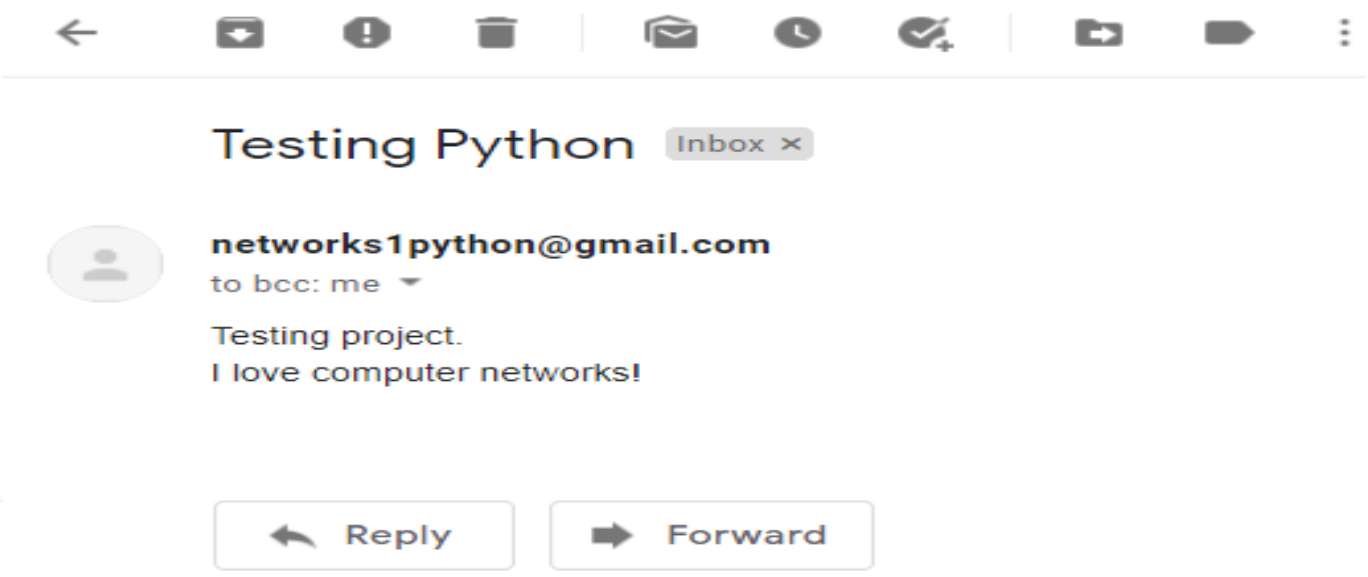


5. Re-run the main program and it should work.

Case II: Successful Connection

```
>>> runfile('C:/Users/yaraa/Desktop/Project/main.py', wdir='C:/Users/yaraa/Desktop/Project')
Enter Your Email Address: >? networks1python@gmail.com
Warning: Password input may be echoed.
Enter Your Password: >? networks1_SMTP
Enter Email Destination: >? networks1python@gmail.com
Enter Email Subject: >? Testing Python
Enter Email Body Message: >? Testing project
b'220 smtp.gmail.com ESMTP m81sm8752906wmf.29 - gsmtp\r\n'
220 reply not received from server.
250 smtp.gmail.com at your service

b'334 VXNlcm5hbWU6\r\n'
334 UGFzc3dvcmQ6

235 2.7.0 Accepted

250 2.1.0 OK m81sm8752906wmf.29 - gsmtp

354  Go ahead m81sm8752906wmf.29 - gsmtp

250 2.0.0 OK  1609361486 m81sm8752906wmf.29 - gsmtp

221 2.0.0 closing connection m81sm8752906wmf.29 - gsmtp

Was successful!

>>>
```

Email Received:



The Code

```
# Portable password input
from getpass import getpass

from socket import *

# Encoding used to convert bytes that have binary or text data into ASCII characters
from base64 import *

# Secure Sockets Layer (used to create a secure connection between server and client)
import ssl


SenderEmail = input("Enter Your Email Address: ")
SenderPassword = getpass("Enter Your Password: ")
ReceiverEmail = input("Enter Email Destination: ")
Subject = input("Enter Email Subject: ")
EmailBody = input("Enter Email Body Message: ")

# Message included in body
msg = '{}. \r\nI love computer networks!'.format(EmailBody)
endmsg = '\r\n.\r\n'

# Choose a mail server (e.g. Google mail server) and call it mailserver
mailServer = 'smtp.gmail.com'
mailPort = 587

#Fill in start

# Creating socket called clientSocket
clientSocket = socket(AF_INET, SOCK_STREAM)

# Establishing a TCP connection with mailserver
clientSocket.connect((mailServer, mailPort))

#Fill in end

confMsg = clientSocket.recv(1024)
print (confMsg)
if confMsg[:3] != '220':
   print ('220 reply not received from server.')

# Send HELO command and print server response.
heloCommand = 'HELO Alice\r\n'.encode()
clientSocket.send(heloCommand)
recv1 = clientSocket.recv(1024).decode()
```

```
print (recv1)
if recv1[:3] != '250':
   print ('250 reply not received from server.')

# Account Authentication
# Fill in start
strtlscmd = "STARTTLS\r\n".encode()
clientSocket.send(strtlscmd)
confMsg2 = clientSocket.recv(1024).decode()

sslClientSocket = ssl.wrap_socket(clientSocket)

EMAIL_ADDRESS = b64encode(SenderEmail.encode())
EMAIL_PASSWORD = b64encode(SenderPassword.encode())

authorizationcmd = "AUTH LOGIN\r\n"

sslClientSocket.send(authorizationcmd.encode())
confMsg2 = sslClientSocket.recv(1024)
print(confMsg2)

sslClientSocket.send(EMAIL_ADDRESS + "\r\n".encode())
confMsg3 = sslClientSocket.recv(1024).decode()
print(confMsg3)

sslClientSocket.send(EMAIL_PASSWORD + "\r\n".encode())
confMsg4 = sslClientSocket.recv(1024).decode()
print(confMsg4)
# Fill in end

# Send MAIL FROM command and print server response.
# Fill in start
mailfrom = "MAIL FROM: <{}>\r\n".format(SenderEmail)
sslClientSocket.send(mailfrom.encode())
confMsg5 = sslClientSocket.recv(1024).decode()
print(confMsg5)
# Fill in end

# Send RCPT TO command and print server response.
# Fill in start
rcptto = "RCPT TO: <{}>\r\n".format(ReceiverEmail)
sslClientSocket.send(rcptto.encode())
confMsg6 = sslClientSocket.recv(1024).decode()
# Fill in end

# Send DATA command and print server response.
# Fill in start
data = 'DATA\r\n'
sslClientSocket.send(data.encode())
confMsg7 = sslClientSocket.recv(1024).decode()
print(confMsg7)
```

```
# Fill in end

# Send message data.
# Fill in start
sslClientSocket.send("Subject: {}\n\n{}".format(Subject, msg).encode())
# Fill in end

# Message ends with a single period.
# Fill in start
sslClientSocket.send(endmsg.encode())
confMsg8 = sslClientSocket.recv(1024).decode()
print(confMsg8)
# Fill in end

# Send QUIT command and get server response.
# Fill in start
quitcommand = 'QUIT\r\n'
sslClientSocket.send(quitcommand.encode())
confMsg9 = sslClientSocket.recv(1024).decode()
print(confMsg9)

sslClientSocket.close()
print('Was successful!')
# Fill in end
```