Python Gmail Email

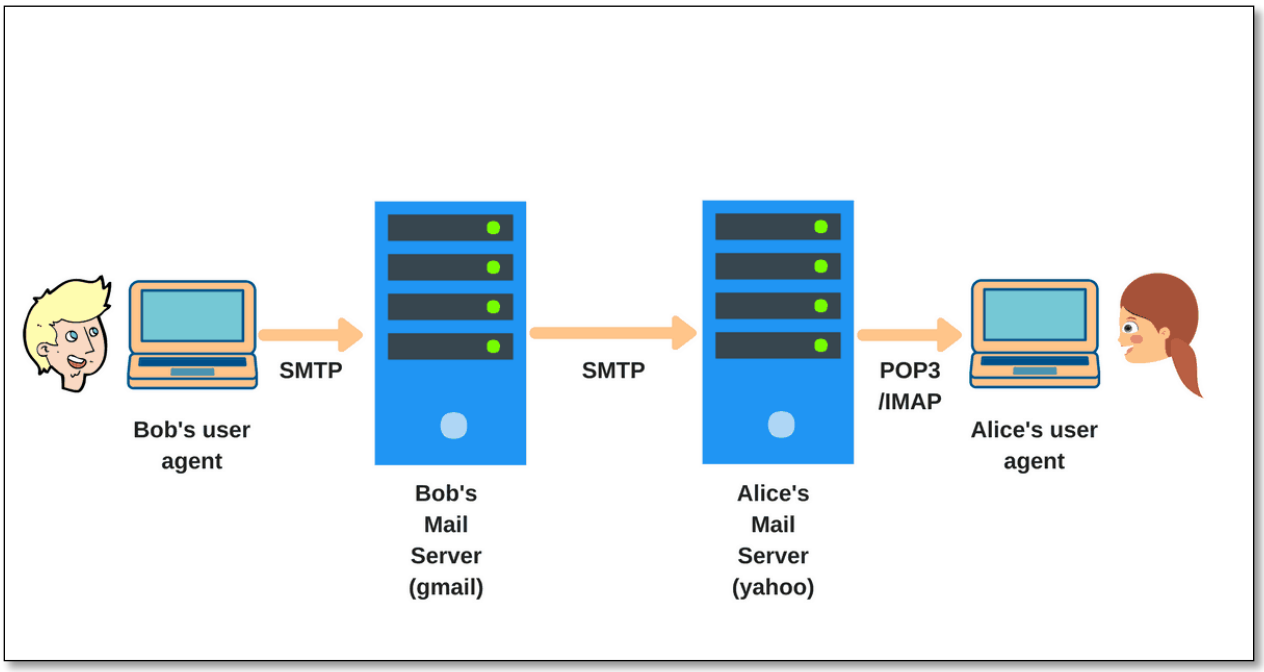
Contents

Time required: 30 minutes

Let's learn how to send email using Python and Gmail. Being able to send an email from Python opens up all sorts of possibilities. We will use this email program in later programs.

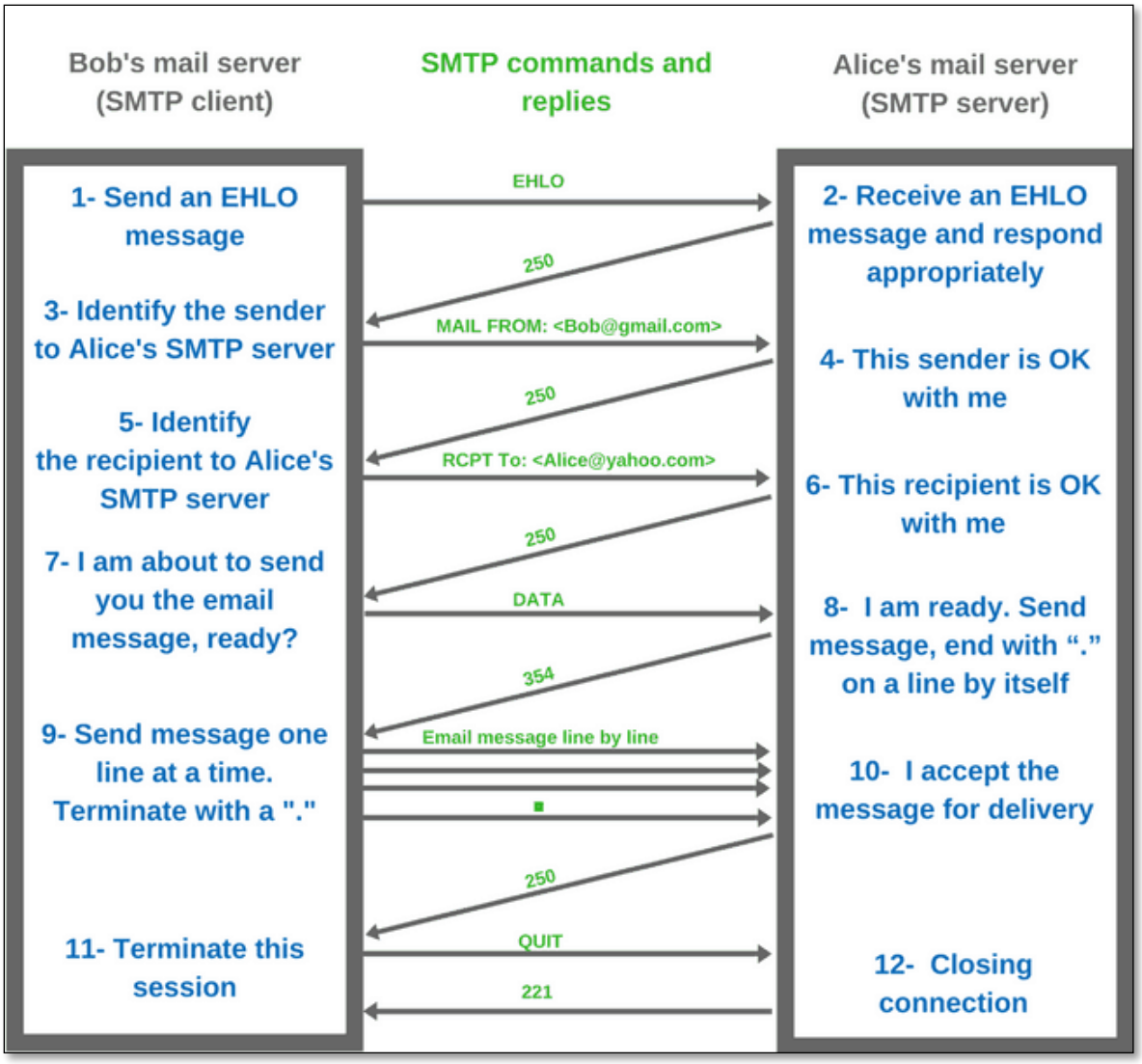
SMTP

Simple Mail Transport Protocol (SMTP).



Let's follow the email message as it travels from Bob to Alice at a high level.

1. Bob opens his Mail app, provides Alice e-mail address (Alice@yahoo.com), writes his message, and clicks the “send” button
2. The Mail app starts communicating with Bob’s mail server and eventually push the email that Bob composed to Bob’s mail server where it is stored to be delivered later to Alice@yahoo.com.
3. Bob’s mail server sees that there is a message pending delivery to Alice@yahoo.com. It starts a communication with the yahoo.com mail server to allow for this message delivery to happen. It is here where the SMTP protocol comes into play. SMTP is the protocol that governs the communication between these two mail servers. In our particular scenario, Bob’s mail server will play the role of an SMTP client while Alice’s mail server will play the role of an SMTP server.
4. After some initial SMTP handshaking between the gmail and yahoo mail servers, the SMTP client sends Bob’s message to Alice’s mail server.
5. Alice’s mail server receives the message and stores it in her mailbox so that she can read it later.
6. Alice uses her Microsoft Outlook to fetch messages from her mailbox and eventually reads Bob’s message.



Gmail uses the secure ports 465 and 587 for SMTP. The port you use depends on whether you're using SSL or TLS encryption.

465 - Use this port for SSL encryption. However, port 465 is a deprecated standard, so most SMTP servers block connections made using this port.

587 - Use this port for TLS encryption. You can configure your on-premise mail server to point to smtp-relay.gmail.com on port 587.

Setting up a Gmail Account for Email

If you don't have a Gmail account, please create one.

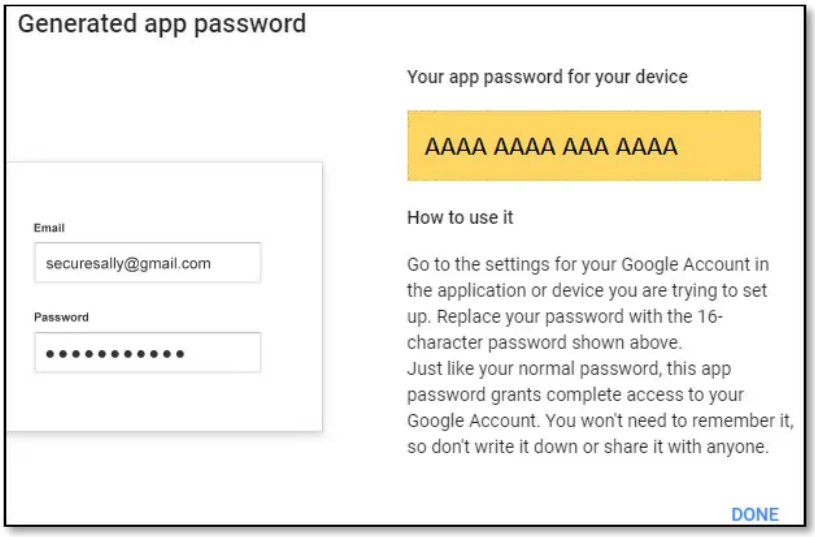
Let's enable your Gmail account to receive connections from external programs, like Python.

1. Open your browser and access your Gmail account.
2. On the login screen → enter your Gmail username and password.
3. After the login → access the following URL: https://myaccount.google.com/signinoptions/two-step-verification
4. Enable the two-step verification on this account.

Create an Application Password

1. Access the following URL: https://security.google.com/settings/security/apppasswords
2. Select Gmail application and the type of device: **Other**.

3. Name the device: **Python**
4. Click on the Generate button and take note of the randomly generated password.



You have finished the required steps for the Gmail integration.

Tutorial: Send Email with Python

Create a Python program named: **gmail_credentials.py**

```python
"""
    Name: gmail_credentials.py
    Author:
    Created:
    Purpose: Credentials to send email through Python using Gmail
"""

SMTP_SERVER = "smtp.gmail.com"
# Secure SMTP port
PORT = 587

#---------- REPLACE WITH YOUR INFORMATION ------------#

LOGIN = "youremailaddress@gmail.com"
APP_PASSWORD = ""

#-----------------------------------------------------#
```

Create a Python program named: **send_gmail.py**

```python
#!/usr/bin/env python3
"""
    Name: send_gmail.py
    Author:
    Created:
    Purpose: Send email with Python and Gmail
"""
# Library to manage communcation with an SMTP server
import smtplib
# Library to create an email message
from email.message import EmailMessage
# Import gmail credentials and SMTP server settings
import gmail_credentials
```

These are the import statements for the different libraries needed.

```python
# -------------- REPLACE WITH YOUR INFORMATION ----------------------------#
email_from = "William Loring <williamaloring@gmail.com>"
# A list containing one or more email addresses
email_dst = [
    "loringw@wncc.edu",
    "williamloring@hotmail.com"
]

subject = "Email from Python Email Program"
message_content = """
This test message is sent from Python Bill.
"""
```

Replace with your email information. Add one of your email addresses to email_dst to make sure the program is working.

```python
# -------------------- CREATE EMAIL MESSAGE -------------------------------#
# Create EmailMessage() object
message = EmailMessage()

# Add properties to message
message["From"] = email_from
message.set_content(message_content)
message["Subject"] = subject
message["To"] = email_dst
```

```python
# ------------------- SEND EMAIL MESSAGE ----------------------------------#
try:
    # Use the with context manager to create an smtp_server object
    # Using a with block automatically closes the connection
    # to the server when the with block is exited
    with smtplib.SMTP(
        gmail_credentials.SMTP_SERVER,
        gmail_credentials.PORT
    ) as smtp_server:
        # Show all communication with the server
        # This line can be commented out
        smtp_server.set_debuglevel(True)

        # Say enhanced hello to the smtp server
        smtp_server.ehlo()

        # Request a TLS connection with the SMTP server
        smtp_server.starttls()

        # Login to the SMTP server
        smtp_server.login(
            gmail_credentials.LOGIN,
            gmail_credentials.APP_PASSWORD
        )

        # Ask smtp_server to send message
        smtp_server.send_message(
            message
        )
    print()
    print(25*"**")
    print("    Email message successfully sent.")
    print(25*"**")
    print()

except Exception as e:
    print(25*"**")
    print(f"Message not sent.")
    print(e)
    print(25*"**")
```

The example run contains all the chatter back and forth between your Python program and the mail server. The debug level is set to True so we can see how much background

communication occurs sending a simple email message. The debug level line can be commented out.

Example run:

```
reply: retcode (250); Msg: b'2.1.0 OK ep25-20020a056870a99900b0011f390fdb0asm361
9396oab.12 - gsmtp'
send: 'rcpt TO:<loringw@wncc.edu>\r\n'
reply: b'250 2.1.5 OK ep25-20020a056870a99900b0011f390fdb0asm3619396oab.12 - gsm
tp\r\n'
reply: retcode (250); Msg: b'2.1.5 OK ep25-20020a056870a99900b0011f390fdb0asm361
9396oab.12 - gsmtp'
send: 'rcpt TO:<williamloring@hotmail.com>\r\n'
reply: b'250 2.1.5 OK ep25-20020a056870a99900b0011f390fdb0asm3619396oab.12 - gsm
tp\r\n'
reply: retcode (250); Msg: b'2.1.5 OK ep25-20020a056870a99900b0011f390fdb0asm361
9396oab.12 - gsmtp'
send: 'data\r\n'
reply: b'354  Go ahead ep25-20020a056870a99900b0011f390fdb0asm3619396oab.12 - gs
mtp\r\n'
reply: retcode (354); Msg: b'Go ahead ep25-20020a056870a99900b0011f390fdb0asm361
9396oab.12 - gsmtp'
data: (354, b'Go ahead ep25-20020a056870a99900b0011f390fdb0asm3619396oab.12 - gs
mtp')
send: b'From: William Loring <williamaloring@gmail.com>\r\nContent-Type: text/pl
ain; charset="utf-8"\r\nContent-Transfer-Encoding: 7bit\r\nMIME-Version: 1.0\r\n
Subject: Email from Python Email Program\r\nTo: loringw@wncc.edu, williamloring@
hotmail.com\r\n\r\n\r\nThis test message is sent from Python Bill.\r\n.\r\n'
reply: b'250 2.0.0 OK  1662293307 ep25-20020a056870a99900b0011f390fdb0asm3619396
oab.12 - gsmtp\r\n'
reply: retcode (250); Msg: b'2.0.0 OK  1662293307 ep25-20020a056870a99900b0011f3
90fdb0asm3619396oab.12 - gsmtp'
data: (250, b'2.0.0 OK  1662293307 ep25-20020a056870a99900b0011f390fdb0asm361939
6oab.12 - gsmtp')
send: 'QUIT\r\n'
reply: b'221 2.0.0 closing connection ep25-20020a056870a99900b0011f390fdb0asm361
9396oab.12 - gsmtp\r\n'
reply: retcode (221); Msg: b'2.0.0 closing connection ep25-20020a056870a99900b00
11f390fdb0asm3619396oab.12 - gsmtp'

**************************************************
    Email message successfully sent.
**************************************************
```

Test your program.

Assignment Submission

- Send the instructor an email at loringw@wncc.edu
- Insert a screenshot of the program run
- Attach the completed program files
- Submit the assignment in Blackboard.