**Setting up a Gmail Account for Development**

If you decide to use a Gmail account to send your emails, I highly recommend setting up a throwaway account for the development of your code. This is because you’ll have to adjust your Gmail account’s security settings to allow access from your Python code, and because there’s a chance you might accidentally expose your login details. Also, I found that the inbox of my testing account rapidly filled up with test emails, which is reason enough to set up a new Gmail account for development.

A nice feature of Gmail is that you can use the + sign to add any modifiers to your email address, right before the @ sign. For example, mail sent to my+person1@gmail.com and my+person2@gmail.com will both arrive at my@gmail.com. When testing email functionality, you can use this to emulate multiple addresses that all point to the same inbox.

**To set up a Gmail address for testing your code, do the following:**

Create a new Google account.

Turn Allow less secure apps to ON. Be aware that this makes it easier for others to gain access to your account.

If you don’t want to lower the security settings of your Gmail account, check out Google’s documentation on how to gain access credentials for your Python script, using the OAuth2 authorization framework.

**Sending Your Plain-text Email**

import smtplib, ssl

smtp\_server = "smtp.gmail.com"

port = 587 # For starttls

sender\_email = "my@gmail.com"

password = input("Type your password and press enter: ")

# Create a secure SSL context

context = ssl.create\_default\_context()

# Try to log in to server and send email

try:

server = smtplib.SMTP(smtp\_server,port)

server.ehlo() # Can be omitted

server.starttls(context=context) # Secure the connection

server.ehlo() # Can be omitted

server.login(sender\_email, password)

# TODO: Send email here

server.sendmail(sender\_email, receiver\_email, message)

except Exception as e:

# Print any error messages to stdout

print(e)

finally:

server.quit()

**FANCY EMAIL**

import smtplib, ssl

from email.mime.text import MIMEText

from email.mime.multipart import MIMEMultipart

sender\_email = "my@gmail.com"

receiver\_email = "your@gmail.com"

password = input("Type your password and press enter:")

message = MIMEMultipart("alternative")

message["Subject"] = "multipart test"

message["From"] = sender\_email

message["To"] = receiver\_email

# Create the plain-text and HTML version of your message

text = """\

Hi,

How are you?

Real Python has many great tutorials:

www.realpython.com"""

html = """\

<html>

<body>

<p>Hi,<br>

How are you?<br>

<a href="http://www.realpython.com">Real Python</a>

has many great tutorials.

</p>

</body>

</html>

"""

# Turn these into plain/html MIMEText objects

part1 = MIMEText(text, "plain")

part2 = MIMEText(html, "html")

# Add HTML/plain-text parts to MIMEMultipart message

# The email client will try to render the last part first

message.attach(part1)

message.attach(part2)

# Create secure connection with server and send email

context = ssl.create\_default\_context()

with smtplib.SMTP\_SSL("smtp.gmail.com", 465, context=context) as server:

server.login(sender\_email, password)

server.sendmail(

sender\_email, receiver\_email, message.as\_string()

)

**WITH ATTACHMENT**

import email, smtplib, ssl

from email import encoders

from email.mime.base import MIMEBase

from email.mime.multipart import MIMEMultipart

from email.mime.text import MIMEText

subject = "An email with attachment from Python"

body = "This is an email with attachment sent from Python"

sender\_email = "my@gmail.com"

receiver\_email = "your@gmail.com"

password = input("Type your password and press enter:")

# Create a multipart message and set headers

message = MIMEMultipart()

message["From"] = sender\_email

message["To"] = receiver\_email

message["Subject"] = subject

message["Bcc"] = receiver\_email # Recommended for mass emails

# Add body to email

message.attach(MIMEText(body, "plain"))

filename = "document.pdf" # In same directory as script

# Open PDF file in binary mode

with open(filename, "rb") as attachment:

# Add file as application/octet-stream

# Email client can usually download this automatically as attachment

part = MIMEBase("application", "octet-stream")

part.set\_payload(attachment.read())

# Encode file in ASCII characters to send by email

encoders.encode\_base64(part)

# Add header as key/value pair to attachment part

part.add\_header(

"Content-Disposition",

f"attachment; filename= {filename}",

)

# Add attachment to message and convert message to string

message.attach(part)

text = message.as\_string()

# Log in to server using secure context and send email

context = ssl.create\_default\_context()

with smtplib.SMTP\_SSL("smtp.gmail.com", 465, context=context) as server:

server.login(sender\_email, password)

server.sendmail(sender\_email, receiver\_email, text)