Practical Assignment 5

Write a program using python to send an email with an attachment securely to the intended recipient.

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)

Another Example

Step 1: Connect to Gmail’s SMTP Server

The first step is to connect to Gmail’s SMTP server. To do this, you need to create an object of the SMTP class using the smtplib module. In the constructor of the SMTP class object, pass the address of Gmail’s SMTP server and the port you’ll be using to connect to the server.

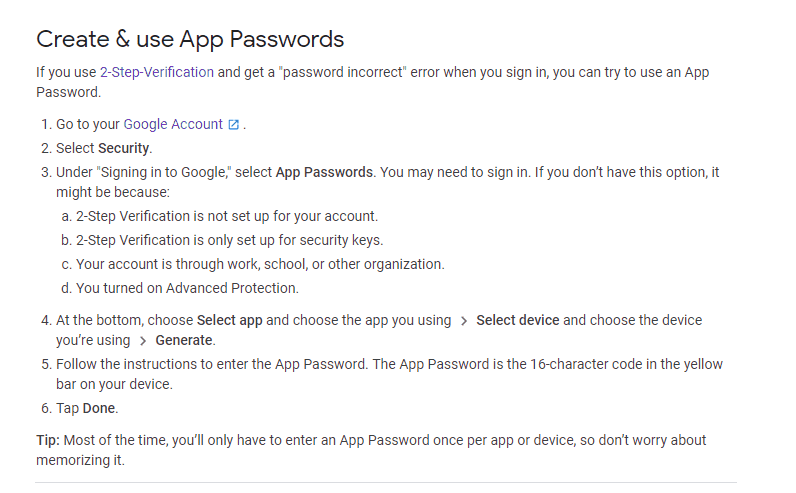
import smtplib

smtp\_obj = smtplib.SMTP("smtp.gmail.com", 587)

smtp\_obj.ehlo()

While establishing connection with the Gmail’s SMTP server, you mentioned 587 as the port number. The port 587 uses TLS encryption to send and receive data in a secure way. Therefore, you need to specify with your SMTP object that the data you are going to send or receive should be TLS encrypted. To do so, you need to call the starttls() method from the SMTP class object. If you are using a port other than port 587, you don’t need to include the following script:

Step 2: Generate Gmail Password for Python App



After enabling 2-step verification, go to your Google account and click “Security.”

Scroll down until you see the option “Signing in to Google.” Here, click the “App passwords” option.

From the “Select app” drop down list, choose “Mail”

Enter any name for your application and click the “GENERATE” button

Step 3: Build Email with Python

import smtplib

smtp\_obj = smtplib.SMTP("smtp.gmail.com", 587)

smtp\_obj.ehlo()

smtp\_obj.starttls()

import getpass

my\_email = getpass.getpass("Enter your email:")

my\_pass = getpass.getpass("Enter your app password:")

smtp\_obj.login(my\_email, my\_pass)

to\_email = getpass.getpass("Enter email of the receiver:")

email\_from\_address = my\_email

email\_to\_address = to\_email

email\_subject = input("Enter the subject of the email: ")

email\_body = input("Enter the body of the email: ")

email\_text = "Subject: "+email\_subject+"\n"+email\_body

smtp\_obj.sendmail(email\_from\_address, email\_to\_address, email\_text)

smtp\_obj.quit()