

This is your **last** free member-only story this month. [Upgrade for unlimited access.](#)

Python Script to Send Emails

Code and explanation to send emails using Python.



Harendra Verma

Follow



Oct 29 · 6 min read ★



Mail

BBC News

Emails are a typical method of official communication nowadays, and they are also useful for transferring files from one person to another. Almost everyone who has an online identity or merely utilizes it has their email address, whether it's Gmail or Outlook.

One of the many cool things you can do with Python is to send and receive emails. Python programming libraries may be used to send emails or to display a list of all the emails in your inbox. Python may also be used to handle certain simple tasks, such as marking emails as read.

And I'm going to show you how to do it yourself in this blog.

Regarding Mail Servers

Before we get started with the code, you need to have a fundamental understanding of email and mail servers.

Mail servers are servers that manage emails. For example, Outlook, Gmail, Yahoo, and Hotmail all have their mail servers that handle respective email services. Mail servers may be further divided into two types:

SMTP (Simple Mail Transfer Protocol) Server

Simple Mail Transfer Protocol (SMTP) is an acronym for Simple Mail Transfer Protocol.

This server is in charge of transmitting or transferring mail from one server to another; for example, when you send an email to someone, you usually utilize the SMTP server.

IMAP (Internet Message Access Protocol) Server

Internet Message Access Protocol (IMAP) is an acronym for Internet Message Access Protocol.

This server is in charge of storing and listing messages from your server; for example, the IMAP server is often used when you access Gmail or Outlook.

SSL and TLS





Photo by [Markus Winkler](#) on [Unsplash](#)

SSL (Secure Socket Layer) and TLS (Transport Layer Security) are the two encryption methods used for emails (Transport Layer Security).

You will use one of these protocols to connect to any mail server.

The server is allocated a port for each protocol.

SSL — port 465 TLS — port 587

Although both protocols are supported by Gmail and Outlook, we will only utilize TLS in this tutorial for simplicity.

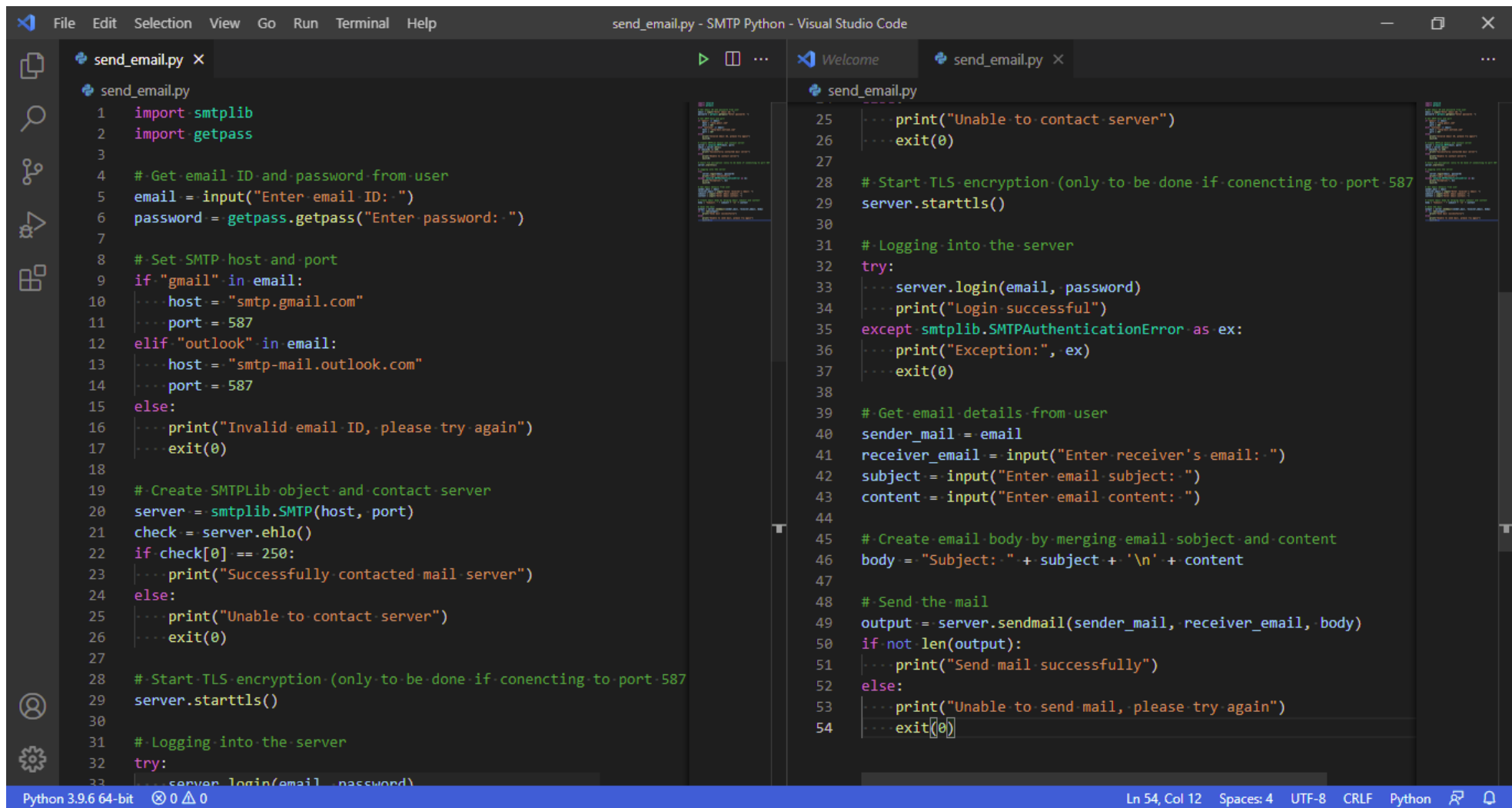
Sending Mails Using Python

Now that we've covered mail servers, let's write our first python script to deliver mail.

To send Gmail or Outlook mail, we'll use Python's smtplib package. You don't need to download this library because it's already built into Python.

Creating the Script

Here's how the script would look like:



```
1 import smtplib
2 import getpass
3
4 # Get email ID and password from user
5 email = input("Enter email ID: ")
6 password = getpass.getpass("Enter password: ")
7
8 # Set SMTP host and port
9 if "gmail" in email:
10     host = "smtp.gmail.com"
11     port = 587
12 elif "outlook" in email:
13     host = "smtp-mail.outlook.com"
14     port = 587
15 else:
16     print("Invalid email ID, please try again")
17     exit(0)
18
19 # Create SMTPLib object and contact server
20 server = smtplib.SMTP(host, port)
21 check = server.ehlo()
22 if check[0] == 250:
23     print("Successfully contacted mail server")
24 else:
25     print("Unable to contact server")
26     exit(0)
27
28 # Start TLS encryption (only to be done if connecting to port 587)
29 server.starttls()
30
31 # Logging into the server
32 try:
33     server.login(email, password)
34
35 ... print("Unable to contact server")
36 ... exit(0)
37
38 # Start TLS encryption (only to be done if connecting to port 587)
39 server.starttls()
40
41 # Logging into the server
42 try:
43     server.login(email, password)
44     print("Login successful")
45 except smtplib.SMTPAuthenticationError as ex:
46     print("Exception:", ex)
47     exit(0)
48
49 # Get email details from user
50 sender_email = email
51 receiver_email = input("Enter receiver's email: ")
52 subject = input("Enter email subject: ")
53 content = input("Enter email content: ")
54
55 # Create email body by merging email subject and content
56 body = "Subject: " + subject + '\n' + content
57
58 # Send the mail
59 output = server.sendmail(sender_email, receiver_email, body)
60 if not len(output):
61     print("Send mail successfully")
62 else:
63     print("Unable to send mail, please try again")
64     exit(0)
```

Vs Code Screenshot

The script may appear confusing, but we'll go over each function and class of the script line by line so you can understand how to utilize it.

```
import smtplib
import getpass
```

piece of code

Importing the smtplib and getpass libraries is the first step in the script. We're going to utilize the getpass library to get the password from the user.

```
# Get email ID and password from user
email = input("Enter email ID: ")
password = getpass.getpass("Enter password: ")
```

Following that, we ask the user to provide their mail account details, which will be used to send the email. To ask the user for their password, we use `getpass()`. Because we're using `getpass()`, the user's password won't be shown on the screen; instead, it'll be kept in the variable.

```
# Set SMTP host and port
if "gmail" in email:
    host = "smtp.gmail.com"
    port = 587
elif "outlook" in email:
    host = "smtp-mail.outlook.com"
    port = 587
else:
    print("Invalid email ID, please try again")
    exit(0)
```

The next step is to configure the SMTP server host and port. The host will be set to Gmail SMTP server if the supplied email ID is a Gmail account, or Outlook SMTP server if it is an Outlook account.

As we stated in the 'SSL and TLS' section, we changed the port to 587. The script will deliver an error message and terminate if the entered email ID cannot be recognized as Gmail or Outlook.

```
# Create SMTPLib object and contact server
server = smtplib.SMTP(host, port)
check = server.ehlo()
if check[0] == 250:
    print("Successfully contacted mail server")
else:
    print("Unable to contact server")
    exit(0)
```

The SMTP class object, which will be used to conduct the tasks, will be created next. We make an object of the `smtplib.SMTP` class and store it with the nameserver.' The hostname and port are required arguments for the class object.

We call the `ehlo()` method of the class object once we've constructed the object, which is used to send a welcome message to the mail server. This step is critical, since failing to do so may result in communication issues with the mail server.

```
# Start TLS encryption (only to be done if connecting to port 587)
server.starttls()
```

We invoke the `starttls()` method to start TLS encryption after obtaining a successful response from the server. This step is only necessary for TLS connections; SSL connections do not require it.

```
# Logging into the server
try:
    server.login(email, password)
    print("Login successful")
except smtplib.SMTPAuthenticationError as ex:
    print("Exception:", ex)
    exit(0)
```

The `login()` method is then used to log into the mail account. The email ID and password, which we had acquired from the user, are required arguments for the function.

```
# Get email details from user
sender_mail = email
receiver_email = input("Enter receiver's email: ")
subject = input("Enter email subject: ")
content = input("Enter email content: ")
```

We ask the user for the recipient's email address, the email's topic, and the email's content after they have successfully logged in.

```
# Send the mail
output = server.sendmail(sender_mail, receiver_email, body)
if not len(output):
    print("Send mail successfully")
else:
    print("Unable to send mail, please try again")
    exit(0)
```

Finally, we execute the `sendmail()` method with three parameters: the sender mail ID, the recipient mail ID, and the message contents (created by merging mail subject and mail content).

Here is the full code:

```
import smtplib
import getpass
```

```

# Get email ID and password from user
email = input("Enter email ID: ")
password = getpass.getpass("Enter password: ")

# Set SMTP host and port
if "gmail" in email:
    host = "smtp.gmail.com"
    port = 587
elif "outlook" in email:
    host = "smtp-mail.outlook.com"
    port = 587
else:
    print("Invalid email ID, please try again")
    exit(0)

# Create SMTPLib object and contact server
server = smtplib.SMTP(host, port)
check = server.ehlo()
if check[0] == 250:
    print("Successfully contacted mail server")
else:
    print("Unable to contact server")
    exit(0)

# Start TLS encryption (only to be done if connecting to port 587
# i.e. TLS)
server.starttls()

# Logging into the server
try:
    server.login(email, password)
    print("Login successful")
except smtplib.SMTPAuthenticationError as ex:
    print("Exception:", ex)
    exit(0)

# Get email details from user
sender_mail = email
receiver_email = input("Enter receiver's email: ")
subject = input("Enter email subject: ")

```

```
content = input("Enter email content: ")

# Create email body by merging emails object and content
body = "Subject: " + subject + '\n' + content

# Send the mail
output = server.sendmail(sender_email, receiver_email, body)
if not len(output):
    print("Send mail successfully")
else:
    print("Unable to send mail, please try again")
    exit(0)
```

And that's it, you have successfully sent a mail through Python.

Conclusion

I've just discussed sending mail with Python in this blog. I'll be starting another blog shortly on getting emails from accounts using python to contact IMAP servers.

Thank you for your interest in reading this article. This knowledge should help you in sending emails using python, in my opinion.

Don't forget to follow me on [Medium](#) if you want to see more pieces like this and clap for this story.

More content at plainenglish.io

Get an email whenever Harendra Verma publishes.

 **Subscribe**

Emails will be sent to mjaworsk@stevens.edu.

[Not you?](#)

Python

Software Development

Programming

Python3

Software Engineering

Learn more.

Medium is an open platform where 170 million readers come to find insightful and dynamic thinking. Here, expert and undiscovered voices alike dive into the heart of any topic and bring new ideas to the surface. [Learn more](#)

Make Medium yours.

Follow the writers, publications, and topics that matter to you, and you'll see them on your homepage and in your inbox. [Explore](#)

Write a story on Medium.

If you have a story to tell, knowledge to share, or a perspective to offer — welcome home. It's easy and free to post your thinking on any topic. [Start a blog](#)



[About](#) [Write](#) [Help](#) [Legal](#)