Asingh

Apr 6 · 3 min read · Listen

Email automation project using Python and the `smtplib` and `email` libraries.

> *"This was created using ChatGPT"*

Introduction

Email automation is a process of sending pre-written emails to a large number of recipients automatically. In this tutorial, we will show you how to build an email automation project using Python and the `smtplib` and `email` libraries.

Prerequisites

Before starting this tutorial, you should have some basic knowledge of Python programming and be familiar with the `smtplib` and `email` libraries.

Step 1: Set Up Email Parameters

The first step in creating an email automation project is to set up the email parameters. This includes the sender's email address, password, recipient's email address, subject, and message.

```
email = 'your_email@example.com'
password = 'your_password'
send_to_email = 'recipient_em
```

```
subject = 'This is the subject'
message = 'This is my message'
```

Replace the email and password with your own email and password, and the send_to_email with the recipient's email address.

Step 2: Create the MIME Message

The next step is to create a MIME (Multipurpose Internet Mail Extensions) message using the `MIMEMultipart` class and attach a plain text message using the `MIMEText` class.

```
from email.mime.text import MIMEText
from email.mime.multipart import MIMEMultipart

msg = MIMEMultipart()
msg['From'] = email
msg['To'] = send_to_email
msg['Subject'] = subject
msg.attach(MIMEText(message, 'plain'))
```

Here, we first import the `MIMEText` and `MIMEMultipart` classes from the `email.mime.text` and `email.mime.multipart` modules, respectively. Then, we create a new `MIMEMultipart` object and set the `From`, `To`, and `Subject` headers using the `msg['From']`, `msg['To']`, and `msg['Subject']` attributes, respectively. Finally, we attach a plain text message using the `MIMEText` class and the `msg.attach()` method.

Step 3: Connect to SMTP Server and Send Email

The final step is to connect to the SMTP (Simple Mail Transfer Protocol) server using the `smtplib.SMTP()` function, login to the email account using the `server.login()` method, convert the MIME message to a string using the `msg.as_string()` method, and send the email using the `server.sendmail()` method.

```
import smtplib

try:
    server = smtplib.SMTP('smtp.gmail.com', 587)
    server.starttls()
    server.login(email, password)
    text = msg.as_string()
    server.sendmail(email, send_to_email, text)
    print('Email sent successfully!')
except Exception as e:
    print('Something went wrong:', e)
finally:
    server.quit()
```

Here, we first import the `smtplib` library and create a new `SMTP` object by calling the `smtplib.SMTP()` function with the hostname and port number of the SMTP server (in this case, the Gmail SMTP server). We then start the TLS (Transport Layer Security) encryption using the `server.starttls()` method to secure the connection.

Next, we login to the email account using the `server.login()` method with the email and password parameters. We convert the MIME message to a string using the `msg.as_string()` method and send the email using the `server.sendmail()` method with the email, send_to_email, and text parameters.

Finally, we handle any exceptions that may occur during the process using a `try-except` block and quit the server connection using the `server.quit()` method.

Putting it All Together

Now that we have covered the individual steps, let's put it all together in a complete script:

```
import smtplib
from email.mime.text import MIMEText
from email.mime.multipart import MIMEMultipart

email = 'your_email@example.com'
password = 'your_password'
```

```
send_to_email = 'recipient_email@example.com'
subject = 'This is the subject'
message = 'This is my message'

msg = MIMEMultipart()
msg['From'] = email
msg['To'] = send_to_email
msg['Subject'] = subject
msg.attach(MIMEText(message, 'plain'))

try:
    server = smtplib.SMTP('smtp.gmail.com', 587)
    server.starttls()
    server.login(email, password)
    text = msg.as_string()
    server.sendmail(email, send_to_email, text)
    print('Email sent successfully!')
except Exception as e:
    print('Something went wrong:', e)
finally:
    server.quit()
```

Conclusion

In this tutorial, we have shown you how to build an email automation project using Python and the `smtplib` and `email` libraries. With this project, you can send pre-written emails to a large number of recipients automatically. You can further enhance this project by adding features such as email templates, email scheduling, and email tracking.