How to Send Automated Email Messages in Python

25 May 2025 10:03 PM

In this article, we are going to see how to send automated email messages which involve delivering text messages, essential photos, and important files, among other things. in Python.

We'll be using two libraries for this: email, and smtplib, as well as the MIMEMultipart object. This object has multiple subclasses; these subclasses will be used to build our email message.

- **MIMEText:** It consists of simple text. This will be the body of the email.
- **MIMEImage:** This would allow us to add images to our emails.
- **MIMEAudio:** If we wish to add audio files, we may do it easily with the help of this subclass.
- **MIMEApplication:** This can be used to add anything or any other attachments.

Step-by-step Implementation

Step 1: Import the following modules

```
from email.mime.text import MIMEText
from email.mime.image import MIMEImage
from email.mime.application import MIMEApplication
from email.mime.multipart import MIMEMultipart
import smtplib
import os
```

Step 2: Let's set up a connection to our email server.

- Provide the server address and port number to initiate our **SMTP** connection
- Then we'll use **smtp.ehlo** to send an **EHLO** (Extended Hello) command.
- Now, we'll use **smtp.starttls** to enable transport layer security (**TLS**) encryption.

```
smtp = smtplib.SMTP('smtp.gmail.com', 587)
smtp.ehlo()
smtp.starttls()
smtp.login('YourMail@gmail.com', 'Your Password')
```

Step 3: Now, built the message content.

- Assign the **MIMEMultipart** object to the msg variable after initializing it.
- The **MIMEText** function will be used to attach text.

```
msg = MIMEMultipart()
msg['Subject'] = subject
msg.attach(MIMEText(text))
```

Step 4: Let's look at how to attach pictures and multiple attachments.

Attaching Images:

- First, read the image as binary data.
- Attach the image data to **MIMEMultipart** using **MIMEImage**, we add the given filename use **os.basename**

```
img_data = open(one_img, 'rb').read()
msg.attach(MIMEImage(img_data,
                     name=os.path.basename(one_img)))
```

Attaching Several Files:

- Read in the attachment using **MIMEApplication**.
- Then we edit the attached file metadata.
- Finally, add the attachment to our message object.

```
with open(one_attachment, 'rb') as f:
    file = MIMEApplication(
        f.read(), name=os.path.basename(one_attachment)
    )
    file['Content-Disposition'] = f'attachment; \
    filename="{os.path.basename(one_attachment)}"'
    msg.attach(file)
```

Step 5: The last step is to send the email.

- Make a list of all the emails you want to send.
- Then, by using the **sendmail** function, pass parameters such as from where, to where, and the message content.
- At last, just quit the server connection.

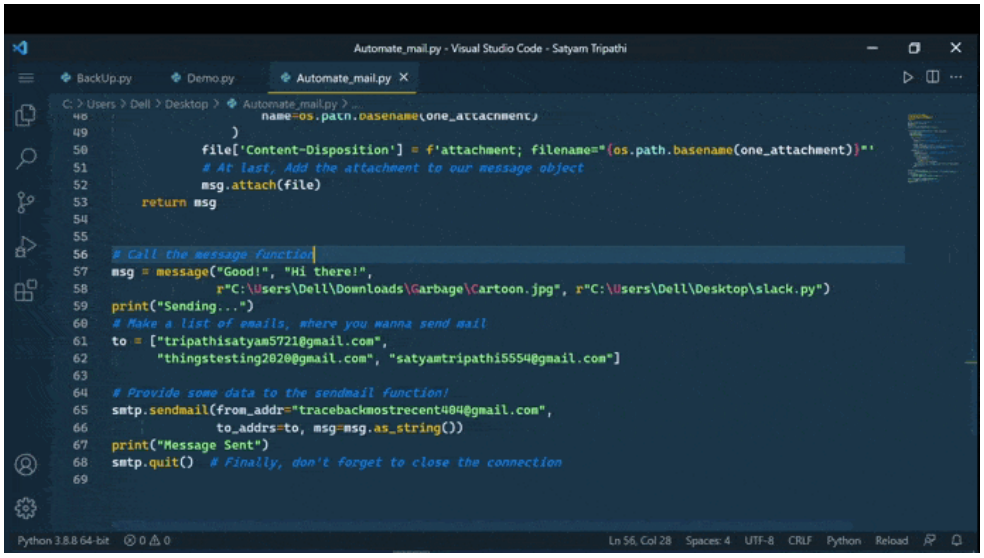
```
to = ["klm@gmail.com", "xyz@gmail.com", "abc@gmail.com"]
smtp.sendmail(from_addr="Your Login Email",
              to_addrs=to, msg=msg.as_string())
smtp.quit()
```

Below is the full implementation:

```
# Import the following module
from email.mime.text import MIMEText
from email.mime.image import MIMEImage
from email.mime.application import MIMEApplication
from email.mime.multipart import MIMEMultipart
import smtplib
import os
# initialize connection to our
# email server, we will use gmail here
smtp = smtplib.SMTP('smtp.gmail.com', 587)
```

```
smtp.ehlo()
smtp.starttls()
# Login with your email and password
smtp.login('Your Email', 'Your Password')
# send our email message 'msg' to our boss
def message(subject="Python Notification",
            text="", img=None,
            attachment=None):

    # build message contents
    msg = MIMEMultipart()

    # Add Subject
    msg['Subject'] = subject

    # Add text contents
    msg.attach(MIMEText(text))
# Check if we have anything
    # given in the img parameter
    if img is not None:

        # Check whether we have the lists of images or not!
        if type(img) is not list:

            # if it isn't a list, make it one
            img = [img]
# Now iterate through our list
        for one_img in img:

            # read the image binary data
            img_data = open(one_img, 'rb').read()
            # Attach the image data to MIMEMultipart
            # using MIMEImage, we add the given filename use os.basename
            msg.attach(MIMEImage(img_data,
                                 name=os.path.basename(one_img)))
# We do the same for
    # attachments as we did for images
    if attachment is not None:

        # Check whether we have the
        # lists of attachments or not!
        if type(attachment) is not list:

            # if it isn't a list, make it one
            attachment = [attachment]
for one_attachment in attachment:
with open(one_attachment, 'rb') as f:

                # Read in the attachment
                # using MIMEApplication
                file = MIMEApplication(
                    f.read(),
                    name=os.path.basename(one_attachment)
                )
            file['Content-Disposition'] = f'attachment;\
            filename="{os.path.basename(one_attachment)}"'

            # At last, Add the attachment to our message object
            msg.attach(file)
    return msg
# Call the message function
msg = message("Good!", "Hi there!",
              r"C:\Users\Dell\Downloads\Garbage\Cartoon.jpg",
              r"C:\Users\Dell\Desktop\slack.py")
# Make a list of emails, where you wanna send mail
to = ["ABC@gmail.com",
      "XYZ@gmail.com", "insaaf@gmail.com"]
# Provide some data to the sendmail function!
smtp.sendmail(from_addr="hello@gmail.com",
              to_addrs=to, msg=msg.as_string())
# Finally, don't forget to close the connection
smtp.quit()
```

Output:



Schedule Email Messages

For scheduling the mail, we will make use of the **schedule** package in python. It is very lightweight and easy to use.

Install the module

`pip install schedule`

Now look at the different functions that are defined in a schedule module and their use:

The below function will call the function mail every 2 seconds.

`schedule.every(2).seconds.do(mail)`

This will call the function mail every 10 minutes.

`schedule.every(10).minutes.do(mail)`

This will call the function in every hour.

`schedule.every().hour.do(mail)`

Calling every day at 10:30 AM.

`schedule.every().day.at("10:30").do(mail)`

Calling a particular day.

`schedule.every().monday.do(mail)`

Below is the implementation:

```
import schedule
import time
from email.mime.text import MIMEText
from email.mime.image import MIMEImage
from email.mime.application import MIMEApplication
from email.mime.multipart import MIMEMultipart
import smtplib
import os
# send our email message 'msg' to our boss
def message(subject="Python Notification",
        text="", img=None, attachment=None):

    # build message contents
    msg = MIMEMultipart()

    # Add Subject
    msg['Subject'] = subject

    # Add text contents
    msg.attach(MIMEText(text))
# Check if we have anything
    # given in the img parameter
    if img is not None:
# Check whether we have the
        # lists of images or not!
        if type(img) is not list:

            # if it isn't a list, make it one
            img = [img]
# Now iterate through our list
        for one_img in img:

            # read the image binary data
            img_data = open(one_img, 'rb').read()

            # Attach the image data to MIMEMultipart
            # using MIMEImage,
            # we add the given filename use os.basename
```

```
            msg.attach(MIMEImage(img_data,
                                 name=os.path.basename(one_img)))
    # We do the same for attachments
       # as we did for images
       if attachment is not None:
    # Check whether we have the
           # lists of attachments or not!
           if type(attachment) is not list:

               # if it isn't a list, make it one
               attachment = [attachment]
    for one_attachment in attachment:
    with open(one_attachment, 'rb') as f:

                # Read in the attachment using MIMEApplication
                file = MIMEApplication(
                    f.read(),
                    name=os.path.basename(one_attachment)
                )
               file['Content-Disposition'] = f'attachment;\
               filename="{os.path.basename(one_attachment)}"'

               # At last, Add the attachment to our message object
               msg.attach(file)
       return msg
    def mail():

       # initialize connection to our email server,
       # we will use gmail here
       smtp = smtplib.SMTP('smtp.gmail.com', 587)
       smtp.ehlo()
       smtp.starttls()

       # Login with your email and password
       smtp.login('Email', 'Password')
    # Call the message function
       msg = message("Good!", "Hi there!",
                     r"C:\Users\Dell\Downloads\Garbage\Cartoon.jpg",
                     r"C:\Users\Dell\Desktop\slack.py")

       # Make a list of emails, where you wanna send mail
       to = ["ABC@gmail.com",
             "XYZ@gmail.com", "insaaf@gmail.com"]
    # Provide some data to the sendmail function!
       smtp.sendmail(from_addr="hello@gmail.com",
                     to_addrs=to, msg=msg.as_string())

       # Finally, don't forget to close the connection
       smtp.quit()
    schedule.every(2).seconds.do(mail)
    schedule.every(10).minutes.do(mail)
    schedule.every().hour.do(mail)
    schedule.every().day.at("10:30").do(mail)
    schedule.every(5).to(10).minutes.do(mail)
    schedule.every().monday.do(mail)
    schedule.every().wednesday.at("13:15").do(mail)
    schedule.every().minute.at(":17").do(mail)
    while True:
       schedule.run_pending()
       time.sleep(1)
```

Output:

From <https://www.geeksforgeeks.org/how-to-send-automated-email-messages-in-python/>