24. Python 3 – Sending Email using SMTP

Simple Mail Transfer Protocol (SMTP) is a protocol, which handles sending an e-mail and routing e-mail between mail servers.

Python provides smtplib module, which defines an SMTP client session object that can be used to send mails to any Internet machine with an SMTP or ESMTP listener daemon.

Here is a simple syntax to create one SMTP object, which can later be used to send an e-mail-

```
import smtplib

smtpObj = smtplib.SMTP( [host [, port [, local_hostname]]] )
```

Here is the detail of the parameters-

- **host:** This is the host running your SMTP server. You can specifiy IP address of the host or a domain name like tutorialspoint.com. This is an optional argument.
- **port:** If you are providing *host* argument, then you need to specify a port, where SMTP server is listening. Usually this port would be 25.
- **local_hostname**: If your SMTP server is running on your local machine, then you can specify just *localhost* as the option.

An SMTP object has an instance method called **sendmail**, which is typically, used to do the work of mailing a message. It takes three parameters-

- The *sender* - A string with the address of the sender.
- The *receivers* - A list of strings, one for each recipient.
- The *message* - A message as a string formatted as specified in the various RFCs.

Example

Here is a simple way to send one e-mail using Python script. Try it once-

```
#!/usr/bin/python3


import smtplib


sender ='from@fromdomain.com'

receivers =['to@todomain.com']


message ="""From: From Person <from@fromdomain.com>

To: To Person <to@todomain.com>

Subject: SMTP e-mail test
```

```
This is a test e-mail message.
"""
try:
   smtpObj = smtplib.SMTP('localhost')
   smtpObj.sendmail(sender, receivers, message)
print ("Successfully sent email")
except smtplib.SMTPException:
print ("Error: unable to send email")
```

Here, you have placed a basic e-mail in message, using a triple quote, taking care to format the headers correctly. An e-mail requires a **From**, **To**, and a **Subject** header, separated from the body of the e-mail with a blank line.

To send the mail you use *smtpObj* to connect to the SMTP server on the local machine. Then use the *sendmail* method along with the message, the from address, and the destination address as parameters (even though the from and to addresses are within the e-mail itself, these are not always used to route the mail).

If you are not running an SMTP server on your local machine, you can the use*smtplib* client to communicate with a remote SMTP server. Unless you are using a webmail service (such as gmail or Yahoo! Mail), your e-mail provider must have provided you with the outgoing mail server details that you can supply them, as follows-

```
mail=smtplib.SMTP('smtp.gmail.com', 587)
```

Sending an HTML e-mail using Python

When you send a text message using Python, then all the content is treated as simple text. Even if you include HTML tags in a text message, it is displayed as simple text and HTML tags will not be formatted according to the HTML syntax. However, Python provides an option to send an HTML message as actual HTML message.

While sending an e-mail message, you can specify a Mime version, content type and the character set to send an HTML e-mail.

Example

Following is an example to send the HTML content as an e-mail. Try it once-

```
#!/usr/bin/python3

import smtplib

message = """From: From Person <from@fromdomain.com>
To: To Person <to@todomain.com>
```

```
MIME-Version: 1.0
Content-type: text/html
Subject: SMTP HTML e-mail test

This is an e-mail message to be sent in HTML format

<b>This is HTML message.</b>
<h1>This is headline.</h1>
"""

try:
   smtpObj = smtplib.SMTP('localhost')
   smtpObj.sendmail(sender, receivers, message)
   print "Successfully sent email"
except SMTPException:
   print "Error: unable to send email"
```

Sending Attachments as an E-mail

To send an e-mail with mixed content requires setting the **Content-type** header to **multipart/mixed**. Then, the text and the attachment sections can be specified within **boundaries**.

A boundary is started with two hyphens followed by a unique number, which cannot appear in the message part of the e-mail. A final boundary denoting the e-mail's final section must also end with two hyphens.

The attached files should be encoded with the **pack("m")** function to have base 64 encoding before transmission.

Example

Following is an example, which sends a file /tmp/test.txt as an attachment. Try it once-

```
#!/usr/bin/python3

import smtplib
import base64

filename = "/tmp/test.txt"

# Read a file and encode it into base64 format
fo = open(filename, "rb")
filecontent = fo.read()
```

```
encodedcontent = base64.b64encode(filecontent)  # base64
sender = 'webmaster@tutorialpoint.com'
reciever = 'amrood.admin@gmail.com'

marker = "AUNIQUEMARKER"
body ="""
This is a test email to send an attachement.
"""
# Define the main headers.
part1 = """From: From Person <me@fromdomain.net>
To: To Person <amrood.admin@gmail.com>
Subject: Sending Attachement
MIME-Version: 1.0
Content-Type: multipart/mixed; boundary=%s
--%s
""" % (marker, marker)

# Define the message action
part2 = """Content-Type: text/plain
Content-Transfer-Encoding:8bit

%s
--%s
""" % (body,marker)

# Define the attachment section
part3 = """Content-Type: multipart/mixed; name=\"%s\"
Content-Transfer-Encoding:base64
Content-Disposition: attachment; filename=%s

%s
--%s--
""" %(filename, filename, encodedcontent, marker)
message = part1 + part2 + part3

try:
   smtpObj = smtplib.SMTP('localhost')
   smtpObj.sendmail(sender, reciever, message)
```

```
   print ("Successfully sent email")
except Exception:
   print ("Error: unable to send email")
```