Python: how to send an email with attachments

In this article we will see how to send an email with attachments in Python.

We need to create an SMTP session with the remote server after creating the message to be sent. The session requires the use of SSL to protect the integrity of our session and the login credentials. If the message contains a reference to a file to be attached, this file must be read in binary mode and encoded in Base64 in order to be sent with the body of the email.

The function we are going to implement accepts two dictionaries as parameters: `smtp_config` contains the data for the SMTP connection and `mail_data` the email data (sender, recipient, subject, plain text body, HTML body and any attached file).

```
import smtplib
import ssl
from email import encoders
from email.mime.base import MIMEBase
from email.mime.text import MIMEText
from email.mime.multipart import MIMEMultipart


def send_email(smtp_config, mail_data):
    if not isinstance(smtp_config, dict) or not
isinstance(mail_data, dict):
        return False

    smtp_port = smtp_config.get('port', 465)
    smtp_host = smtp_config.get('host', 'localhost')
```

```
    smtp_user = smtp_config.get('user', '')
    smtp_pass = smtp_config.get('pass', '')

    sender = mail_data.get('from')
    to_address = mail_data.get('to')

    if not sender or not to_address:
        return False

    receiver = [to_address]
    msg = MIMEMultipart('alternative')

    msg['Subject'] = mail_data.get('subject', 'Test
Message')
    msg['From'] = sender
    msg['To'] = to_address

    text_plain = MIMEText(mail_data.get('body', ''),
'plain', 'UTF-8')
    html = MIMEText(mail_data.get('body_html', ''),
'html', 'UTF-8')

    msg.attach(text_plain)
    msg.attach(html)

    attachment_file = mail_data.get('attachment')

    if attachment_file:
        try:
            with open(attachment_file, 'rb') as f:
                part = MIMEBase('application',
'octet-stream')
                part.set_payload(f.read())
            encoders.encode_base64(part)
```

```
            part.add_header('Content-Disposition',
f'attachment; filename= {attachment_file}')
            msg.attach(part)
        except IOError:
            return False

    context = ssl.create_default_context()

    with smtplib.SMTP_SSL(smtp_host, smtp_port,
context=context) as client:

        client.login(smtp_user, smtp_pass)
        client.sendmail(sender, receiver,
msg.as_string())
        return True
```

The function returns a boolean value that will inform us of the outcome of the operation.