SMTP (Simple Mail Transfer Protocol) — это протокол, который обрабатывает отправку и маршрутизацию электронной почты между почтовыми серверами, это протокол связи RFC 5321 для передачи электронной почты. Почтовые серверы используют этот сетевой протокол для отправки почтовых сообщений.

Модуль smtplib использует протокол SMTP и определяет объект сеанса — клиент SMTP, который может использоваться для отправки почты на любой компьютер в Интернете с прослушивателем SMTP (или Extended-SMTP).

Код для создания сообщения:

**import smtplib**

**from email import encoders**

**from email.mime.text import MIMEText**

**from email.mime.multipart import MIMEMultipart**

**from email.mime.base import MIMEBase**

**from bs4 import BeautifulSoup as bs**

**# ваши учетные данные**

**email = "address@gmail.com"**

**password = "password"**

**# электронная почта отправителя**

**FROM = "from address@gmail.com"**

**# адрес электронной почты получателя**

**TO = "to address@gmail.com"**

**# тема письма (тема)**

**subject = "Just a subject"**

**# инициализируем сообщение, которое хотим отправить**

**msg = MIMEMultipart("alternative")**

**# установить адрес электронной почты отправителя**

**msg["From"] = FROM**

**# установить адрес электронной почты получателя**

**msg["To"] = TO**

**# задаем тему**

**msg["Subject"] = subject**

**# установить тело письма как HTML**

**html = """**

**This email is sent using <b>Python</b>!**

**"""**

**# делаем текстовую версию HTML**

**text = bs(html, "html.parser").text**

**# установить тело письма как HTML**

**html = open("mail.html").read()**

**# делаем текстовую версию HTML**

**text = bs(html, "html.parser").text**

**text\_part = MIMEText(text, "plain")**

**html\_part = MIMEText(html, "html")**

**# прикрепить тело письма к почтовому сообщению**

**# сначала прикрепите текстовую версию**

**msg.attach(text\_part)**

**msg.attach(html\_part)**

**print(msg.as\_string())**

В результате выведется текст, где каждая часть разделена фиксированным набором символов, первая часть содержит заголовок письма, а вторая содержит фактическое тело сообщения, которое было прикреплено.

Код для отправки сообщения:

**def send\_mail(email, password, FROM, TO, msg):**

**# инициализировать SMTP-сервер**

**server = smtplib.SMTP("smtp.gmail.com", 587)**

**# подключиться к SMTP-серверу в режиме TLS (безопасный) и отправить EHLO**

**server.starttls()**

**# войти в учетную запись, используя учетные данные**

**server.login(email, password)**

**# отправить электронное письмо**

**server.sendmail(FROM, TO, msg.as\_string())**

**# завершить сеанс SMTP**

**server.quit()**

**# отправить почту**

**send\_mail(email, password, FROM, TO, msg)**

В результате выполнения приведённого выше кода придет электронное письмо, в котором выведется выведется «This email is sent using Python!».

Для того, чтобы использовать встроенный модуль imaplib для составления списка и чтения ваших писем на Python, понадобится помощь протокола IMAP. IMAP — это стандартный протокол Интернета, используемый почтовыми клиентами для получения сообщений электронной почты с почтового сервера. В отличие от протокола POP3, который загружает электронную почту и удаляет ее с сервера, с IMAP сообщение не остается на локальном компьютере, оно остается на сервере.

Код для чтения сообщения:

**import imaplib**

**import email**

**from email.header import decode\_header**

**import webbrowser**

**import os**

**# учетные данные**

**username = "address@gmail.com"**

**password = "password"**

**def clean(text):**

**# чистый текст для создания папки**

**return "".join(c if c.isalnum() else "\_" for c in text)**

**# создаем класс IMAP4 с SSL**

**imap = imaplib.IMAP4\_SSL("imap.gmail.com")**

**# аутентифицируем**

**imap.login(username, password)**

**status, messages = imap.select("INBOX")**

**# количество популярных писем для получения**

**N = 3**

**# общее количество писем**

**messages = int(messages[0])**

**for i in range(messages, messages-N, -1):**

**# получаем сообщение электронной почты по идентификатору**

**res, msg = imap.fetch(str(i), "(RFC822)")**

**for response in msg:**

**if isinstance(response, tuple):**

**# анализируем байтовое электронное письмо в объект сообщения**

**msg = email.message\_from\_bytes(response[1])**

**# расшифроваем тему письма**

**subject, encoding = decode\_header(msg["Subject"])[0]**

**if isinstance(subject, bytes):**

**# если это байты, декодируем в строку**

**subject = subject.decode(encoding)**

**# расшифровываем отправителя электронной почты**

**From, encoding = decode\_header(msg.get("From"))[0]**

**if isinstance(From, bytes):**

**From = From.decode(encoding)**

**print("Subject:", subject)**

**print("From:", From)**

**# если сообщение электронной почты составное**

**if msg.is\_multipart():**

**# перебираем части письма**

**for part in msg.walk():**

**# извлекаем тип содержимого электронной почты**

**content\_type = part.get\_content\_type()**

**content\_disposition = str(part.get("Content-Disposition"))**

**try:**

**# получаем содержимое письма**

**body = part.get\_payload(decode=True).decode()**

**except:**

**pass**

**if content\_type == "text/plain" and "attachment" not in content\_disposition:**

**# печатаем текст/обычные электронные письма и пропускаем вложения**

**print(body)**

**elif "attachment" in content\_disposition:**

**# скачиваем вложение**

**filename = part.get\_filename()**

**if filename:**

**folder\_name = clean(subject)**

**if not os.path.isdir(folder\_name):**

**# создаем папку для этого электронного письма (названную в соответствии с темой)**

**os.mkdir(folder\_name)**

**filepath = os.path.join(folder\_name, filename)**

**# скачиваем вложение и сохраняем его**

**open(filepath, "wb").write(part.get\_payload(decode=True))**

**else:**

**# извлекаем тип содержимого электронной почты**

**content\_type = msg.get\_content\_type()**

**# получаем содержимое письма**

**body = msg.get\_payload(decode=True).decode()**

**if content\_type == "text/plain":**

**# печатаем только текстовые части электронного письма**

**print(body)**

**if content\_type == "text/html":**

**# если это HTML, создаем новый HTML-файл и открываем его в браузере**

**folder\_name = clean(subject)**

**if not os.path.isdir(folder\_name):**

**# создаем папку для этого электронного письма (названную в соответствии с темой)**

**os.mkdir(folder\_name)**

**filename = "index.html"**

**filepath = os.path.join(folder\_name, filename)**

**# пишем файл**

**open(filepath, "w").write(body)**

**# открываем в браузере по умолчанию**

**webbrowser.open(filepath)**

**print("="\*100)**

**# закрываем соединение и выходим из системы**

**imap.close()**

**imap.logout()**

В результате код будет печатать только текстовые/обычные сообщения, он будет создавать папку для каждого электронного письма, в котором он содержит вложение и HTML-версию электронного письма, он также открывает HTML-письмо в браузере по умолчанию для каждого извлеченного электронного письма с HTML-контентом.

imaplib реализует протокол IMAP в Python, IMAP — это стандартный протокол Интернета, используемый почтовыми клиентами для получения и удаления сообщений электронной почты с почтового сервера.

С помощью модуля imaplib в Python можно автоматически удалять письма:

**import imaplib**

**import email**

**from email.header import decode\_header**

**# Измените учетные данные**

**username = "address@gmail.com"**

**password = "password"**

**# создать класс IMAP4 с SSL**

**imap = imaplib.IMAP4\_SSL("imap.gmail.com")**

**# аутентификация**

**imap.login(username, password)**

**# выберите почтовый ящик, в котором надо удалить письма**

**# если вам нужен SPAM, используйте вместо него imap.select("SPAM")**

**imap.select("INBOX")**

**# поиск определенных писем по отправителю**

**status, messages = imap.search(None, 'FROM "googlealerts-noreply@google.com"')**

**# чтобы получить все письма:**

**# status, messages = imap.search(None, "ALL")**

**# для получения писем по теме**

**# status, messages = imap.search(None, 'SUBJECT «Спасибо за подписку на нашу рассылку!»')**

**# для получения писем после определенной даты**

**# status, messages = imap.search(None, ''SINCE "01-JAN-2020"')**

**# для получения писем до определенной даты**

**# status, messages = imap.search (Нет, 'BEFORE "01-JAN-2020"')**

**# преобразовать сообщения в список адресов электронной почты**

**messages = messages[0].split(b' ')**

**for mail in messages:**

**\_, msg = imap.fetch(mail, "(RFC822)")**

**# если у вас длинный список писем, то можно удалить цикл for для повышения производительности**

**# потому что он предназначен только для Subject цэлектронных писем для удаления**

**for response in msg:**

**if isinstance(response, tuple):**

**msg = email.message\_from\_bytes(response[1])**

**# декодирование темы email**

**subject = decode\_header(msg["Subject"])[0][0]**

**if isinstance(subject, bytes):**

**# если это байтовый тип, декодировать в str**

**subject = subject.decode()**

**print("Deleting", subject)**

**# отметить письмо как удаленное**

**imap.store(mail, "+FLAGS", "\\Deleted")**

**# навсегда удалить письма, помеченные как удаленные**

**# из выбранного почтового ящика (в нашем случае INBOX)**

**imap.expunge()**

**# закрыть email**

**imap.close()**

**# выйти из аккаунта**

**imap.logout()**

В результате после запуска кода выведется сообщение об удалении писем.

Таким образом, с помощью выше приведенных кодов можно создавать, читать, удалять и отправлять электронные письма.