

## **EXPERIMENT 3: DEVELOPING AN EMAIL CLIENT USING PYTHON**

### **(TEAM 9)**

#### **AIM:**

Develop an email client using python:

1. Compose parsing email;
2. Compose Multiple Input Multiple Execution (MIME) attachment(s) and their decoding;  
and
3. Decode the email header.

#### **SOFTWARES REQUIRED:**

1. Microsoft Visual Studio Code, Microsoft Corporation
2. Oracle VM VirtualBox 6.1.38, Oracle Corporation
3. Python 3.10.7 (64-bit), Python Software Foundation
4. Termux Mobile Application (Available for Free on Play Store), Fredrik Fornwall
5. Ubuntu 22.04 (64-bit) Operating System

#### **THEORY:**

##### **Q. What is meant by an email client?**

A Mail User Agent is another name for an email client. Actually, sending and receiving emails is done through a programme or application that a person has installed on his or her computer. Emails are written, sent, and viewed using an email client. It is mostly utilised for managing emails, including their creation, transmission, receipt, and deletion.

Every email address has three components, which are required for sending and receiving emails:

1. Username – Anything is acceptable as long as it complies with the email provider's rules.
2. Domain Name (or) Host – This is the mail server that will store the message. It might be @outlook, @gmail, etc.
3. Domain Kind – This information indicates if an email address is for commercial use (.com), educational use (.edu), or any other special type.

##### **Q. What does an email client contain?**

1. Through the use of a user-friendly interface, it conceals the complexity of sending and receiving emails. Any user, regardless of experience level, may send and receive emails with ease by utilising this interface.

2. The majority of email client servers have some sort of backup, so if a message is deleted, it just moves to the trash can and may be recovered from there.
3. Each email client offers a high level of encryption to the messages as they travel over the internet to the server, protecting the communications' confidentiality or privacy.
4. Accessibility, which refers to the ability to read messages from any computer or mobile device from anywhere in the globe via an internet connection, is one of the most crucial characteristics.
5. There are no memory or storage issues because email clients offer more than enough RAM on email servers.
6. As a to-do list for establishing crucial reminders, email clients may also be utilised.

### **Microsoft Outlook, A Well-Known Email Client:**

It is one of the greatest email programmes since it contains a tonne of options that people can use to handle their personal information. It is a component of the Office 365 programme. Both standalone and multi-user applications are possible with this programme. Microsoft Outlook has a number of functions, including:

- Managing Tasks
- Management of Contact
- Organizing Reminders
- Calendar management and taking notes

### **Q. What are some of the applications of an email client?**

1. Internet Communication with Others – By sending and receiving messages, email clients facilitate safe online communication.
2. Expansion and Development of Business – In the current digital era, it is almost difficult to avoid interacting with an email client for any reason. It aids in business growth by offering a digital platform that makes it simple to carry out branding or advertising.
3. Transfer of Various File Types – The usage of email clients makes it very simple to exchange files with distant customers. Different sorts of files and media may be transmitted with ease utilising email clients.
4. Data Backup – Historically, backing up useful files has been a major challenge for everyone, but thanks to email, it is now simple to save useful material that can be retrieved at any time and from any location.
5. Record Keeping – It is also used to retain written records of client transactions, ensuring that everyone's assertions are true and that no one can refute them.

### **Q. What are some of the issues with an email client?**

1. System Updates – Webmail provides updates every few weeks, however, email clients may not release any changes for many years.
2. Access Email on Several Computers – IMAP is a popular email protocol that certain email providers employ and is great for synchronising between devices. Other POP-enabled

email clients don't have reliable synchronisation. Users will be left without many options and in a bind when trying to access their email from other machines.

3. Data Backup – Because email clients save all sent and received emails on the computer, if there is a software or hardware fault and the emails have not been backed up, then there is a chance that all emails will be lost.

### Key Difference Between Email and Web Client:

They both have the same function: sending emails, downloading files, using a calendar, and storing contact information. The methods for accessing them and retrieving deleted emails do, however, differ significantly. Webmail can only be accessed using web browsers, whereas email clients may be accessed through desktop programmes.



Figure 1 – Demonstration of Email Client Process

### SOURCE CODE:

#### Email.py

```
import smtplib # Defines an SMTP client session object that can be used to
send mail to any internet machine with an SMTP or ESMTP listener daemon.

"""
The email package provides some convenient encoders in its encoders module.
These encoders are actually used by the MIMEAudio and MIMEImage class
constructors to provide default encodings.
"""

from email import import encoders

from email.mime.text import MIMEText # A subclass of MIMENonMultipart, the
MIMEText class is used to create MIME objects of major type text.
from email.mime.base import MIMEBase # This is the base class for all the
MIME-specific subclasses of Message.
from email.mime.multipart import MIMEMultipart # A subclass of MIMEBase, this
is an intermediate base class for MIME messages that are multipart.

server = smtplib.SMTP('smtp.gmail.com', 587) # Forms an email client with SMTP
listener module
# server.connect("smtp.gmail.com", 587)
```

```

# server.ehlo()
# with open('password.txt' , 'r') as f:

server.starttls()
server.login('group9cn@gmail.com', 'fbyj klbk yfsf zapk') # Log in with mail
credentials
msg = MIMEMultipart() # Secure Multipurpose Internet Mail Extensions (S/MIME)
is used to send attachments

msg['From'] = 'Santosh' # Sender Name
msg['To'] = 'group9cn@gmail.com' # Receiver Email Address
msg['Subject'] = 'Test File - Experiment 3' # Subject of the Mail

# with open('attachment.txt', 'r') as f:
# message = f.read()
message = 'Hi! This is Santosh (53) here. I have sent an email sent from
Python.'
msg.attach(MIMEText(message , 'plain')) # Text Message Attachment

filename = 'Email.py' # File Name of Attachment
attachment = open(filename, 'rb') # Opens a file, and returns it as a file
object
p = MIMEBase('application', 'octet-stream') # File Attachment
p.set_payload(attachment.read())

encoders.encode_base64(p) # Encodes the payload into base64 form and sets the
Content-Transfer-Encoding header to base64.
p.add_header('content-Disposition', f'attachment; filename = {filename}') #
Header name is the same as file name
msg.attach(p)

text = msg.as_string() # Allows str(msg) to produce a string containing the
serialized message in a readable format.
server.sendmail('group9cn@gmail.com' , 'group9cn@gmail.com', text) # Send
mail, from sender to receiver with text message
print("\nThe mail along with the attachment(s) has been sent to your
inbox.\n")

```

#### Email\_Decode.py

```

import imaplib # This module defines three classes, IMAP4, IMAP4_SSL and
IMAP4_stream, which encapsulate a connection to an IMAP4 server and implement
a large subset of the IMAP4rev1 client protocol as defined in RFC 2060.

import email # Library for managing email messages
imap_server = 'imap.gmail.com' # IMAP server with the IMAP listener module
email_address = 'group9cn@gmail.com'
password = 'fbyj klbk yfsf zapk'

```

```

imap = imaplib.IMAP4_SSL(imap_server) # This is a subclass derived from IMAP4
that connects over an SSL encrypted socket
imap.login(email_address, password) # Log in with mail credentials
imap.select("Inbox") # Item selection for decoding
_, msgnum = imap.search(None, "ALL") # Return the count of mails in 'Inbox'
print("\nCount of Mails in Inbox: " + str(msgnum) + "\n")

# Print Decoded Details:-
for msgnum in msgnum[0].split(): # Splits a string into a list
    _, data = imap.fetch(msgnum, ("RFC822")) # Fetch (parts of) messages

    message = email.message_from_bytes(data[0][1]) # Return a message object
    structure from a bytes-like object.
    print(f"Message Index Number: {msgnum}")
    print(f"From : {message.get('From')}")
    print(f"To : {message.get('To')}")

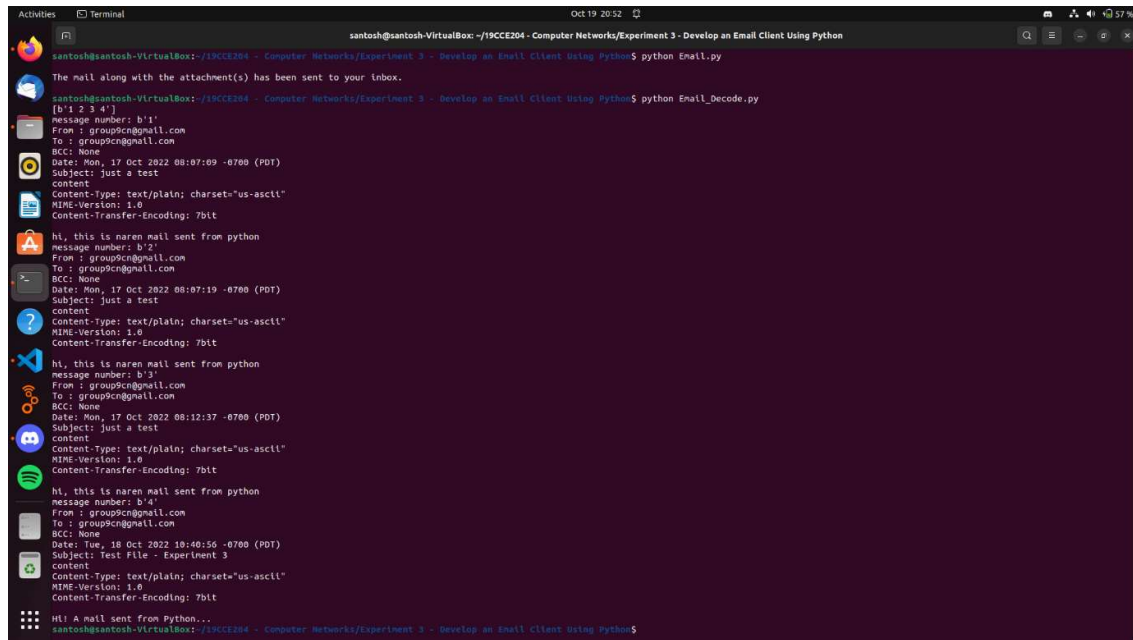
    print(f"BCC: {message.get('BCC')}")
    print(f>Date: {message.get('Date')}")
    print(f"Subject: {message.get('Subject')}")
    print("Content: ")

    # The walk() method is an all-purpose generator which can be used to
    iterate over all the parts and subparts of a message object tree, in depth-
    first traversal order.
    for part in message.walk():
        if part.get_content_type() == "text/plain": # Return the message's
        content type, coerced to lower case of the form maintype/subtype.
            print(part.as_string()) # Allows str(msg) to produce a string
            containing the serialized message in a readable format.

    print("\n")

```

## SCREENSHOTS OF INPUTS & OUTPUTS:

A terminal window titled 'santosh@santosh-VirtualBox: ~/19CCE204 - Computer Networks/Experiment 3 - Develop an Email Client Using Python'. The terminal shows the execution of 'python Email.py' and 'python Email\_Decode.py'. The output displays four email messages sent from 'group9cn@gmail.com' to 'santosh@group9cn@gmail.com'. The messages are: 1) 'The mail along with the attachment(s) has been sent to your inbox.', 2) 'hi, this is naren mail sent from python message number: b'2'', 3) 'hi, this is naren mail sent from python message number: b'3'', and 4) 'hi, this is naren mail sent from python message number: b'4''. Each message header includes 'From: group9cn@gmail.com', 'To: santosh@group9cn@gmail.com', 'BCC: None', 'Date: Mon, 17 Oct 2022 08:07:09 -0700 (PDT)', 'Subject: Just a test', 'Content-Type: text/plain; charset="us-ascii"', 'MIME-Version: 1.0', and 'Content-Transfer-Encoding: 7bit'. The terminal also shows the output of 'python Email\_Decode.py' which prints the decoded content of the emails.

```
santosh@santosh-VirtualBox: ~/19CCE204 - Computer Networks/Experiment 3 - Develop an Email Client Using Python$ python Email.py
The mail along with the attachment(s) has been sent to your inbox.

santosh@santosh-VirtualBox: ~/19CCE204 - Computer Networks/Experiment 3 - Develop an Email Client Using Python$ python Email_Decode.py
[0 1 2 3 4]
Message number: b'1'
From: group9cn@gmail.com
To: santosh@group9cn@gmail.com
BCC: None
Date: Mon, 17 Oct 2022 08:07:09 -0700 (PDT)
Subject: Just a test
Content-Type: text/plain; charset="us-ascii"
MIME-Version: 1.0
Content-Transfer-Encoding: 7bit
Content:
hi, this is naren mail sent from python
message number: b'2'
From: group9cn@gmail.com
To: santosh@group9cn@gmail.com
BCC: None
Date: Mon, 17 Oct 2022 08:07:19 -0700 (PDT)
Subject: Just a test
Content-Type: text/plain; charset="us-ascii"
MIME-Version: 1.0
Content-Transfer-Encoding: 7bit
Content:
hi, this is naren mail sent from python
message number: b'3'
From: group9cn@gmail.com
To: santosh@group9cn@gmail.com
BCC: None
Date: Mon, 17 Oct 2022 08:12:37 -0700 (PDT)
Subject: Just a test
Content-Type: text/plain; charset="us-ascii"
MIME-Version: 1.0
Content-Transfer-Encoding: 7bit
Content:
hi, this is naren mail sent from python
message number: b'4'
From: group9cn@gmail.com
To: santosh@group9cn@gmail.com
BCC: None
Date: Tue, 18 Oct 2022 10:40:56 -0700 (PDT)
Subject: Test File - Experiment 3
Content-Type: text/plain; charset="us-ascii"
MIME-Version: 1.0
Content-Transfer-Encoding: 7bit
Content:
Hi! A mail sent from Python...

santosh@santosh-VirtualBox: ~/19CCE204 - Computer Networks/Experiment 3 - Develop an Email Client Using Python$
```

Figure 2 – Program Output Via Terminal

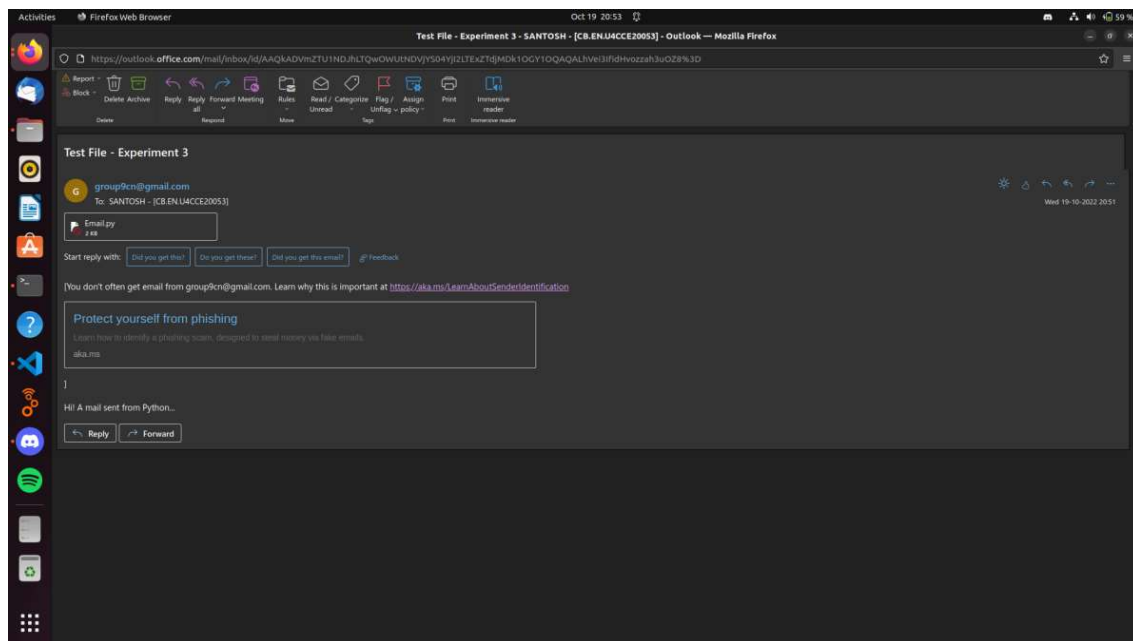


Figure 3 – Email Along with Attachment

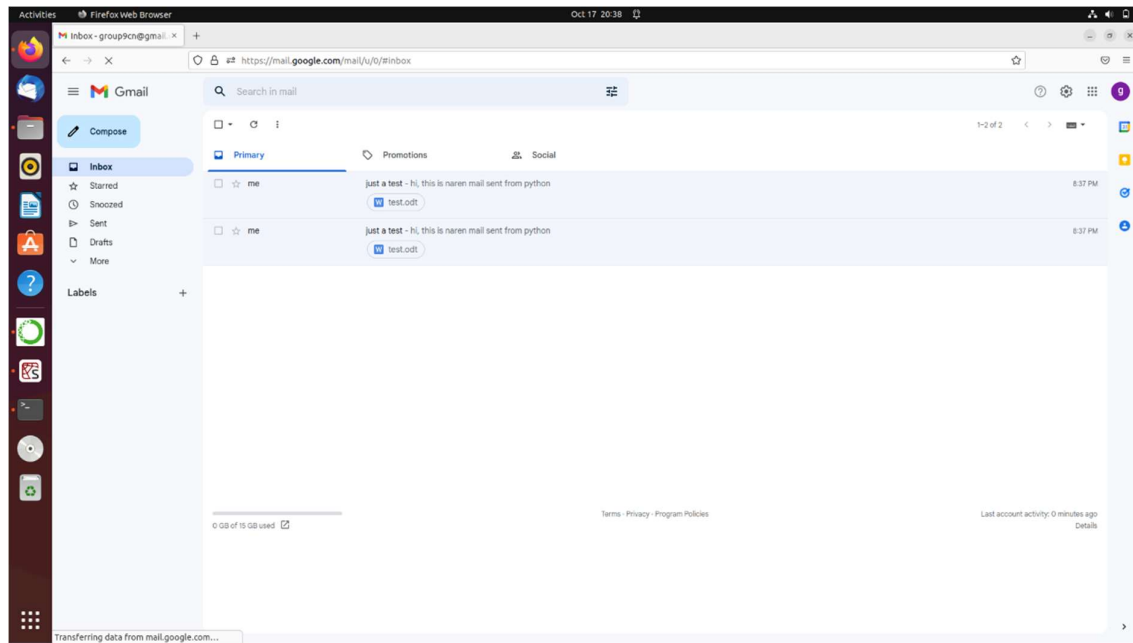


Figure 4 – Mails in Inbox

## **CONCLUSION:**

Thus, developed an email client using Python that composes parsing email and MIME attachments. Decoding of the same along with the email header has been performed too. All the simulation results were verified successfully.

## **REFERENCES:**

1. Foundations of Python 3 Network Programming, John Goerzen, Brandon Rhodes, Second Edition, Apress Publisher, ISBN-10: 1430258543, ISBN-13: 978-1430258544
2. Foundations of Python Network Programming by Beaulne, Alexandre Goerzen, John Membrey, Peter Rhodes, Brandon 2014.

## **CONTRIBUTION:**

1. Narendran S [CB.EN.U4CCE20036] – Compose parsing email + Code Debugging
2. Narun T [CB.EN.U4CCE20037] – Compose Multiple Input Multiple Execution (MIME) attachments + Code Reviewing
3. Pabbathi Greeshma [CB.EN.U4CCE20040] – Decode attachments and email header + Code Testing
4. Santosh [CB.EN.U4CCE20053] – Topic Research + Documentation