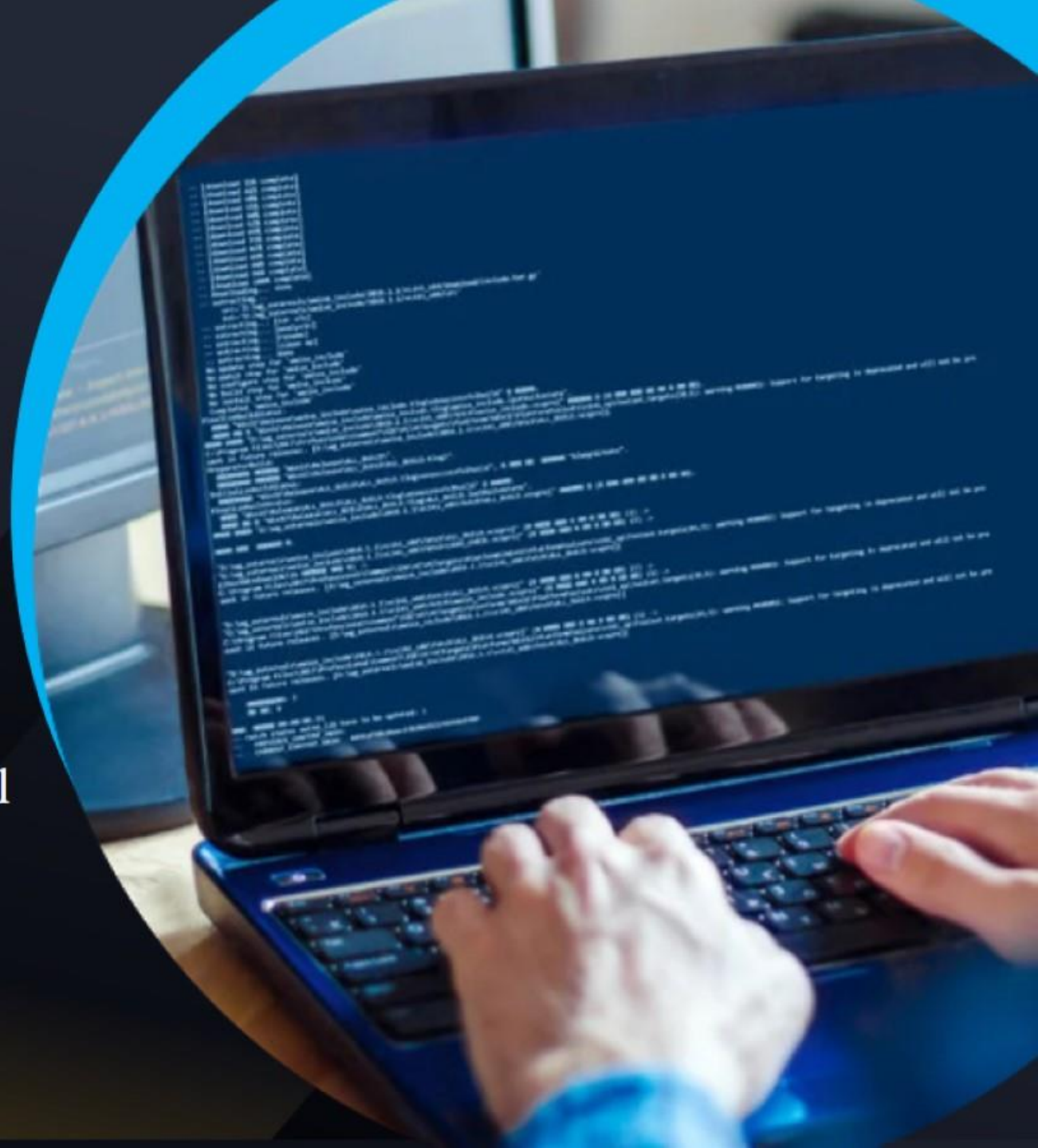


Network Programming in Python

Prepared by Dr. Hamada I. AbdulWakel

Lecture 8



The background features a series of vertical panels with rounded ends. The first panel on the left shows a person's hands typing on a laptop. The subsequent panels are semi-transparent and contain various white line-art icons: a graduation cap, a target with an arrow, an open book, a magnifying glass, a play button, and a group of people. The title 'Email Composition and Decoding' is written in a large, bold, blue font across the middle of these panels.

Email Composition and Decoding

Lecture 8

Prepared by
Dr. Hamada I. AbdulWakel

www.midocean.edu.km

Lecture

08

RECAP: LECTURE 7

- Domain Name System (DNS).
- DNS module in Python.

Lecture

08

LECTURE MAIN POINTS

- Email Messages.
- E-mail Module in Python.
- The Multipurpose Internet Mail Extensions.

Lecture

08

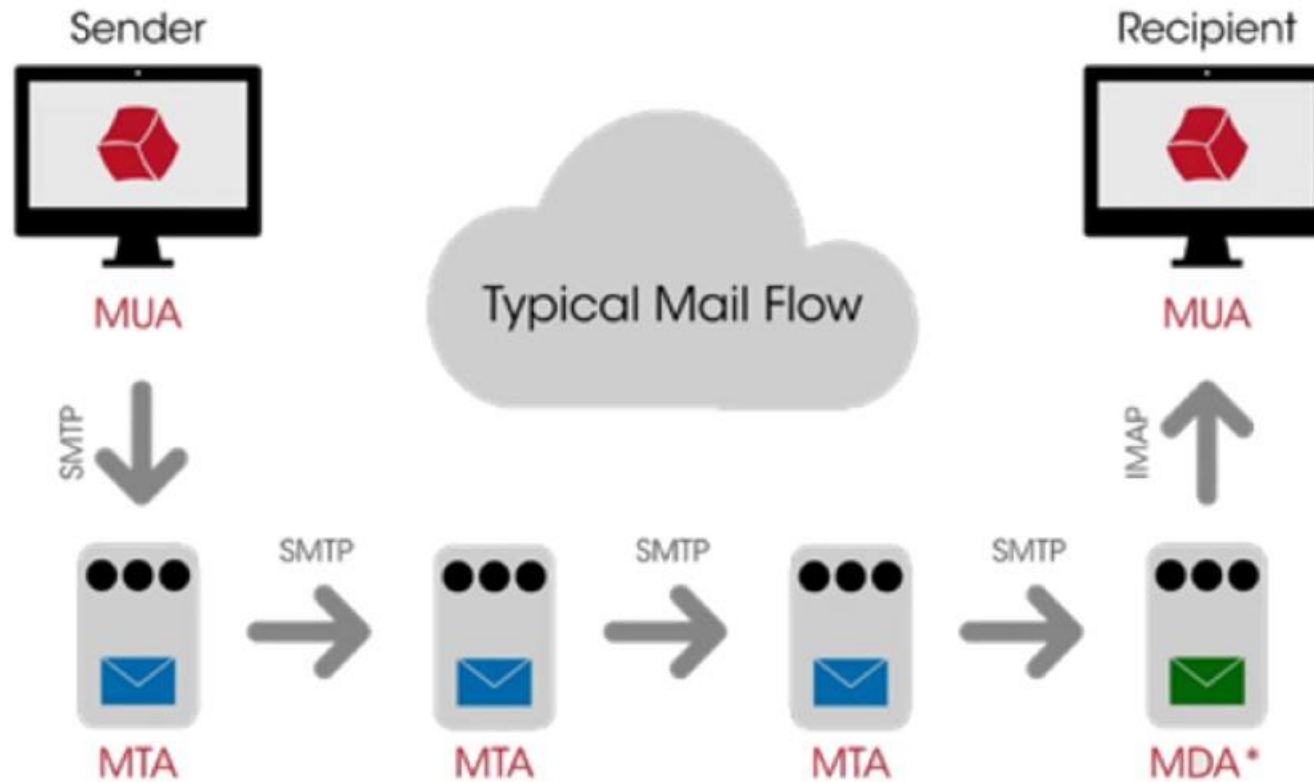
E-MAIL DEFINITION

- Sending and receiving of information through internet is known as an E-mail.
- Today, email is one of the most popular methods of digital communication.
- Sharing an email is done through the use of email address.
- Email is a critical means of communication in today's world, but it can be time-consuming and tedious to send out emails manually.

Lecture

08

E-MAIL BASIC FLOW



- **A Mail User Agent (MUA):** the client application running on a computer that is used to send and receive mail.
 - Examples: Apple Mail and Microsoft Outlook.
- **A Mail Transfer Agent (MTA):** accepts messages from a sender and routes them along to their destinations.
 - Examples: Sendmail, Postfix, and Microsoft Exchange.
- **A Mail Delivery Agent (MDA):** receives email from an MTA for delivery to a mailbox.

Lecture

08

E-MAIL PARTS

- **Email headers:** consists of the sender, receiver, subject, and timestamp. Headers are part of every email and is has many usages.
- **Email message:**
 - The message or body of the email is where a lot of the information about the email exists.
 - Using this information, we can extract key details about the purpose of the message.

- **Email attachment:**
 - Files that are part of the email body. Since data is sent in various forms, parsing attachments sent through email is important to create a powerful user experience.
- **Email HTML:**
 - Emails generally come in plain text and HTML.
 - Parsing HTML emails is a powerful way to get extra information to display in a different format and from various sources such as newsletters and transactional emails.

- **Email specific data fields:**
 - We can parse emails using different fields available to use. For example, if you want to find more important emails, we can consider using the starred field with communication providers like Gmail.

Lecture

08

E-MAIL HEADER INFORMATION

**Hamada Abdul-wakel** <hamada@midocean

to

from: **Hamada Abdul-wakel** <hamada@midocean.
to: @gmail.com>
date: Nov 22, 2023, 6:45 PM
subject: Re: الاختبار الثاني
mailed-by: midocean

E-MAIL HEADER INFORMATION

Cont'd

- **An Email Header:** is metadata that accompanies every email and contains detailed information, such as the sender, receiver, route, timestamp, and more.
- Mailbox and email service providers use email headers to authenticate the email senders and properly allocate an email in the inbox.
- Each field in the header has a corresponding name, followed by a separator character (:) and a value identifier. The main fields located in the email message header are the **from, to, subject, and date**.

E-MAIL HEADER INFORMATION

Cont'd

- **Message-ID:** is a unique identifier, generated automatically to prevent multiple deliveries. It includes various (around 50) letters and numbers.

```
Message-ID: <CAPMaAR-phjcd-Bke+omb6fCcwLQ4y-20b8G+3P9oRZQm+bcn0g@mail.gmail.com>  
Subject: Re: الاختبار الثاني  
From: Hamada Abdul-wakel <hamada@midocean  
To: <l .com>  
Content-Type: multipart/related; boundary="00000000000000a2779060ac07426"
```

Lecture

08

E-MAIL AUTHENTICATION

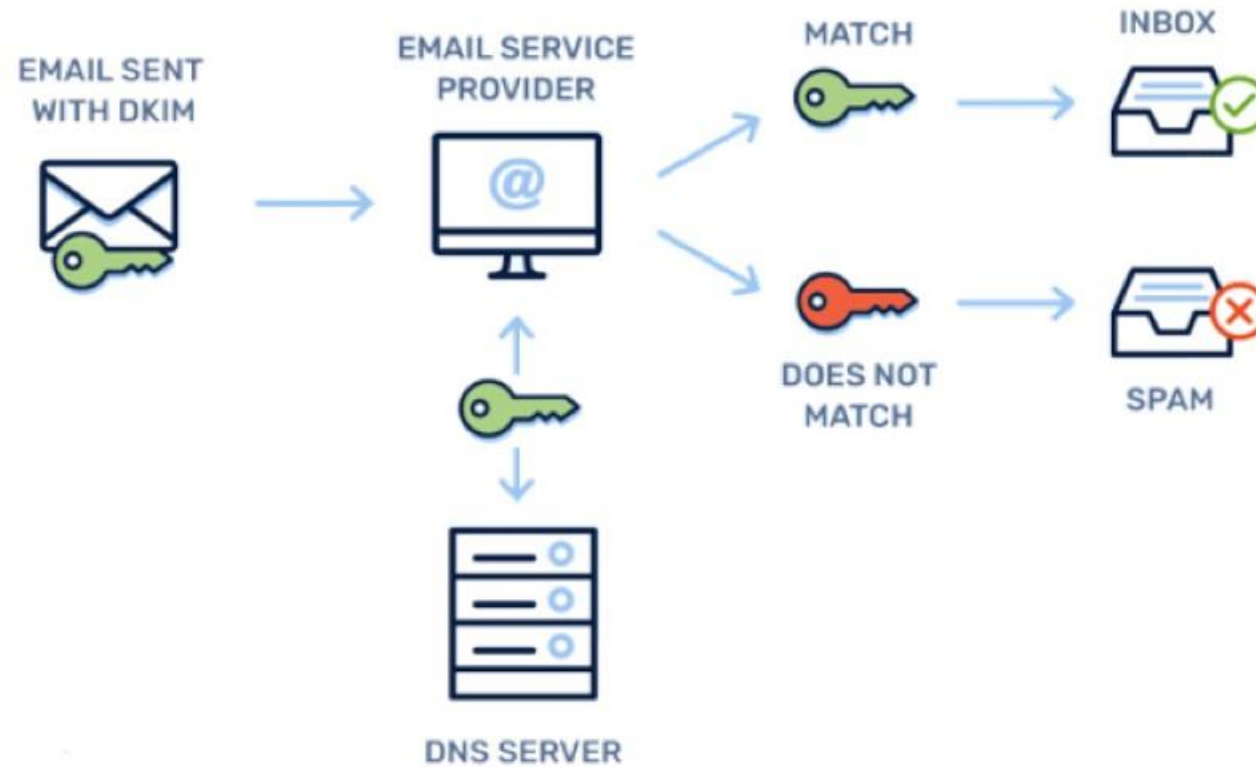
- **Email authentication** helps keep all of us safe from spam and email attacks. There are multiple email authentication protocols. Each protocol protects against a specific type of email forging.
 - **Sender policy framework (SPF)** authentication keeps malicious senders from spoofing emails and IP addresses. It used as a spam detection mechanism.
 - **Domain Keys Identified Mail (DKIM)** authentication protects against emails that have been altered in transit by a bad actor between the sender and the recipient. The email is signed with a digital signature, which can be verified by checking the sender's public key in the DNS records of the sender's domain.

Lecture

08

E-MAIL AUTHENTICATION

Cont'd



Lecture

08

E-MAIL MODULE

- The email module is an email message management library.
`import email`
- It has three major components:
 - The central component is an “object model” that represents email messages.
 - The parser component takes the serialized version of an email message (a stream of bytes) and converts it into a `EmailMessage` object.
 - The generator component takes a `EmailMessage` object and turns it back into a serialized byte stream.

- It used to create the EmailMessage object and add or remove email subcomponents. We can not send an email message using the email module alone.
- The EmailMessage dictionary-like interface is indexed by the header names. The values of the dictionary are strings with some extra methods.
- The email module also provides support for parsing e-mail messages. Reconstructing the EmailMessage object.

E-MAIL MODULE METHODS

- **email.message** submodule used to create instances of the EmailMessage class.
 - **EmailMessage()**: create a message object.
 - **set_payload()**: set the body of a message object.
 - **as_string()**: returns the message object as string.
 - **is_multipart()**: check if the message object includes a list of sub-EmailMessage objects.
 - **keys()**: return a list of all the message's header field names.
 - **values()**: return a list of all the message's header values.

- **email.utils** submodule methods:
 - **make_msgid()**: returns a string represents ant Message-ID header.
 - **formatdate()**: returns a date time as string.
 - The localtime argument used to return time related to the local timezone.
 - **parsedate_tz()**: returns either None or a 10-tuple datetime.
 - **mktime_tz()**: create a Universal Time Coordinated (UTC) timestamp from a 10-tuple datetime.
 - Unix time: the number of seconds that have elapsed since 00:00:00 UTC on 1 January 1970.

Lecture

08

CREATING AN E-MAIL EXAMPLE

• Code:

```
import email.utils, email.message
text = " Hello, This is a test message. I hope you enjoy it! "
msg = email.message.EmailMessage()
msg['To'] = 'recipient@example.com'
msg['From'] = 'sender@example.com'
msg['Subject'] = 'Test Message'
msg['Date'] = email.utils.formatdate(localtime = 1)
msg['Message-ID'] = email.utils.make_msgid()
msg.set_payload(text);  print(msg.as_string())
```

Lecture

08

PARSING AN E-MAIL EXAMPLE

• Code:

```
import email
popular_headers = ('From', 'To', 'Subject', 'Date')
msg = email.message_from_file( open('/path/to/message.txt') )
headers = sorted(msg.keys())
for header in headers:
    if header not in popular_headers:
        print("Header info. not listed---", header + ':', msg[header] )
    else:
        print("Header info. listed---", header + ':', msg[header] )
```

Lecture

08

PARSING AN E-MAIL EXAMPLE

Cont'd

- Code:

```
if msg.is_multipart():  
    print("This program cannot handle MIME multipart messages.")  
else:  
    print(msg.get_payload())
```

Lecture

08

PARSING DATA IN AN E-MAIL

• Code:

```
from email.utils import parsedate_tz, mktime_tz
```

```
timetuple1 = parsedate_tz('Wed, 18 March 2026 09:00:00 +1000')  
print(timetuple1)
```

```
timestamp1 = mktime_tz(timetuple1)  
print(timestamp1)
```

Lecture

08

THE MULTIPURPOSE INTERNET MAIL EXTENSIONS

- **MIME** is a set of rules for encoding data, rather than deal with simple plain text only inside e-mails.
- **MIME** message can include several different parts (e.g., a plain-text message, an image file attachment, a PDF attachment) into the message body.
- **MIME** operates by adding headers to an e-mail message and then giving it content that looks like plain text to the machine.

Lecture

08

COMMON MIME TYPES

- There are two primary MIME types:
 - **text/plain:** the default value for textual files.
 - A textual file should be human-readable and must not contain binary data.
 - **application/octet-stream:** the default value for all other cases.
 - An unknown file type should use this type.
 - Browsers pay a particular care when manipulating these files, to protect users from software vulnerabilities and possible dangerous behavior.

Lecture

08

MIME TYPES EXAMPLE

• Code:

```
import mimetypes
print(mimetypes.guess_type('test.py'))
mtype = mimetypes.guess_type('test.py')[0]
print(mtype)
print(mimetypes.guess_all_extensions('image/jpeg'))
print(mimetypes.types_map)
```

Lecture

08

CREATING A MIME EMAIL

- To compose a message with attachments, you will generally follow these steps:
 1. Create a **MIMEMultipart** object and set its message headers.
 2. Create a **MIMEText** object with the message body text and attach it to the MIMEMultipart object.
 3. Create appropriate **MIME** objects for each attachment and attach them to the MIMEMultipart object.
 4. Finally, call `as_string()` on the MIMEMultipart object to write out the resulting message.

Lecture

08

MIME TYPES EXAMPLE

- Code:

```
import email.mime.application, email.mime.text, email.mime.multipart, email.utils, os

def attachment(email_message, filename):
    with open(filename, 'rb') as f:
        file_attachment = email.mime.application.MIMEApplication(f.read())
        file_attachment.add_header("Content-Disposition", f"attachments;
filename={filename}")
    email_message.attach(file_attachment)
```

MIME TYPES EXAMPLE

Cont'd

• Code:

```
message = "Hello, This is a test message. I hope you enjoy it!"  
msg = email.mime.multipart.MIMEMultipart()  
msg['To'] = 'recipient@example.com'  
msg['From'] = 'sender@example.com'  
msg['Subject'] = 'Test Message'  
msg['Message-ID'] = email.utils.make_msgid()  
body = email.mime.text.MIMEText(message, 'plain')  
msg.attach(body)  
attachment(msg, 'Document2.pdf'); print(msg.as_string())
```

Lecture

08

QUESTIONS FOR ONLINE DISCUSSION

- Write a program to build an email with attachment. The attached file should be a Python code file.



Thank You

