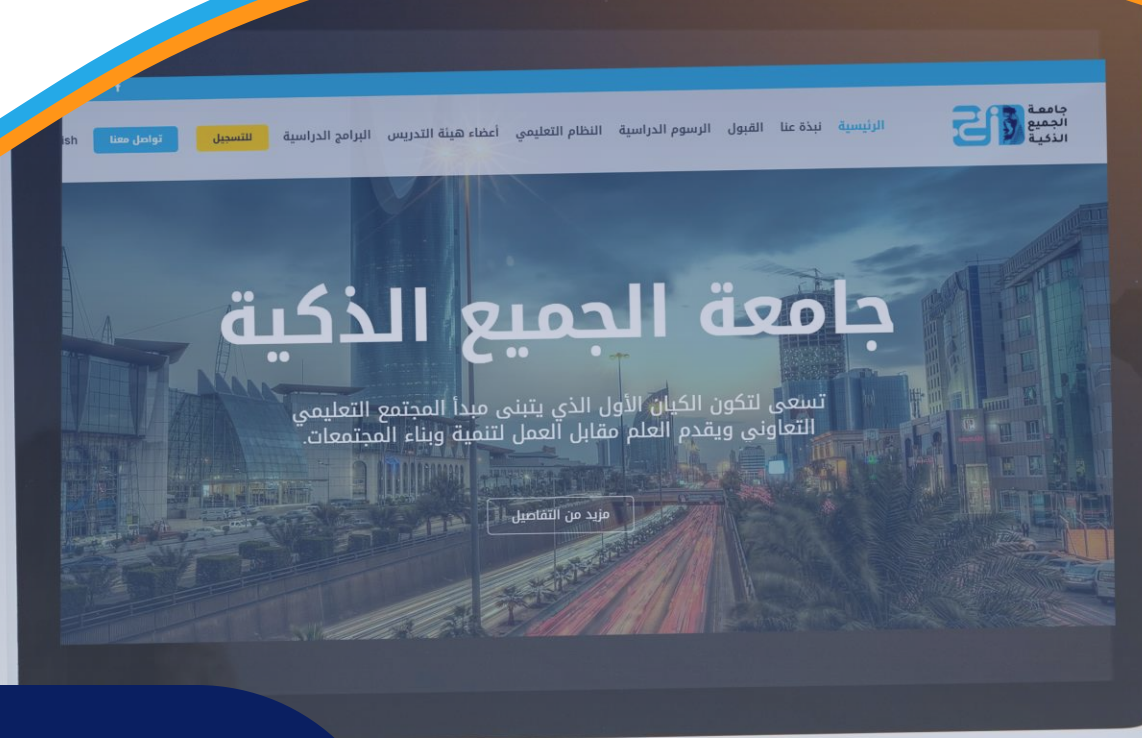




جامعة الجميع الذكية
EVERYONE'S SMART UNIVERSITY

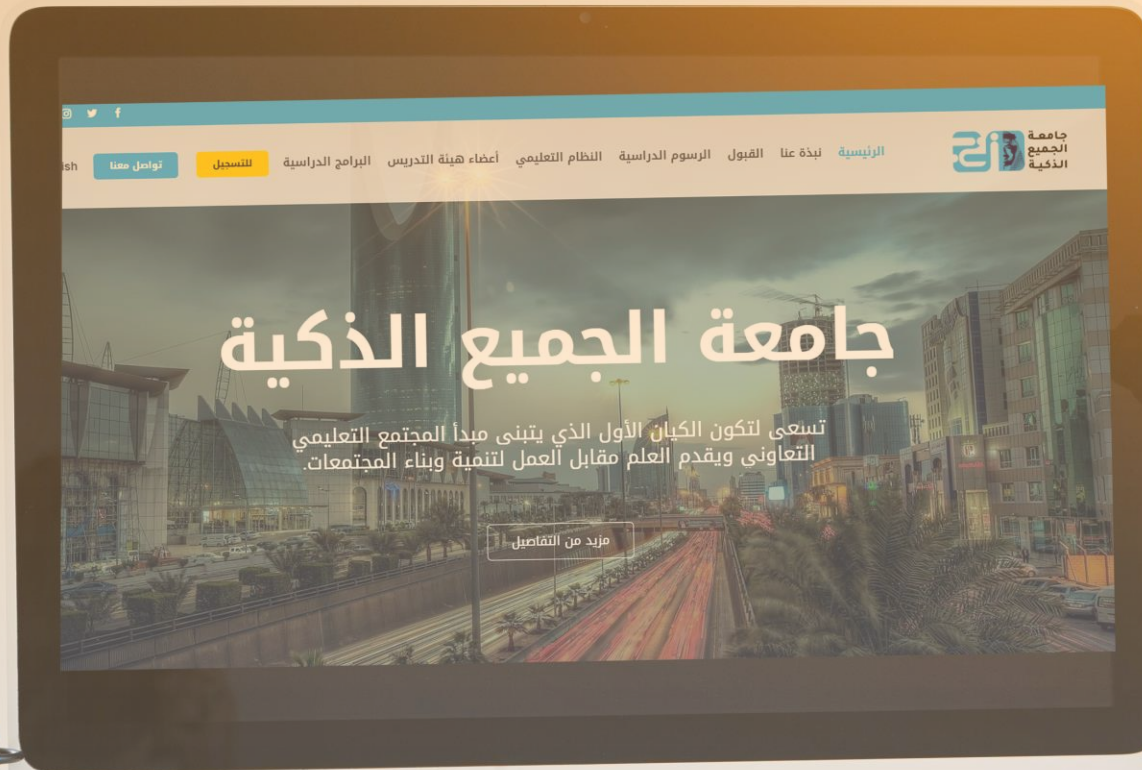


Prepared by
Dr. Hamada I. AbdulWakel
Lecture 8

CYB615
Network Programming



جامعة الجميع الذكية
EVERYONE'S SMART UNIVERSITY



Module8: Email Composition and Decoding

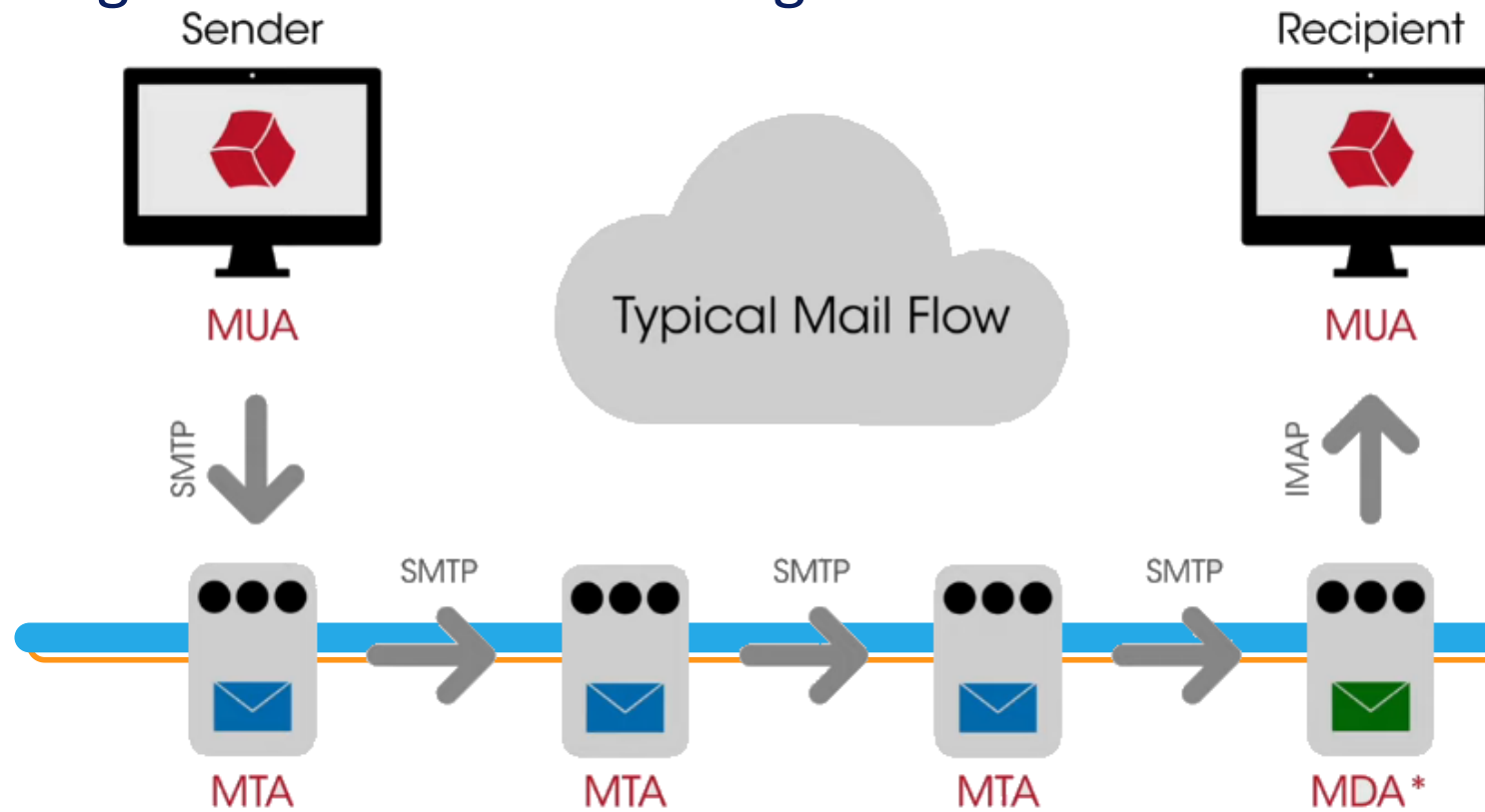
- Email Messages.
- E-mail Module in Python.
- The Multipurpose Internet Mail Extensions.



Lecture 8

E-mail Definition & Its Basic Flow

- Sending and receiving of information through internet is known as an E-mail. Sharing an email is done through the use of email address.



- A Mail User Agent (**MUA**): client app. 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



- Sender Policy Framework (**SPF**): defines a mechanism by which an organization can specify server(s) that are allowed to send email on behalf of that domain.
 - If an email fails an SPF check, it can be an easy mechanism we can use to detect spam.
- DomainKeys Identified Mail (**DKIM**): provides a cryptographic method of verifying a received email actually originated from the sending domain



- The Simple Mail Transfer Protocol (**SMTP**): a widely used **TCP** protocol for email sending.
 - Mainly used by the clients to send emails to the servers or for the email communications between servers.
- The Internet Message Access Protocol (**IMAP**): a protocol for receiving emails from a server.
 - Allows access to emails from multiple locations simultaneously, it keeps the email on the server after being delivered.



Lecture 8

E-mail Header Information

Folder shared with you: "CYB615 حمادة عبدالوكيل" Inb



Ramy Omar (via Google Drive) <drive-shares-dm-noreply@google.com>

to me ▾

from: **Ramy Omar (via Google Drive)** <drive-shares-dm-noreply@google.com>
reply-to: Ramy Omar <ramy@esu.ac.ae>
to: hamada@esu.ac.ae
date: Oct 6, 2022, 1:17 PM
subject: Folder shared with you: "CYB615 حمادة عبدالوكيل"
mailed-by: doclist.bounces.google.com
signed-by: google.com
security: Standard encryption (TLS) [Learn more](#)
 : Important according to Google magic.



— Summary

Subject test 4
Message Id <PNZPR01MB513628451DA5E14390761514A33B9@PNZPR01M85136.INDPRD01.PROD.OUTLOOK.COM>
Creation time Fri, 4 Jun 2021 14:02:27 +0000 (Delivered after 0 seconds)
From User C <userc@365conceptsLabs.onmicrosoft.com>
To Amit Kumar <Admin@365conceptsLabs.onmicrosoft.com>



- The email module is an email message management library.

`import email`

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



- Using the EmailMessage object, we can add or remove email subcomponents.
- We **can not** send an email message using the email module alone.
- The email module also provides support for parsing e-mail messages. Re-constructing the EmailMessage object.
 - Any e-mail has:
 - The standard format.
 - Headers look like the dictionary key-values of the EmailMessage object.



- The **email** method:
 - `EmailMessage()`: create a message object.
- The **email.message** methods:
 - `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.
 - `message_from_file()`: read and parse the content of a text file to a EmailMessage object.
 - `keys()`: return a list of all the message's header field names.



- The **email.utils** methods:
 - **make_msgid()**: returns a string appropriate for a 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 8

Creating Email Headers Example

```
import email.utils
from email.message import EmailMessage
text = " Hello, This is a test message. I hope you enjoy it! "
msg = 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 8

Parsing Email Headers Example

```
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 + ':', msg[header] )
for header in headers:
    if header in popular_headers: print(header + ':', msg[header] )
if msg.is_multipart():
    print("This program cannot handle MIME multipart messages.")
else: print(msg.get_payload())
```



Lecture 8

Parsing Email Date Header Example

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

```
timetuple1 = parsedate_tz('Sun, 27 May 2007 11:34:43 +1000')  
print(timetuple1)
```

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



- **MIME** is a set of rules for encoding data, rather than simple plain text, 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.



- 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.



```
import mimetypes, os

print(os.getcwd())
print(os.listdir())
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)
```



- 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 8

Parsing Email Date Header Example

```
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)
```

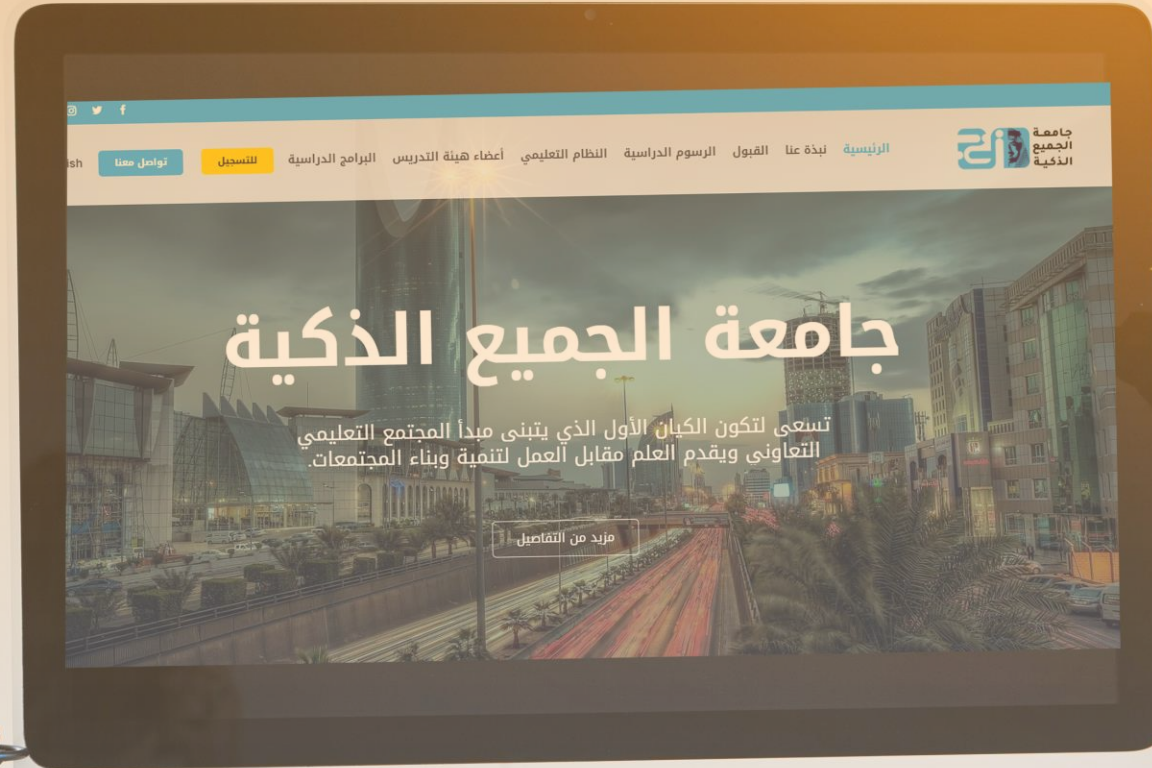


Lecture 8

Parsing Email Date Header Example

```
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['Date'] = email.utils.formatdate(localtime = 1)  
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())
```





جامعة الجميع الذكية
EVERYONE'S SMART UNIVERSITY

THANK YOU