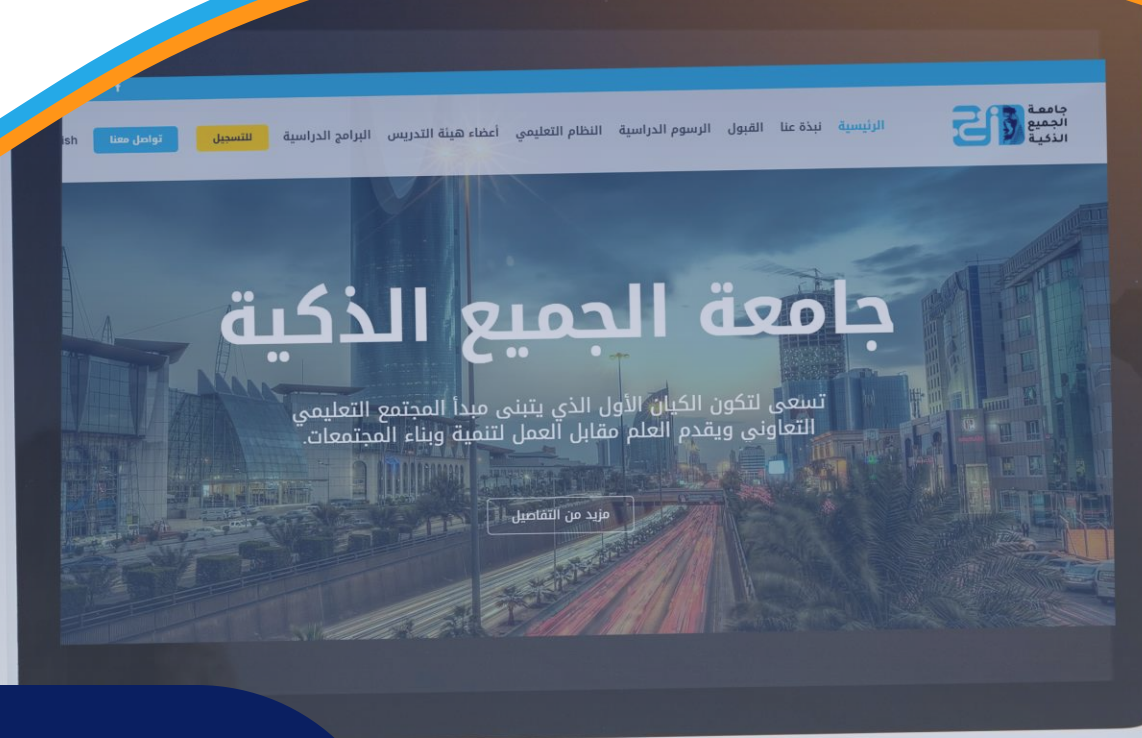




جامعة الجميع الذكية
EVERYONE'S SMART UNIVERSITY

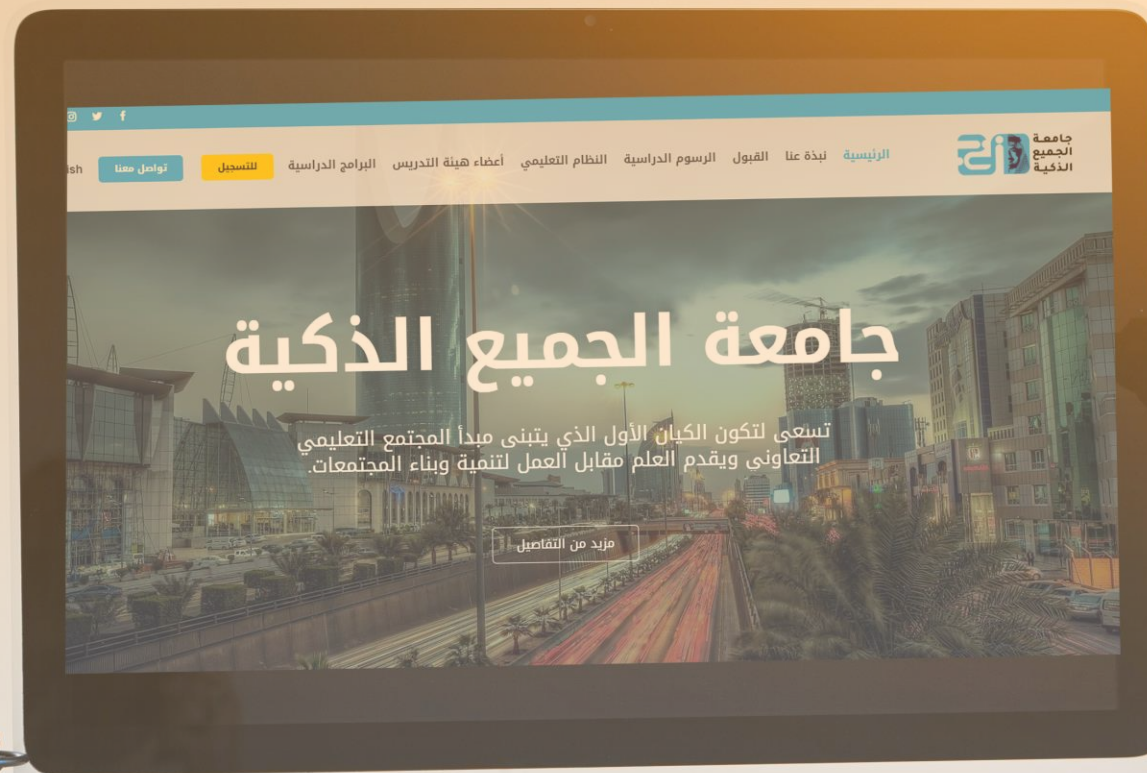


Prepared by
Dr. Hamada I. AbdulWakel
Lecture 9

CYB615
Network Programming



جامعة الجميع الذكية
EVERYONE'S SMART UNIVERSITY



Module9: Handling E-mails

- Email Protocols.
- Sending Emails.
- Securing Emails.
- Retrieving Emails.



- **Simple Mail Transfer Protocol (SMTP):** It can only be used by the MTA for sending e-mails from one host to another.
- **Post Office Protocol 3 (POP3):** It provides a simple and standardized way for the users to gain access to the mailboxes and then download the messages.
- **Internet Message Access Protocol (IMAP):** It provides a simple and standardized way for accessing your e-mail. It requires only a small data transfer; this scheme works well even over a slow connection.



- Python has three modules:
 - `smtplib` for SMTP protocol.
 - `poplib` for POP3 protocol.
 - `imaplib` for IMAP protocol.
- Each module has options for transmitting the information securely by using the Transport Layer Security (TLS) protocol.
- Each protocol also uses some form of authentication for ensuring the confidentiality of the data.



- smtplib has these methods:
 - `SMTP()`: returns an object of SMTP class.
 - `ehlo()`: identifies the server. An extended hello message will be sent by the client to the server.
 - `starttls()`: starts TLS mode.
 - `login()`: takes username and password arguments to login to SMTP server.
 - `sendmail()`: performs an entire mail transaction.
 - `quit()`: terminates SMTP session.



Lecture 9

Checking Email Client Running

```
import smtplib
```

```
session = smtplib.SMTP('localhost', 1025)
```

```
subject = 'Test'
```

```
body = "A test message to check the email sending using STMP"
```

```
msg = f"Subject: {subject} \n\n{body}"
```

```
session.sendemail("hamada@esu.ae.ac", "hamada_mbsoft@yahoo.com", msg)
```



Lecture 9

Checking Email Client Running

Cont'd

```
drhamada@drhamada-Latitude-E6520: ~  
File Edit View Search Terminal Help  
drhamada@drhamada-Latitude-E6520:~$ python3 -m smtpd -c DebuggingServer -n local  
host:1025  
/usr/lib/python3.10/smtpd.py:105: DeprecationWarning: The asyncore module is dep  
recated and will be removed in Python 3.12. The recommended replacement is async  
io  
    import asyncore  
/usr/lib/python3.10/smtpd.py:106: DeprecationWarning: The asynchat module is dep  
recated and will be removed in Python 3.12. The recommended replacement is async  
io  
    import asynchat  
----- MESSAGE FOLLOWS -----  
b'Subject: Test '  
b'X-Peer: 127.0.0.1'  
b''  
b'A test message to check the email sending using STMP'  
----- END MESSAGE -----
```



Lecture 9

Simple Email Client Example

```
import smtplib, imghdr
from email.message import EmailMessage
from email.utils import make_msgid

sender = input("Enter sender email address: ")
recipient = input("Enter recipient email address: ")
msg = EmailMessage()
msg['subject'] = 'A new message with attachment'
msg['from'] = sender
msg['to'] = recipient
msg['Message-ID'] = make_msgid()
msg.set_content("Image attached ...")
```



Lecture 9

Simple Email Client Example

Cont'd

```
with open('image.jpg', 'rb') as f:  
    file = f.read()  
    fileType = imghdr.what(f.name); print(fileType)  
    fileName = f.name  
msg.add_attachment(file, maintype='image', subtype=fileType, filename = fileName)
```

```
mailserver = 'smtp.gmail.com'; mailport = 587  
session = smtplib.SMTP(mailserver, mailport)  
session.ehlo()  
session.sendmail(sender, recipient, msg.as_string())  
print(f"You email is sent to {recipient}.")  
session.quit()
```

Incoming Mail (IMAP) Server

imap.gmail.com Requires SSL: Yes Port: 993

Outgoing Mail (SMTP) Server

smtp.gmail.com Requires SSL: Yes Requires
TLS: Yes (if available) Requires Authentication:
Yes Port for SSL: 465 Port for TLS/STARTTLS:
587



Lecture 9

Simple Email Client Example

Cont'd

```
drhamada@drhamada-Latitude-E6520:/mnt/3126DF085226C5FC$ /bin/python3 /mnt/3126DF085226C5FC/Codes/server.py
Enter sender email address: hamada@esu.ac.ae
Enter recipient email address: hamada_mbsoft@yahoo.com
Enter your email subject: test
Enter your email message. Press Enter when finished. test
Traceback (most recent call last):
  File "/mnt/3126DF085226C5FC/Codes/server.py", line 31, in <module>
    send_email(sender, recipient)
  File "/mnt/3126DF085226C5FC/Codes/server.py", line 24, in send_email
    session.sendmail(sender, recipient, msg.as_string())
  File "/usr/lib/python3.10/smtplib.py", line 887, in sendmail
    raise SMTPSenderRefused(code, resp, from_addr)
smtplib.SMTPSenderRefused: (530, b'5.7.0 Must issue a STARTTLS command first.
.47 - gsmtplib', 'hamada@esu.ac.ae')
```



Lecture 9

Less Secure Apps

<https://support.google.com/accounts/answer/6010255?hl=en>

To help keep your account secure, from **May 30, 2022**, Google no longer supports the use of third-party apps or devices which ask you to sign in to your Google Account using only your username and password.

Important: This deadline does not apply to Google Workspace or Google Cloud Identity customers. The enforcement date for these customers will be announced on the Workspace blog at a later date.

For more information, continue to read.



Google Account

← Less secure app access

Some apps and devices use less secure sign-in technology, which makes your account vulnerable. You can turn off access for these apps, which we recommend, or turn it on if you want to use them despite the risks. Google will automatically turn this setting OFF if it's not being used. [Learn more](#)

This setting is not available for accounts with 2-Step Verification enabled. Such accounts require an application-specific password for less secure apps access. [Learn more](#)



Lecture 9

Less Secure Apps

Cont'd

Google Account



Home



Personal info



Data & privacy



Security



People & sharing



Payments & subscriptions

Signing in to Google



Password

Last changed Jun 27, 2018



2-Step Verification

☒ On



App passwords

None



جامعة الجميع الذكية
EVERYONE'S SMART UNIVERSITY



Google Account

← App passwords

App passwords let you sign in to your Google Account from apps on devices that don't support 2-Step Verification. You'll only need to enter it once so you don't need to remember it. [Learn more](#)

You don't have any app passwords.

Select the app and device you want to generate the app password for.

Select app



Select device



GENERATE

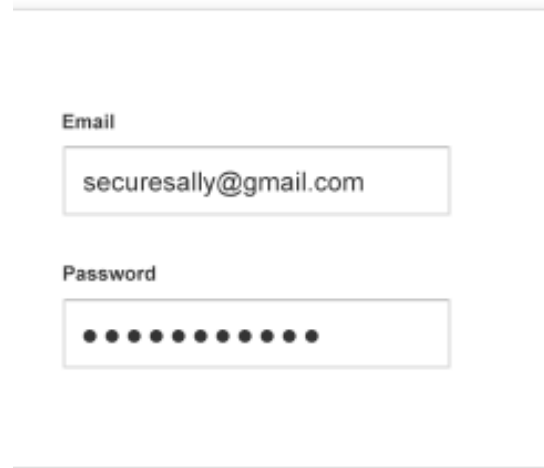


Lecture 9

Less Secure Apps

Cont'd

Generated app password



A screenshot of a login form. It has two input fields: 'Email' and 'Password'. The 'Email' field contains the text 'securesally@gmail.com'. The 'Password' field contains 12 dots, indicating a masked password.

Your app password for your device

yhtn ypwc oagn bfno

How to use it

Go to the settings for your Google Account in the application or device you are trying to set up. Replace your password with the 16-character password shown above.

Just like your normal password, this app password grants complete access to your Google Account. You won't need to remember it, so don't write it down or share it with anyone.

DONE



Lecture 9

Secured Connection in Email Client

```
import getpass
```

```
session = smtplib.SMTP(mailserver, mailport)
```

```
session.ehlo()
```

```
session.starttls() #to encrypt the traffic
```

```
session.ehlo() # re-identify themselves as encrypted connection
```

```
# "yhtnypwcoagnbfno"
```

```
password = getpass.getpass(prompt="Enter you email password:")
```

```
session.login(sender, password)
```



- poplib has these methods:
 - POP3(): creates a POP3 object.
 - POP3_SSL(): creates a secured POP3 object.
 - user(): sets the username to get the mailbox.
 - pass_(): sets the password for the username.
 - stat(): returns mailbox status in a tuple format (message count, mailbox size).
 - list(): retrieves a list of messages in the tuple format (responsecode, msglist, sizeofresponse).
 - retr(): retrieves a single message in the form of a tuple (responsecode, lines, size) and marks it as read.



Lecture 9

Retrieving E-mails Using POP3

Cont'd

```
import getpass, poplib

GPOP3SERVER = 'pop.gmail.com'
POP3PORT = 995
username = input("Enter your email user ID: ")
password = getpass.getpass(prompt="Enter your email password: ")
mailbox = poplib.POP3_SSL(GPOP3SERVER, POP3PORT)
mailbox.user(username)
mailbox.pass_(password)
num_messages = len(mailbox.list()[1])
print(f"Total emails: {num_messages}")
print("Getting last message")
for msg in mailbox.retr(num_messages)[1]:
    print(msg)
mailbox.quit()
```



- imaplib has these methods:
 - `IMAP4_SSL()`: creates a secure IMAP object.
 - `login()`: logs in to the server.
 - `list()`: lists the folders present in mailbox. Returns a tuple of ('Ok', [list])
 - `select()`: selects a specific mail folder to get messages from it. Returns a tuple of ('Ok', [list of one item specify number of emails in this folder])
 - `fetch()`: returns a particular message. Its 1st argu. Is the number of message, the second is the parsing format. Return a tuple of ('Ok', [list of message header]).



Lecture 9

Retrieving E-mails Using IMAP

Cont'd

```
import imaplib, email
imapServer = 'imap.gmail.com'
sender = 'hamada.mbsoft@gmail.com'
password = 'yhthnypwcoagnbfno' #Invalid credentials
conn = imaplib.IMAP4_SSL(imapServer)
conn.login(sender, password)
print(conn.list())
print(conn.select('INBOX'))
conn.select('INBOX')
typ, data = conn.fetch(b'6433','(RFC822)')
print(data[0])
print(data[0][1])
```



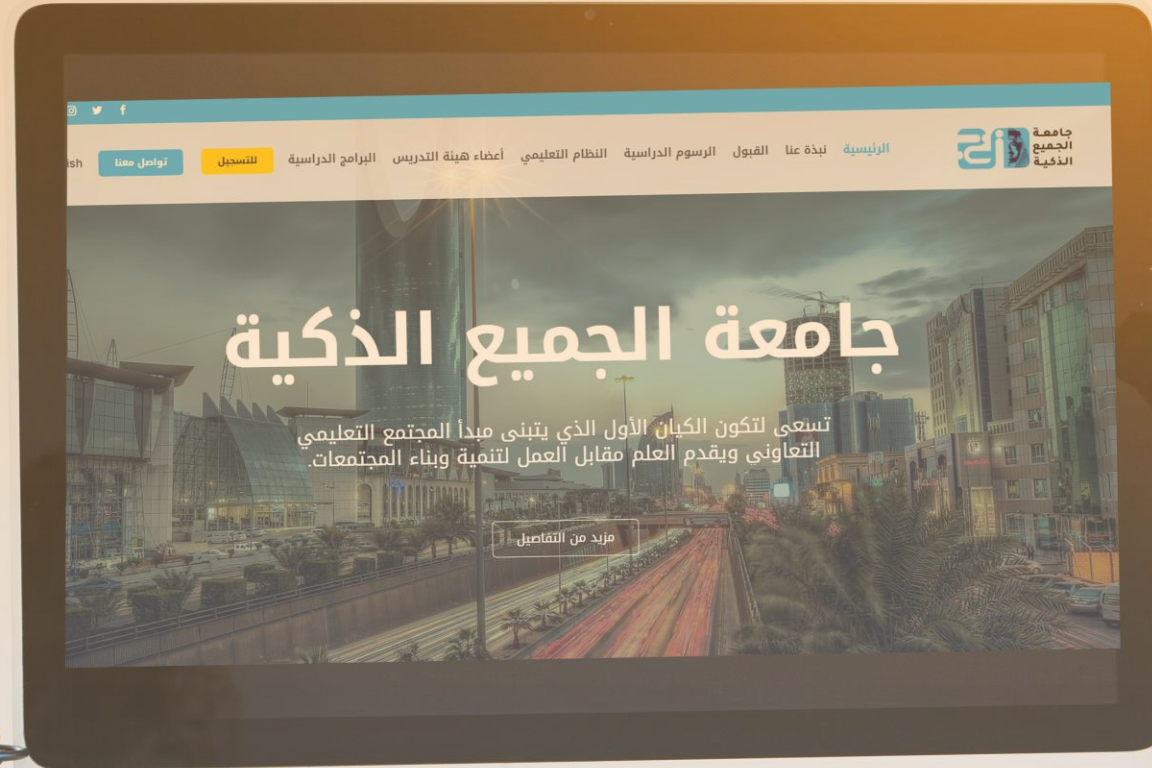
.....

```
# convert data into email message object
msg = email.message_from_string(data[0][1].decode('utf-8'))
print("To:", msg['to'])
print("From:", msg['from'])
print("Subject: ", msg['subject'])
# use walk() to get message parts and check its content type
for p in msg.walk():
    if p.get_content_type() == 'text/plain':
        body = p.get_payload()
print(body)
```



- Exercise 1:
Create your own Mail Client and use it to send e-mail to "hamada@esu.ac.ae".





جامعة الجميع الذكية
EVERYONE'S SMART UNIVERSITY

THANK YOU