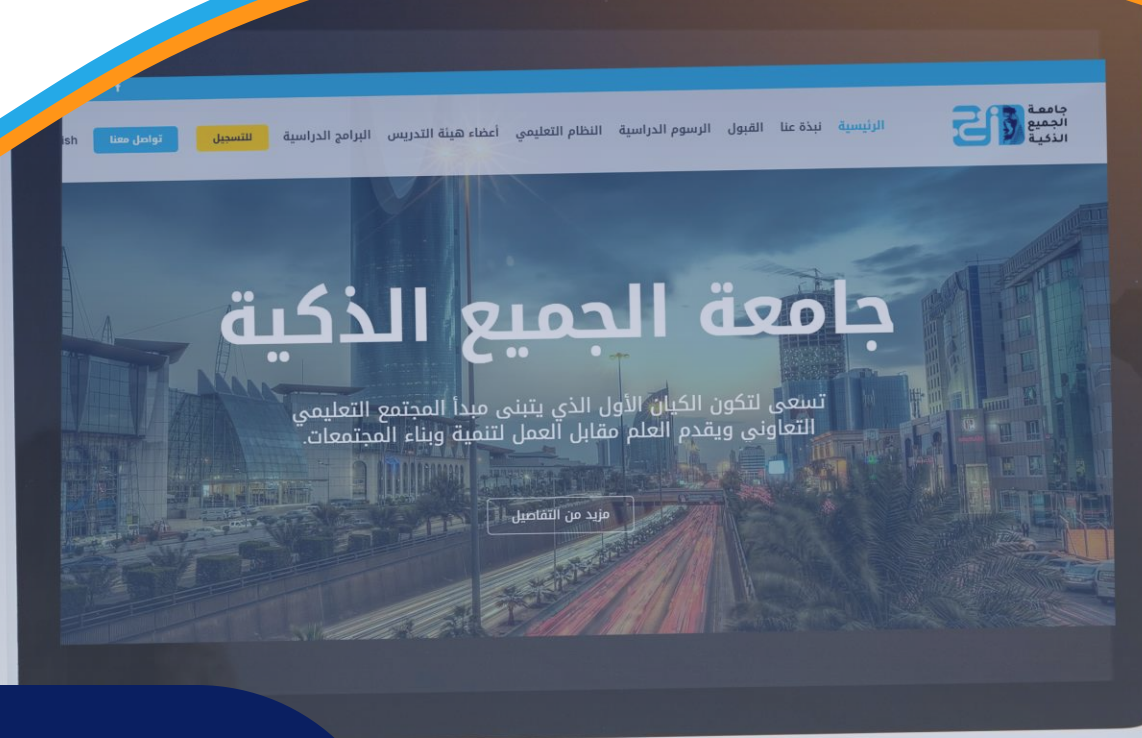




جامعة الجميع الذكية
EVERYONE'S SMART UNIVERSITY

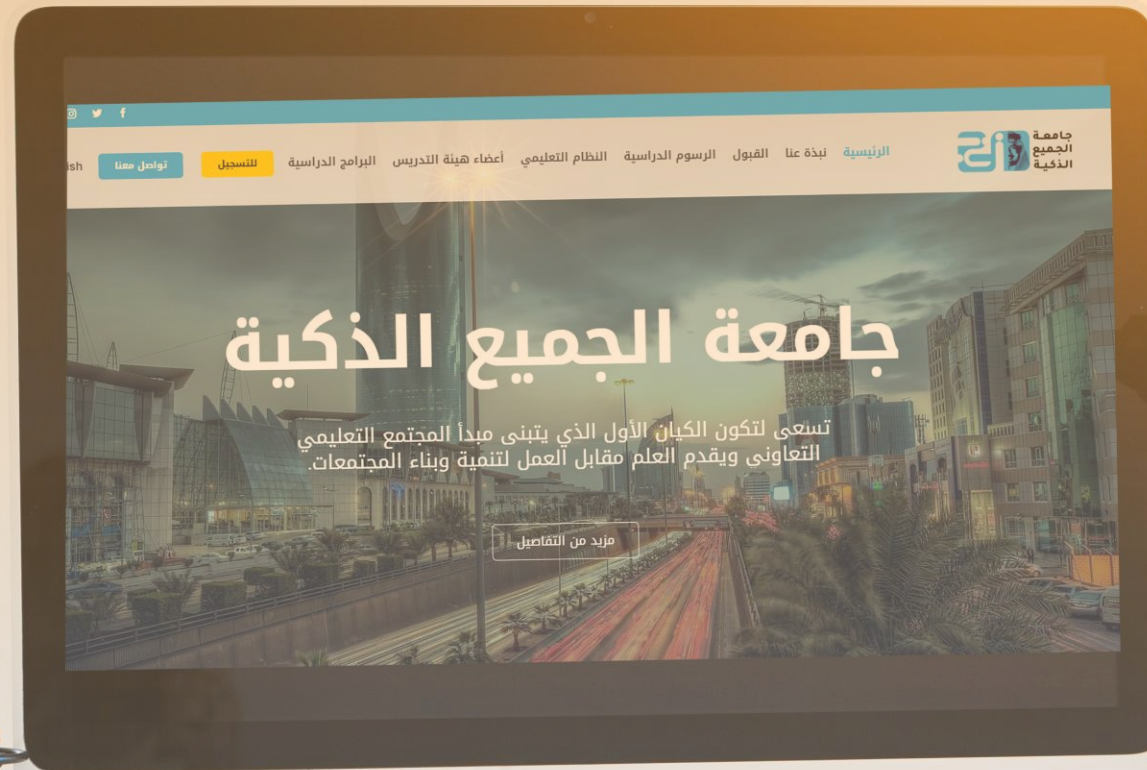


Prepared by
Dr. Hamada I. AbdulWakel
Lecture 10

CYB615
Network Programming



جامعة الجميع الذكية
EVERYONE'S SMART UNIVERSITY



Module10: File Transfer Protocol Using Python

- The File Transfer Protocol.
- FTP Python Module.
- Creating an FTP Client.

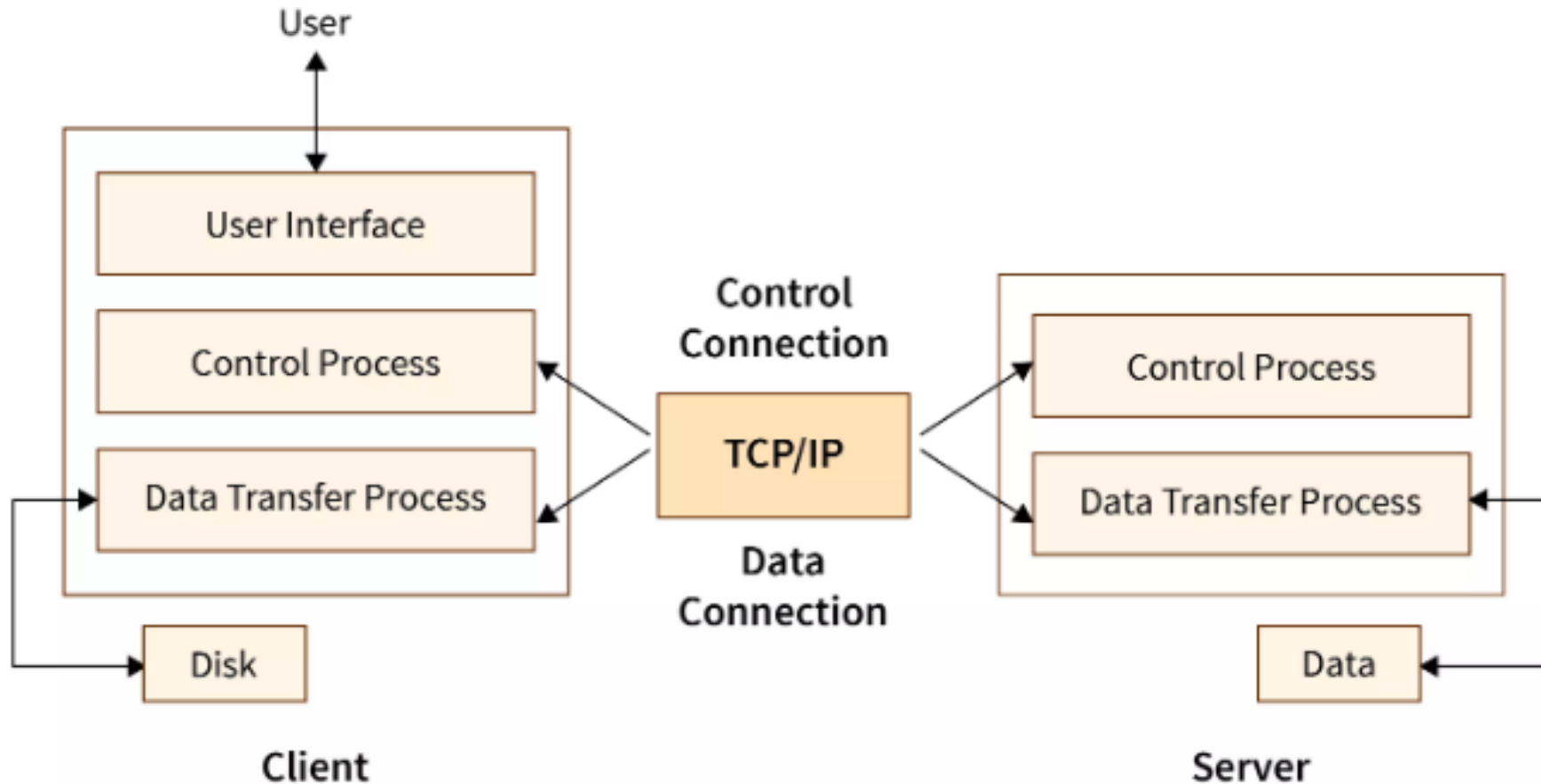


- FTP is a standard internet protocol provided by TCP/IP used for transmitting/downloading files between client and server.
- FTP provides the sharing of files.
- FTP overcomes the problem of transferring and translating file types between different operating systems.



Lecture 10

Mechanism of FTP



- FTP client has three components:
 - The user interface.
 - The control process.
 - The data transfer process.
- The server has two components:
 - The server control process.
 - The server data transfer process.



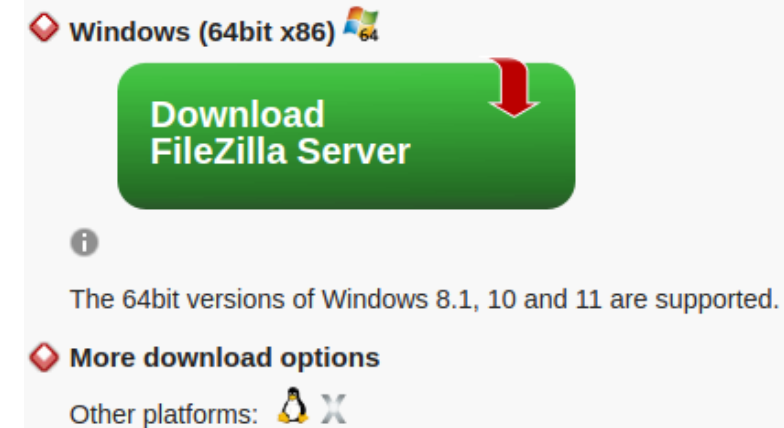
- The control connection:
 - Uses very simple rules for communication.
 - Remains opened during the entire interactive FTP session.
- The data transfer connection:
 - Uses very complex rules as data types may vary.
 - Opens when a command comes for transferring the files and closes when the file is transferred.



- **OPEN**: used to open new connection with the server.
- **USER**: sends the user identification to the server.
- **PASS**: sends the user password to the server.
- **CWD**: allows the user to work with a different directory without altering his login information.
- **MKD**: causes the directory specified in the path-name to be created as a directory.
- **PUT**: used to put a file in the server.
- **CLOSE**: used to close connection with the server.



- Download [Filezilla server](#).
- Run the executable and complete the installation. While installation, set the server administration password and remember the server administration port.



Lecture 10

Working with Local FTP Server

Cont'd

The screenshot displays the FileZilla Server 1.2.0 Administration interface. A 'Connection' dialog box is open, showing the following fields and options:

- Host: 127.0.0.1
- Port: 14148
- Password: (masked with dots)
- ☒ Save the password
- ☐ Automatically connect to this server at startup
- Buttons: OK, Cancel

The main window shows a 'Connect to FileZilla FTP Server' button. The status bar at the bottom indicates 'Disconnected'. A 'Server' menu is open, showing options: Connect... (Ctrl-H), Disconnect... (Ctrl-D), Configure... (Ctrl-F), and Quit. A message box displays: 'Message: Successfully connected to server 127.0.0.1.' Below the message is a table with the following columns: Date, Session ID, Username, Host, Transfer status, Path, Size, Inst. Speed, Avg. Speed, and Progress.

Date	Session ID	Username	Host	Transfer status	Path	Size	Inst. Speed	Avg. Speed	Progress
------	------------	----------	------	-----------------	------	------	-------------	------------	----------



Lecture 10

Working with Local FTP Server

Cont'd

Settings for server 127.0.0.1

FTP Server

Connection Connection security

Listeners:

Address	Port	Protocol
0.0.0.0	21	Explicit FTP over TLS and insecure plain FTP
::	21	Explicit FTP over TLS and insecure plain FTP

Add Remove

Settings for server 127.0.0.1

FTP Server

Users

Available users

<system user>
abcd

Add Remove Duplicate Rename

General Filters Speed Limits

☒ User is enabled

Credentials:

Require a password to log in

Member of groups:

Mount points:

Virtual path	Native path
/	C:\Users\Nagasudhir\Downloads\Programs\ftp\ftp

Add Remove

Permissions

Access mode:

Read + Write

☒ Apply permissions to subdirectories

☒ Writable directory structure

Description:

OK Cancel Apply



- Python provides FTP module in its standard library called `ftplib` that implements the client side of the FTP protocol.
- Features:
 - Connecting to an FTP server.
 - Navigating its structure.
 - Downloading files from the FTP server.
 - Uploading files to an FTP server.



- `FTP()`: create an FTP client instance.
- `FTP_TLS()`: create an FTP client instance under a secure connection.
- `connect()`: connect to an FTP server.
- `login()`: login to an FTP server.
- `prot_p()`: used with a secure FTP connection.
- `pwd()`: get current working directory in FTP server.
- `cwd()`: change the working directory.
- `nlst()`: return a list of file and folder names in a working directory.



- `storbinary()`: used to upload a file to an FTP server.
- `retrbinary()`: used to download a file from an FTP server.
- `rename()`: rename a file or folder in an FTP server.
- `delete()`: delete a file or folder from an FTP server.
- `rmd()`: delete a folder from an FTP server.
- `mkd()`: create a folder in an FTP server.
- `size()`: return the size of a file in an FTP server.
- `quit()`: close an FTP server connection.



- Connection steps:
 - Set up connection parameters.
 - Create an FTP client instance.
 - Connect to FTP server.
 - Login to FTP server.
 - Perform required operation on FTP server.
 - Close connection.



Lecture 10

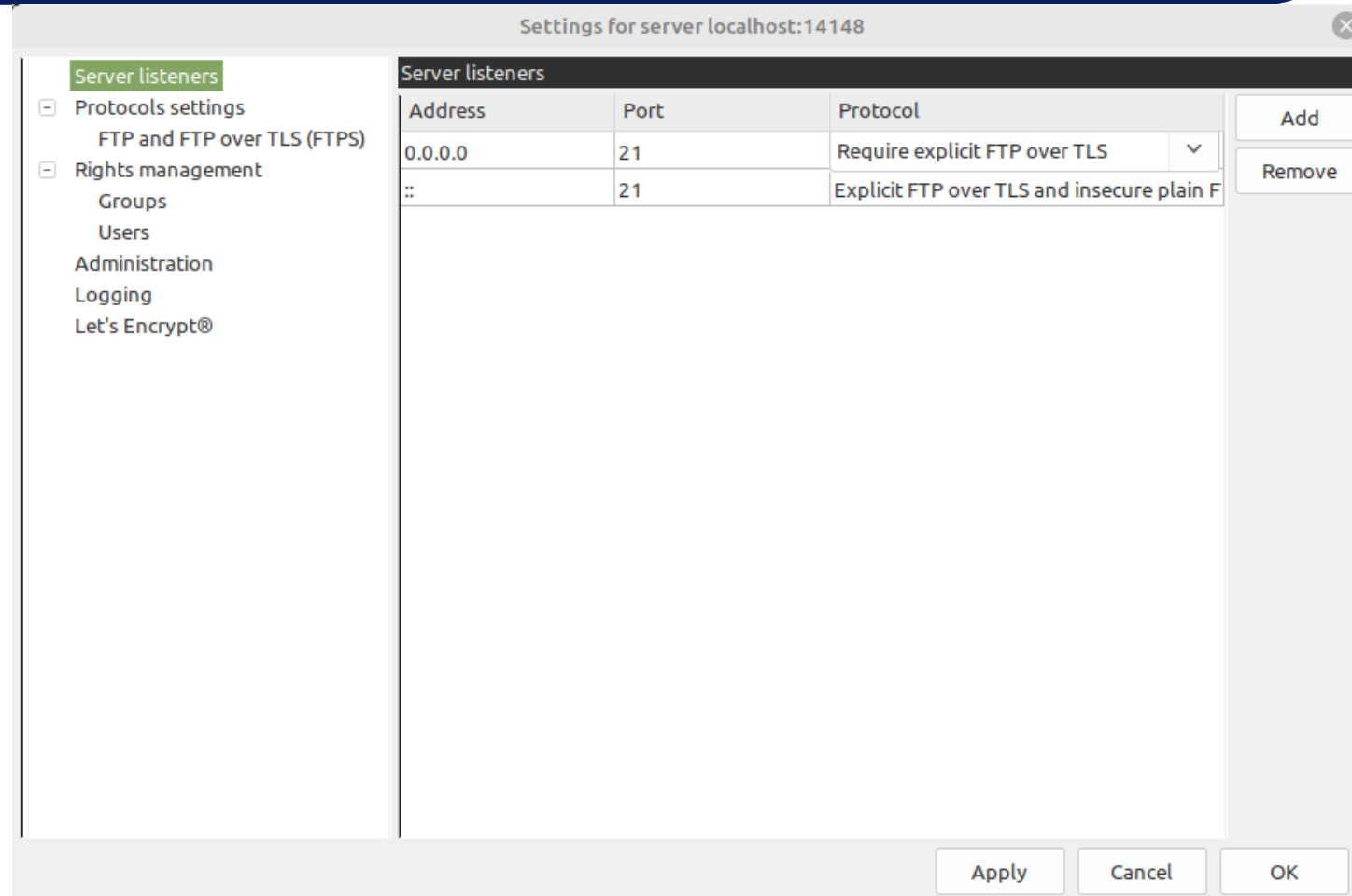
Connection to A Local FTP Server Code

```
import ftplib
ftphost = "localhost"
ftpport = 21
ftpusername = "yourUserName"
ftppassword = "youPassword"
# # of seconds the server waits before ending the connection with the client.
ftp = ftplib.FTP(timeout=30) # to increase connection time
ftp.connect(ftphost, ftpport)
ftp.login(ftpusername, ftppassword)
ftp.quit()
print("FTP handling performed successfully")
```



Lecture 10

Securing The FTP Server Connection



Lecture 10

Securing The FTP Server Connection Code

```
import ftplib
ftphost = "localhost"
ftpport = 21
ftpusername = "yourUserName"
ftppassword = "youPassword"

ftp = ftplib.FTP_TLS(timeout=30)
ftp.connect(ftphost, ftpport)
ftp.login(ftpusername, ftppassword)
ftp.prot_p()
ftp.quit()
print("FTP handling performed successfully")
```



```
ftp.login(ftpusername, ftppassword)
#ftplib.error_perm
ftp.cwd('/mnt/3126D/Codes')
ftp.cwd('/test')
ftp.cwd('/testt')
ftp.pwd()
fileNames = ftp.nlst()
print(fileNames)
ftp.quit()
print("FTP handling performed successfully")
```



```
ftp.login(ftpusername, ftppassword)
with open("message.txt", 'rb') as file:
    rescode = ftp.storbinary("STOR message.txt", file, blocksize= 1024*1024)

if rescode.startswith("226"):
    print("File uploaded seccussefully")
else:
    print("File not uploaded!")
ftp.quit()
print("FTP handling performed successfully")
```



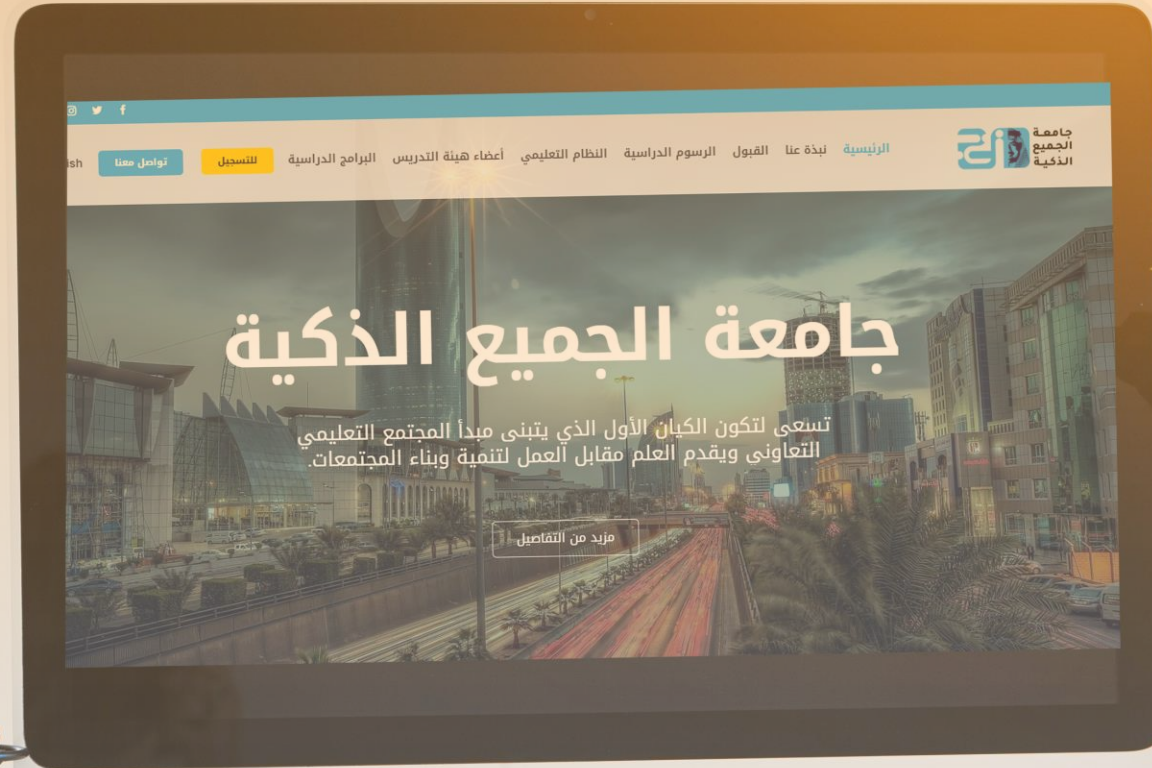
```
ftp.login(ftpusername, ftppassword)
with open("message.txt", 'wb') as file:
    rescode = ftp.retrbinary("RETR message.txt", file.write)

if rescode.startswith("226"):
    print("File downloaded seccussefully")
else:
    print("File not downloaded!")
ftp.quit()
print("FTP handling performed successfully")
```



```
ftp.login(ftpusername, ftppassword)
ftp.rename("oldFileName.extension", "newFileName.extension")
ftp.delete("fileName.extension")
ftp.mkd("folderName")
ftp.rmd("folderName")
print(ftp.size("fileName.extension"))
print(ftp.dir())
ftp.quit()
print("FTP handling performed successfully")
```





جامعة الجميع الذكية
EVERYONE'S SMART UNIVERSITY

THANK YOU