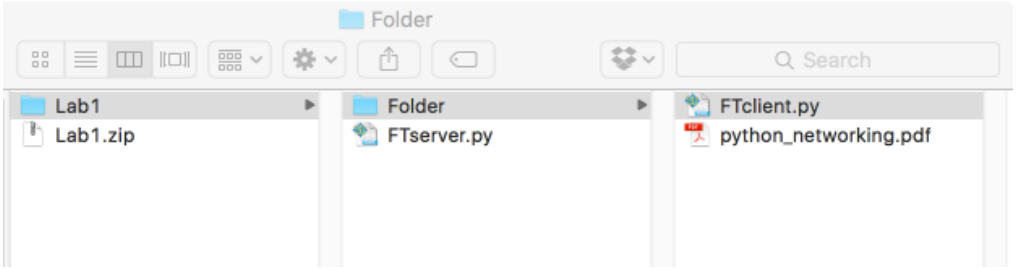
Lab 1 – A simple file transfer Client/Server application

The purpose of this lab exercise is to let you be familiar with socket programming in Python so as to prepare for your programming project. You will be guided through the whole process in designing a simple file transfer application that runs over TCP. The application involves a pair of processes – client and server programs. The server program will be the process that receives a file, which is sent by the client process.

Step 1

- Download the program framework Lab1.zip from the course's Moodle site.
- Unzip it to your Desktop.
- You should find the framework of the server program (FTserver.py) in Lab1 directory and the framework of the client program (FTclient.py) in the Lab1/Folder subdirectory



Step 2 – Implement FTclient.py

- Open the client program FTclient.py with a code editor of your choice and **complete sections A to G**.

FTclient.py

```python
#!/usr/bin/python

import socket
import os.path
import sys

def main(argv):

    # open the target file; get file size
    A

    # create socket and connect to server
    B

    # once the connection is set up; print out
    # the socket address of your local socket
    C

    # send file name and file size as one string separate by ':'
    # e.g., socketprogramming.pdf:435678
    D

    # receive acknowledge - e.g., "OK"
    E

    # send the file contents
    print("Start sending ...")
    F

    # close connection
    print("[Completed]")
    G

if __name__ == '__main__':
    if len(sys.argv) != 4:
        print("Usage: FTclient.py <Server_addr> <Server_port> <filename>")
        sys.exit(1)
    main(sys.argv)
```

Section A

You can use the getsize() method (under os.path module) to check whether a file exists and get its file size.
(https://docs.python.org/3.6/library/os.path.html?highlight=os.path.getsize#os.path.getsize)

To make the task easier, we assume that the target file is located at the same directory as where the FTclient.py program is located at.

Use the python built-in open() method to open the file for reading.
(https://docs.python.org/3.6/library/functions.html#open)

Section B

Follow the description in slide # 19, 20, 31, & 33 of 02-SocketProgramming.pdf to implement this part.

Section C

You can use the getsockname() method to print out the ***local*** socket address of a connected socket. (https://docs.python.org/3.6/library/socket.html?highlight=socket.getsockname#socket.socket.getsockname)

Section D

Use standard string concatenation method to construct the message, then use the string encode() method to convert the string to bytes object, and follow the description in slide # 25, 31, & 33 of 02-SocketProgramming.pdf to implement this part.
(https://docs.python.org/3.6/library/stdtypes.html#str.encode)

Section E

Follow the description in slide # 25, 31, & 33 of 02-SocketProgramming.pdf to implement the receiving of message. You can simply use "OK" as your acknowledgment message.

Section F

Use the file read() method to read in a block of data from the opened file.
(https://docs.python.org/3.6/tutorial/inputoutput.html?highlight=file%20read#methods-of-file-objects)

The best way to send a large file is to send it block by block. As nowadays networks are usually limited to a packet size of 1500 bytes. I suggest you use a block size of 1000 bytes. Here is the recommended logic:

```
while (filesize > 0)
        blockLen = read in at most 1000 bytes from the opened file
        send the block to server
        filesize -= blockLen
```

Section G

Follow the description in slide # 27 & 33 of 02-SocketProgramming.pdf to implement this part.

Step 3 – Implement FTserver.py

- Open the server program (FTserver.py) with a code editor and **complete sections H to O**.

FTserver.py

```
#!/usr/bin/python

import socket
import sys

def main(argv):
    # set port number
    # default is 32341 if no input argument
    H

    # create socket and bind
    I

    # listen and accept new connection
    J

    # print out peer socket address information
    K

    # receive file name, file size; and create the file
    L

    # send acknowledge - e.g., "OK"
    M

    # receive the file contents
    print("Start receiving . . .")
    N

    # close connection
    print("[Completed]")
    O

if __name__ == '__main__':
    if len(sys.argv) > 2:
        print("Usage: FTserver [<Server_port>]")
        sys.exit(1)
    main(sys.argv)
```

Section H

Python uses the same notional as in C/C++ in handling command-line input argument. (https://docs.python.org/3.6/library/sys.html?highlight=sys.argv#sys.argv)

Section I

Follow the description in slide # 20, 21, 31, & 33 of 02-SocketProgramming.pdf to implement this part.

Section J

Follow the description in slide # 22, 23, 31, & 33 of 02-SocketProgramming.pdf to implement this part.

Section K

Using the example program in slide # 33 of 02-SocketProgramming.pdf as a hint to implement this part.

Section L

Use the socket recv() method to receive a message; use open() method to create a new file for "write".

To extract individual component from the message, you can use the bytes split() or string split() method to break the message into pieces.
(https://docs.python.org/3.6/library/stdtypes.html#str.split)
(https://docs.python.org/3.6/library/stdtypes.html#bytes.split)

Print a line to indicate that the program has opened a new file.

Section M

Just send the bytes object b'OK' to the other end.

Section N

Now you know how many bytes to receive and you know that you are going to receive many blocks. The size of each block is 1000 bytes. Here is the suggested logic:

```
received = 0
while (received < filesize)
        rmsg = receive one block of at most 1000 bytes
        write rmsg to the opened file
        received += size of rmsg
```

Section O

Use close() to close all sockets and opened file.

Step 4 – Test the application

- Test the client/server application locally by running the client and server programs in your PC/laptop.
  python3 FTserver.py
  python3 FTclient.py localhost 32341 python_networking.pdf

  We should expect to see a new python_networking.pdf file is created in the same directory where FTserver.py program is located. ***In principle, you should be able to open and read this file with a standard PDF reader.***
- Test the client/server application across machines.
  python3 FTserver.py
  python3 FTclient.py 192.168.0.100 32341 python_networking.pdf

Here is a sample output when running the application on the same machine:

Lab1-sol — -bash — 80×24

```
Anthonys-MacBook-Pro:Lab1-sol atctam$ python3 FTserver.py
Connection established. Here is the remote peer info: ('127.0.0.1', 49847)
Open a file with name 'python_networking.pdf' with size 48655 bytes
Start receiving . . .
[Completed]
Anthonys-MacBook-Pro:Lab1-sol atctam$
```

Folder — -bash — 80×24

```
Anthonys-MacBook-Pro:Folder atctam$ python3 FTclient.py localhost 32341 python_n
etworking.pdf
Connection established. My socket address is ('127.0.0.1', 49847)
Start sending ...
[Completed]
Anthonys-MacBook-Pro:Folder atctam$
```