SRM INSTITUTE OF SCIENCE AND TECHNOLOGY

SRM Nagar, Kattankulathur – 603 203,

SCHOOL OF COMPUTING

DEPARTMENT OF NETWORKING & COMMUNICATIONS

Course Project

Title : Multi-threaded File Transfer using TCP Socket in Python

Course Code : 18CSC310J

Course Name : Data Centric Networking and System Design

Faculty : Dr. N. Senthamarai

Team Members:

Nune Nitesh – RA2011028010124

Marla Sai Ruthwik – RA2011028010124

Manthuri Hrithikesh – RA2011028010133

Gatta Venkata Amruth – RA2011028010136

INDEX

OBJECTIVE:

A Multi threaded file transfer client server program build using a python programming language. The server has a capability to handle multiple clients concurrently at the same time by using threading. The server assigns each client a thread to handle the work for the client.

PROJECT OVERVIEW:

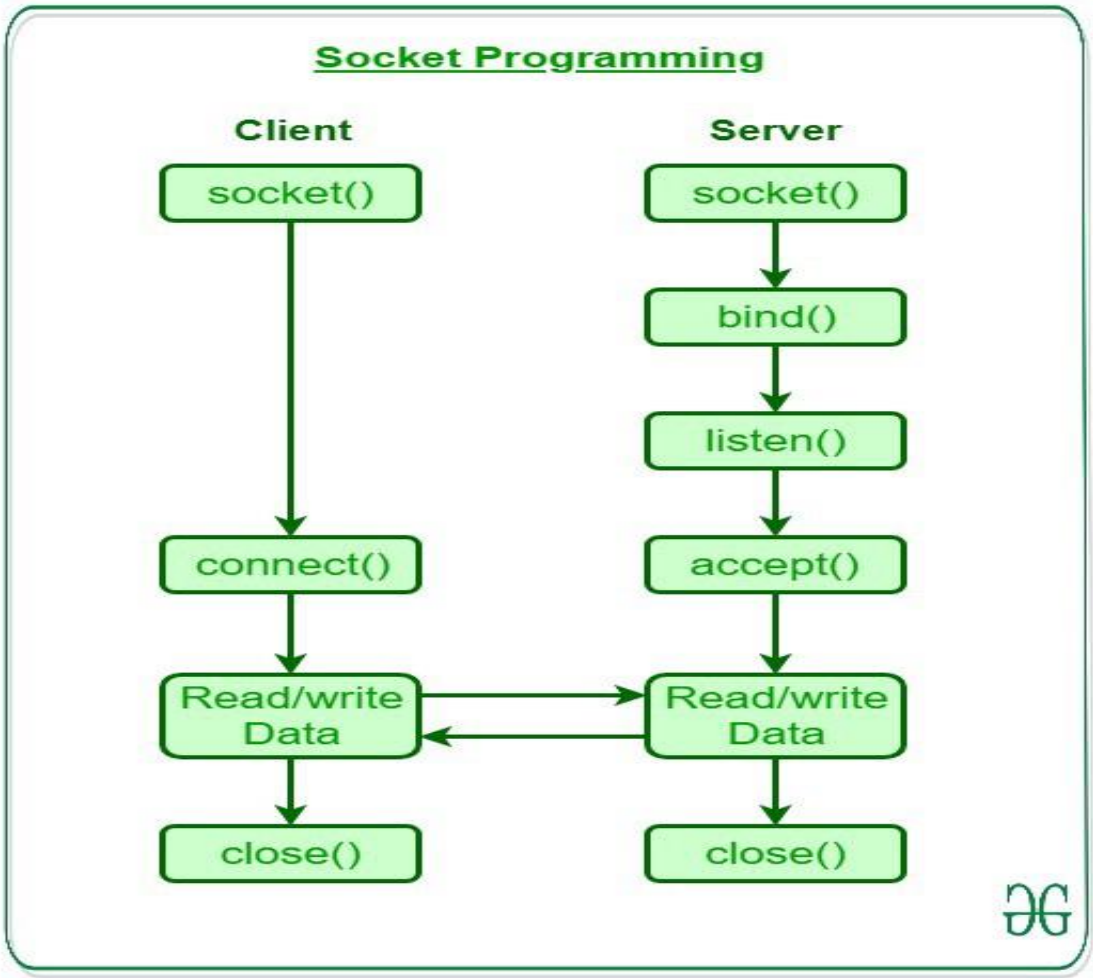
The server program can handle an arbitrary number of concurrent connections and file exchanges, only limited by system configuration or memory. The server is started without any parameters and creates a TCP socket at an OS-assigned port. It prints out the assigned port number and stores it in a local file port, which is used when starting clients. The server listens on its main socket and accepts client connections as they arrive. Clients perform an upload or download operation or instruct the server to terminate.

REQUIREMENT ANALYSIS

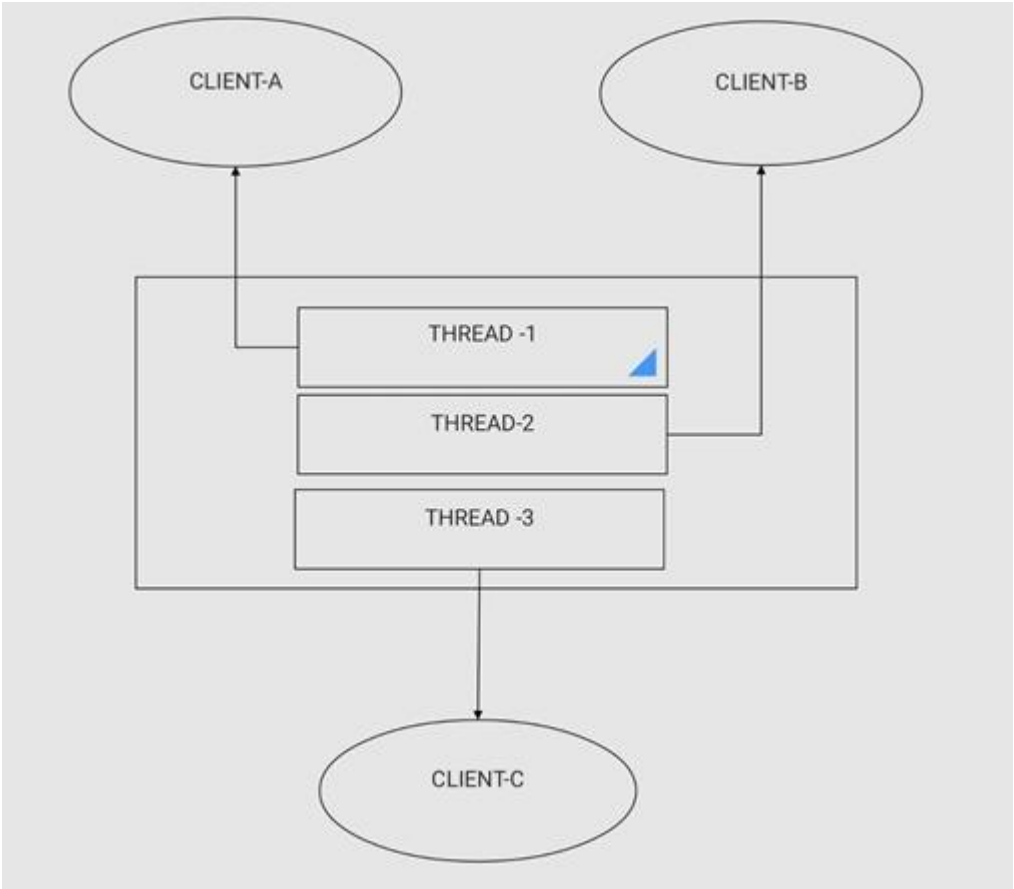
- Socket programming
- Multi-threading
- Programming Language – PYTHON
- Server side and Client side script
- Platform used - ATOM

ARCHITECTURE AND DESIGN

SERVER CLIENT ARCHITECTURE



MULTITHREADING ARCHITECTURE



BASIC FUNCTIONALITIES

- LIST : List all the files from the server data.
- UPLOAD : Upload a file to the server data.
- DOWNLOAD : Download a file to the client data.
- DELETE : Delete a file from the server data.
- LOGOUT : Disconnect from the server.
- HELP : List all the commands.

CODE PARAMETERS

- PORT : Specifies port address, present in both client and server.
- FORMAT : Specifies format of packet that is sent or received over the network.
- SERVER_DATA_PATH : Specifies the server data path in both client and server side.
- CLIENT_DATA_PATH : Specifies the data path in client side.
- ADDR : Holds the information about the IP address and Port address.

CODES:

Client.py

```
import socket

IP = socket.gethostbyname(socket.gethostname())
PORT = 4456
ADDR = (IP, PORT)
FORMAT = "utf-8"
SIZE = 1024

def main():
    client = socket.socket(socket.AF_INET, socket.SOCK_STREAM)
    client.connect(ADDR)

    while True:
        data = client.recv(SIZE).decode(FORMAT)
        cmd, msg = data.split("@")

        if cmd == "DISCONNECTED":
            print(f"[SERVER]: {msg}")
            break
        elif cmd == "OK":
            print(f"{msg}")

        data = input("> ")
        data = data.split(" ")
        cmd = data[0]
```

```
        if cmd == "HELP":
            client.send(cmd.encode(FORMAT))
        elif cmd == "LOGOUT":
            client.send(cmd.encode(FORMAT))
            break
        elif cmd == "LIST":
            client.send(cmd.encode(FORMAT))
        elif cmd == "DELETE":
            client.send(f"{cmd}@{data[1]}".encode(FORMAT))
        elif cmd == "UPLOAD":
            path = data[1]

            with open(f"{path}", "r") as f:
                text = f.read()

            filename = path.split("/")[-1]
            send_data = f"{cmd}@{filename}@{text}"
            client.send(send_data.encode(FORMAT))

    print("Disconnected from the server.")
    client.close()

if __name__ == "__main__":
    main()
```

Server.py

```
import os
import socket
import threading

IP = socket.gethostbyname(socket.gethostname())
PORT = 4456
ADDR = (IP, PORT)
SIZE = 1024
FORMAT = "utf-8"
SERVER_DATA_PATH = "server_data"
```

```
def handle_client(conn, addr):
    print(f"[NEW CONNECTION] {addr} connected.")
    conn.send("OK@Welcome to the File Server.".encode(FORMAT))

    while True:
        data = conn.recv(SIZE).decode(FORMAT)
        data = data.split("@")
        cmd = data[0]

        if cmd == "LIST":
            files = os.listdir(SERVER_DATA_PATH)
            send_data = "OK@"

            if len(files) == 0:
                send_data += "The server directory is empty"
            else:
                send_data += "\n".join(f for f in files)
            conn.send(send_data.encode(FORMAT))

        elif cmd == "UPLOAD":
            name, text = data[1], data[2]
            filepath = os.path.join(SERVER_DATA_PATH, name)
            with open(filepath, "w") as f:
                f.write(text)

            send_data = "OK@File uploaded successfully."
            conn.send(send_data.encode(FORMAT))

        elif cmd == "DELETE":
            files = os.listdir(SERVER_DATA_PATH)
            send_data = "OK@"
            filename = data[1]

            if len(files) == 0:
                send_data += "The server directory is empty"
            else:
                if filename in files:
                    os.system(f"rm {SERVER_DATA_PATH}/{filename}")
                    send_data += "File deleted successfully."
                else:
                    send_data += "File not found."
```

```
            conn.send(send_data.encode(FORMAT))

        elif cmd == "LOGOUT":
            break
        elif cmd == "HELP":
            data = "OK@"
            data += "LIST: List all the files from the server.\n"
            data += "UPLOAD <path>: Upload a file to the server.\n"
            data += "DELETE <filename>: Delete a file from the server.\n"
            data += "LOGOUT: Disconnect from the server.\n"
            data += "HELP: List all the commands."

            conn.send(data.encode(FORMAT))

    print(f"[DISCONNECTED] {addr} disconnected")
    conn.close()

def main():
    print("[STARTING] Server is starting")
    server = socket.socket(socket.AF_INET, socket.SOCK_STREAM)
    server.bind(ADDR)
    server.listen()
    print(f"[LISTENING] Server is listening on {IP}:{PORT}.")

    while True:
        conn, addr = server.accept()
        thread = threading.Thread(target=handle_client, args=(conn, addr))
        thread.start()
        print(f"[ACTIVE CONNECTIONS] {threading.activeCount() - 1}")

if __name__ == "__main__":
    main()
```

DEMO & SCREENSHOT:

Server side code

server.py client.py

```python
import os
import socket
import threading

IP = socket.gethostbyname(socket.gethostname())
PORT = 5544
ADDR = (IP ,PORT)
SIZE = 1024
FORMAT = "utf-8"
SERVER_DATA_PATH = "server_data"

"""
CMD@Msg
"""
def handle_client(conn,addr):
    print(f"[NEW CONNECTION] {addr} connected.")
    conn.send("OK@Welcome to the File Server.".encode(FORMAT))
```

server.py client.py

```python
while True:
    data = conn.recv(SIZE).decode(FORMAT)
    data = data.split("@")
    cmd = data[0]

    if cmd == "HELP":
        send_data = "OK@"
        send_data += "LIST: List all the files from the server.\n"
        send_data += "UPLOAD <path>: Upload a file to the server.\n"
        send_data += "DOWNLOAD <filename>: Download a file from the server.\n"
        send_data += "DELETE <filename>: Delete a file from the server.\n"
        send_data += "LOGOUT: Disconnect from the server.\n"
        send_data += "HELP: List all the commands."

        conn.send(send_data.encode(FORMAT))

    elif cmd == "LOGOUT":
        break

    elif cmd == "LIST":
        files = os.listdir(SERVER_DATA_PATH)
        send_data = "OK@"
```

server.py client.py

```python
    elif cmd == "LIST":
        files = os.listdir(SERVER_DATA_PATH)
        send_data = "OK@"

        if len(files) == 0:
            send_data += "The server directory is empty."
        else:
            send_data += "\n".join(f for f in files)
        conn.send(send_data.encode(FORMAT))


    elif cmd == "UPLOAD":
        name = data[1]
        text = data[2]

        filepath = os.path.join(SERVER_DATA_PATH , name)
        with open(filepath ,"w") as f:
            f.write(text)

        send_data = "OK@File uploaded."
        conn.send(send_data.encode(FORMAT))
```

server.py client.py

```python

        elif cmd == "DOWNLOAD":
            name = data[1]

            filepath = os.path.join(SERVER_DATA_PATH, name)

            with open(filepath, "r") as f:
                text = f.read()

            filepath = data[2]

            with open(filepath, "w") as f:
                f.write(text)

            send_data = "OK@File downloaded successfully."
            conn.send(send_data.encode(FORMAT))

        elif cmd == "DELETE":
            files = os.listdir(SERVER_DATA_PATH)
            send_data = "OK@"
            filename = data[1]
```

server.py client.py

```python
                send_data += "The server directory is empty."
            else:
                if filename in files:
                    os.system(f"del  {SERVER_DATA_PATH}\{filename}")
                    send_data += "File deleted."
                else:
                    send_data += "File not found."

            conn.send(send_data.encode(FORMAT))
    print(f"[DISCONNECTED] {addr} disconnected")
def main():
    print("[STARTING] Server is starting")
    server = socket.socket(socket.AF_INET , socket.SOCK_STREAM)
    server.bind(ADDR)
    server.listen()
    print("[LISTENING] Server is listening")
    while True:
        conn, addr = server.accept()
        thread = threading.Thread( target = handle_client, args=(conn, addr) )
        thread.start()
if __name__ == "__main__":
    main()
```

Client side code

server.py client.py

```python
import os
import socket

IP = socket.gethostbyname(socket.gethostname())
PORT = 5544
ADDR = (IP ,PORT)
SIZE = 1024
FORMAT = "utf-8"
SERVER_DATA_PATH = "server_data"
CLIENT_DATA_PATH = "client_data"

def main():
    client = socket.socket(socket.AF_INET , socket.SOCK_STREAM)
    client.connect(ADDR)
    while True:
        data = client.recv(SIZE).decode(FORMAT)
        cmd, msg = data.split("@")
        if cmd == "OK":
            print(f"{msg}")
        elif cmd == "DISCONNECTED":
            print(f"{msg}")
            break
```

server.py client.py

```python
        elif cmd == "DISCONNECTED":
            print(f"{msg}")
            break
        data = input("> ")
        data = data.split(" ")
        cmd =  data[0]

        if cmd == "HELP":
            client.send(cmd.encode(FORMAT))

        elif cmd == "LOGOUT":
            client.send(cmd.encode(FORMAT))
            break

        elif cmd == "LIST":
            client.send(cmd.encode(FORMAT))

        elif cmd == "UPLOAD":
            ## UPLOAD@filename@text
            path = data[1]
            with open(f"{path}", "r") as f:
```

server.py | client.py

```python
        elif cmd == "UPLOAD":
            ## UPLOAD@filename@text
            path = data[1]
            with open(f"{path}", "r") as f:
                text = f.read()
            ##client_data / data.txt
            filename = path.split("/")[-1]
            send_data = f"{cmd}@{filename}@{text}"
            client.send(send_data.encode(FORMAT))

        elif cmd == "DOWNLOAD":
            filepath = os.path.join(CLIENT_DATA_PATH, data[1])
            client.send(f"{cmd}@{data[1]}@{filepath}".encode(FORMAT))

        elif cmd == "DELETE":
            client.send(f"{cmd}@{data[1]}".encode(FORMAT))
    print("Disconnected from the server.")
    client.close()

if __name__ == "__main__":
    main()
```

Server side output

```
PS C:\Users\HP\Desktop\CN> python server.py
[STARTING] Server is starting
[LISTENING] Server is listening
[NEW CONNECTION] ('192.168.56.1', 56549) connected.
```

server.py 59:1

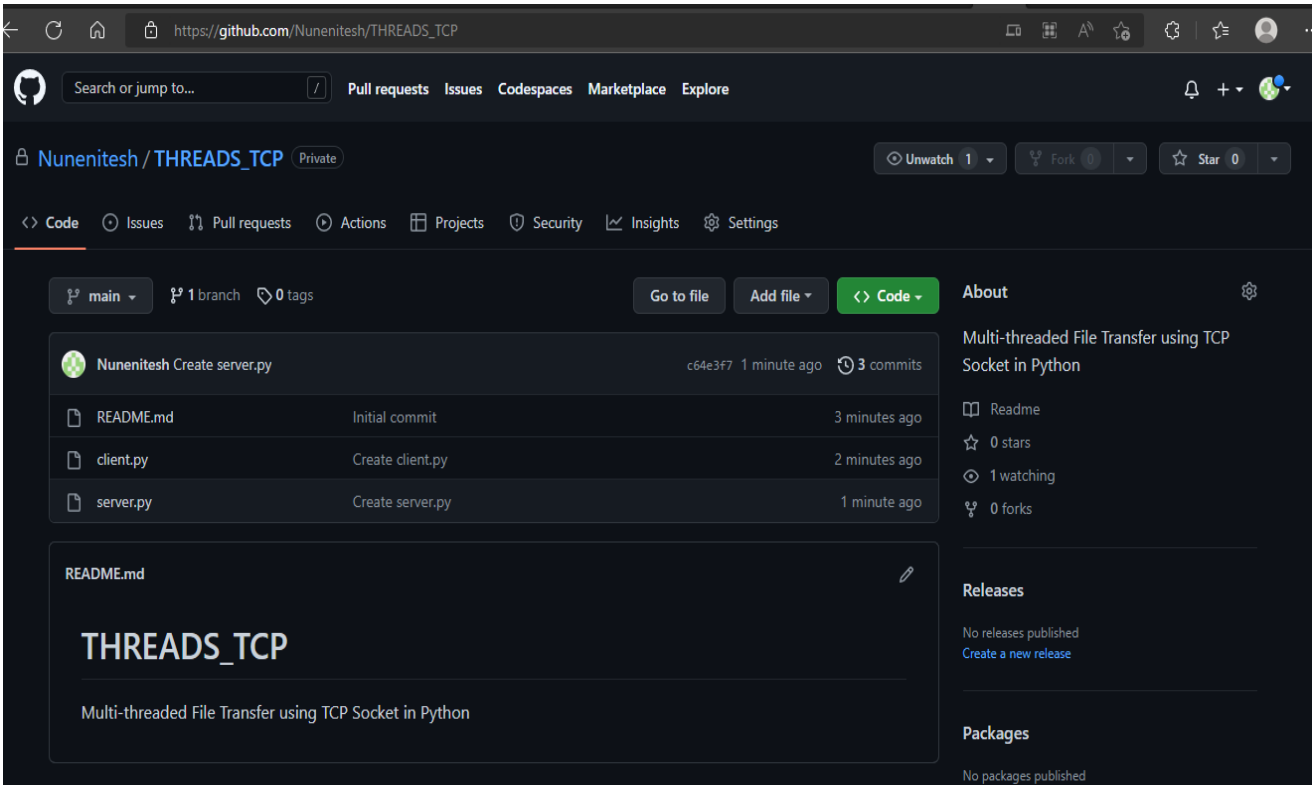
Client side output

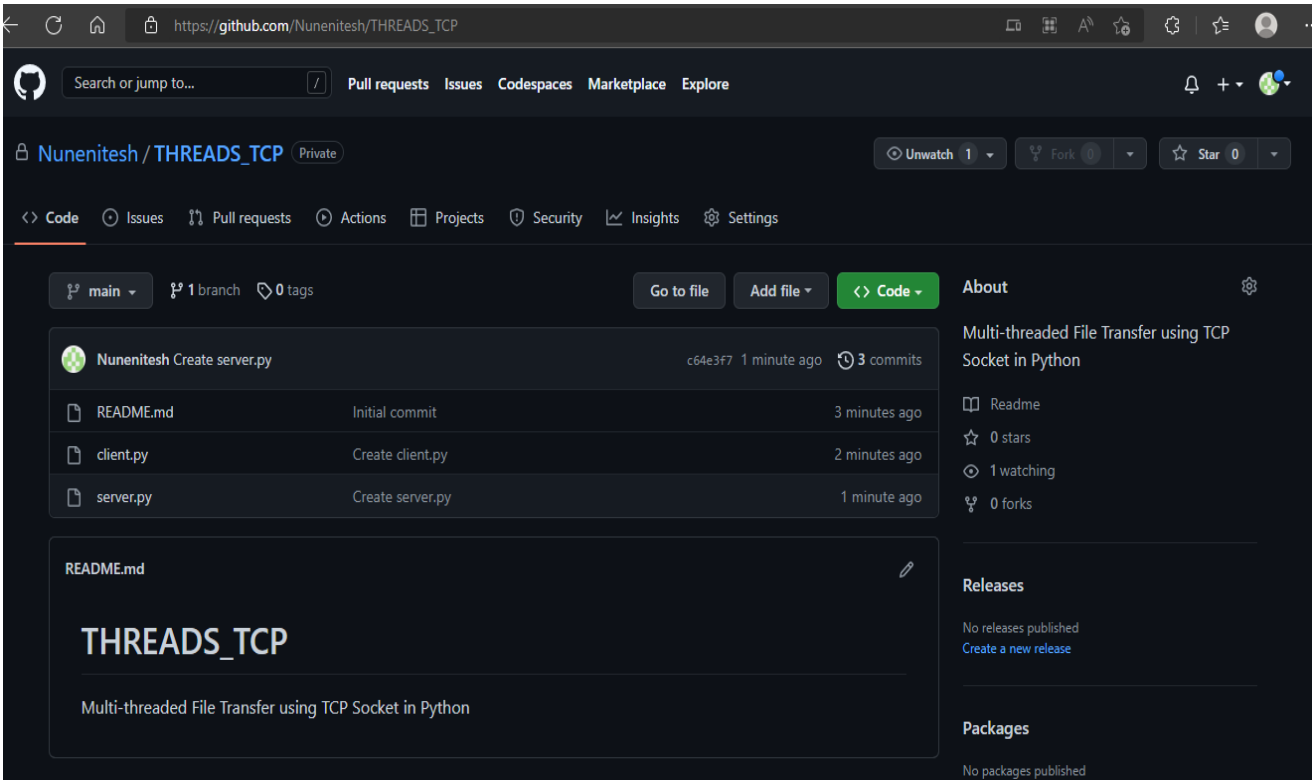
```
PS C:\Users\HP\Desktop\CN> python client.py
Welcome to the File Server.
> HELP
LIST: List all the files from the server.
UPLOAD <path>: Upload a file to the server.
DOWNLOAD <filename>: Download a file from the server.
DELETE <filename>: Delete a file from the server.
LOGOUT: Disconnect from the server.
HELP: List all the commands.
>
```

server.py 59:1

PROOF OF GITHUB UPLOAD:

Repository Link: https://github.com/Nunenitesh/THREADS_TCP





REFERENCES:

- https://github.com/nikhilroxtomar/Multithreaded-File-Transfer-using-TCP-Socket-in-Python
- https://www.geeksforgeeks.org/multithreading-python-set-1/