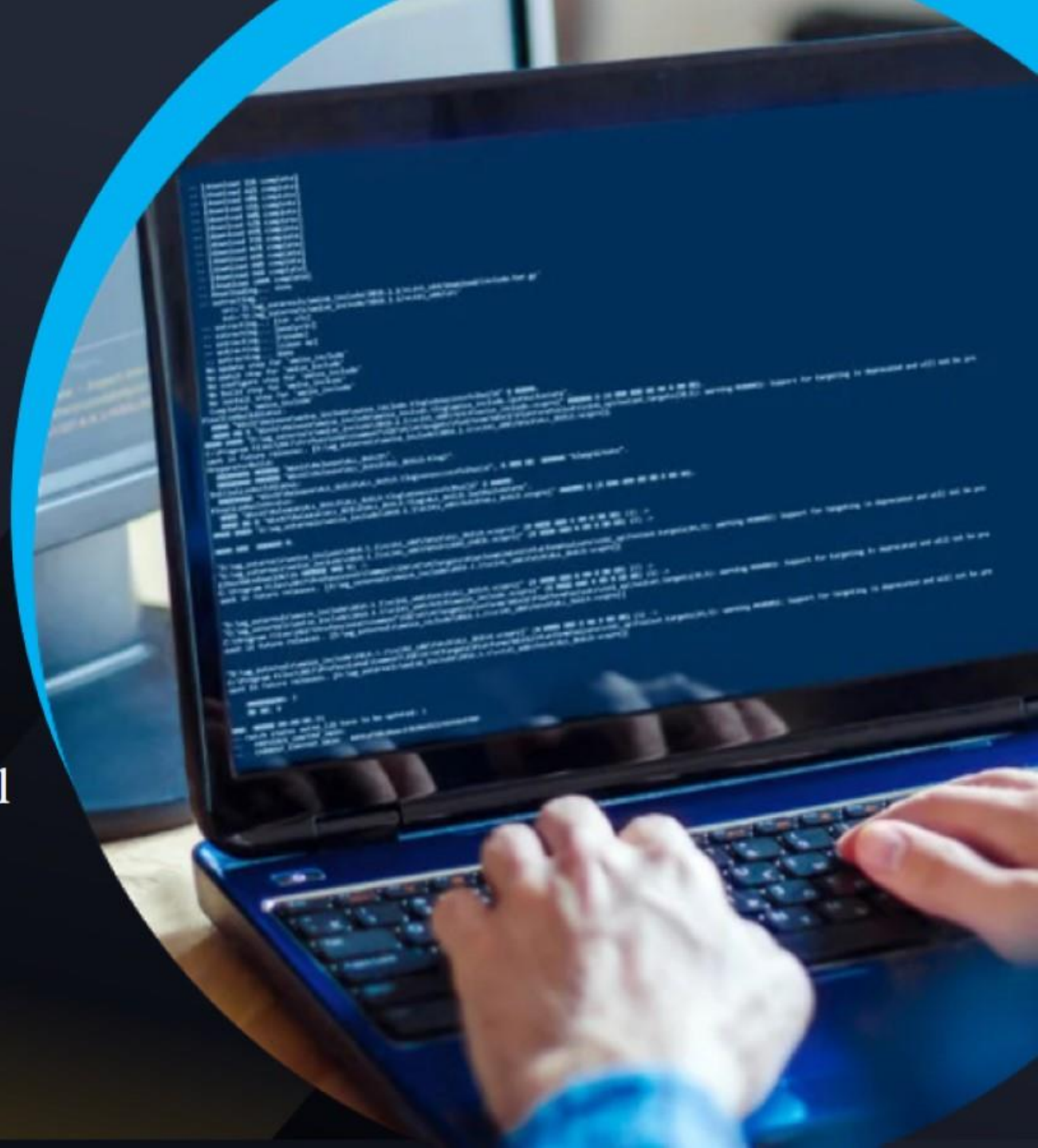


# Network Programming in Python

Prepared by Dr. Hamada I. AbdulWakel

## Lecture 5



# Socket Programming Using Python – Part II

## Lecture 5

Prepared by  
**Dr. Hamada I. AbdulWakel**

[www.midocean.edu.km](http://www.midocean.edu.km)

Lecture

05

**RECAP: LECTURE 4**

- The Client/Server Programming.
- The Communication Protocols.
- TCP and UDP Programming in Python.

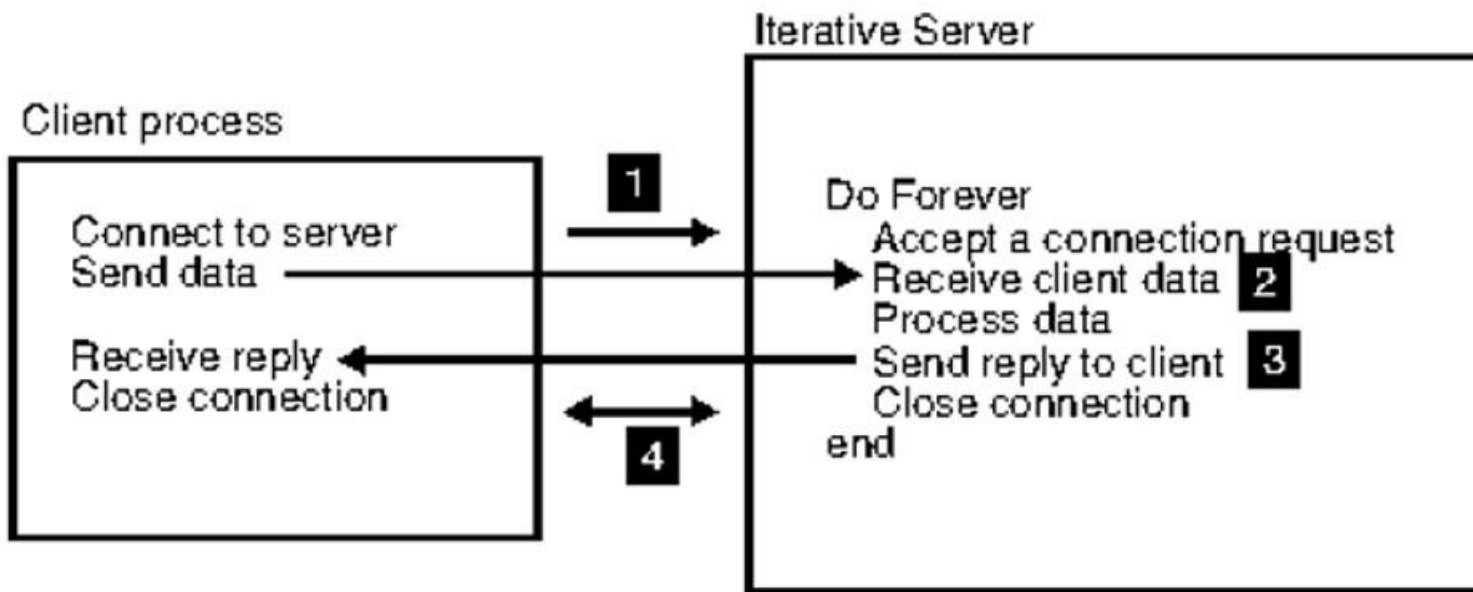
Lecture

05

## LECTURE MAIN POINTS

- Sockets and concurrency.
- The threading programming.
- Programming the multiple client/server model.
- The casting/communication types.

# THE ITERATIVE SERVER



## THE ITERATIVE SERVER

Cont'd

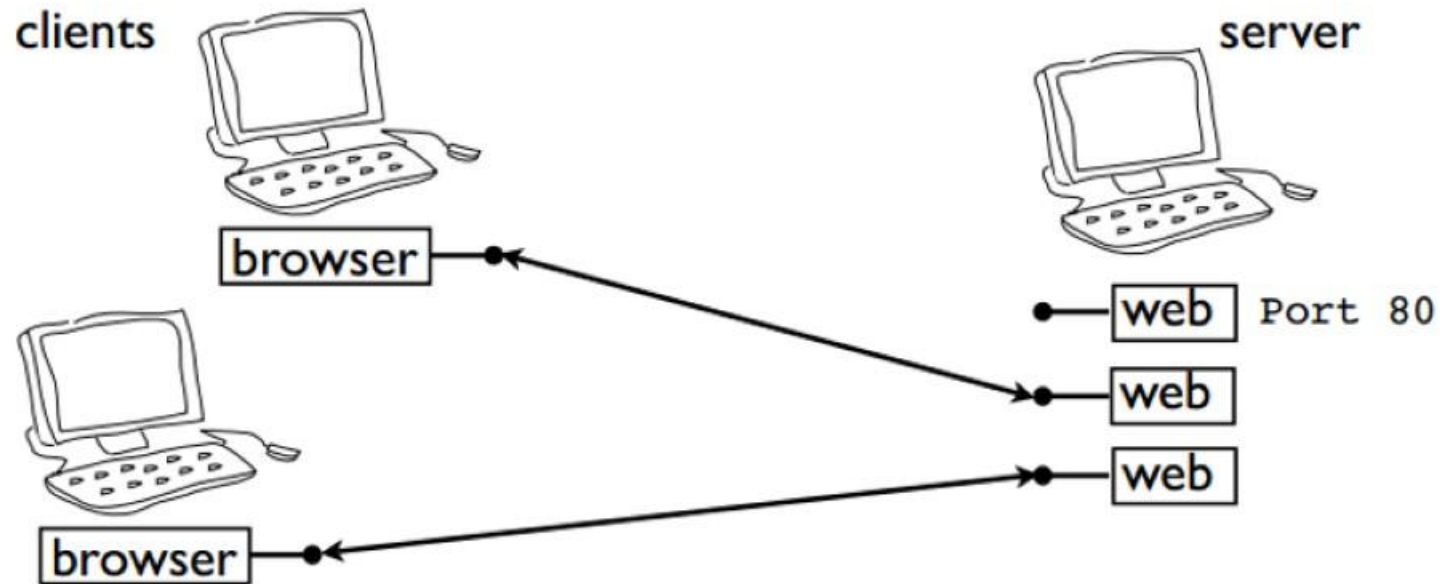
- **An iterative server** iterates through each client, handling it one at a time.
- The following list describes the iterative server socket process.
  - When a connection request arrives, it accepts the connection, receives the client data and does whatever has to be done to build a reply.
  - The server sends the data back to the client.
  - The server closes the socket and waits for the next connection request from the network.

## Lecture

## 05

## SOCKETS AND CONCURRENCY

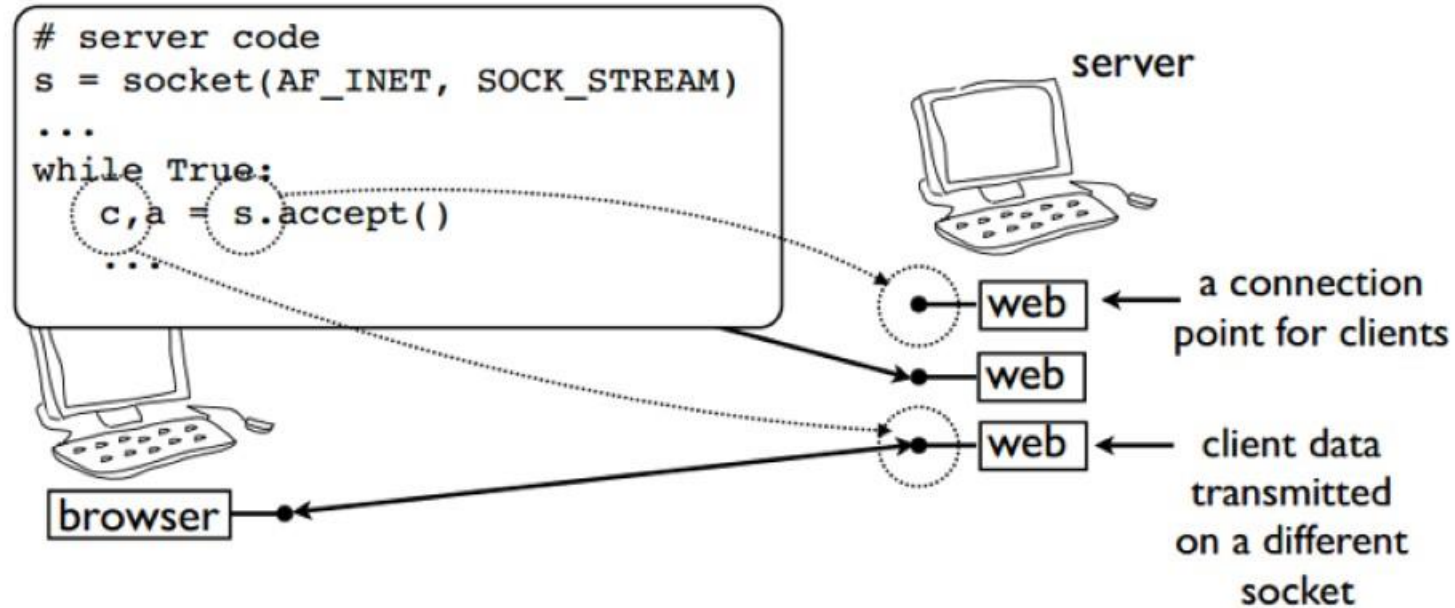
- Servers usually handle multiple clients.



## SOCKETS AND CONCURRENCY

Cont'd

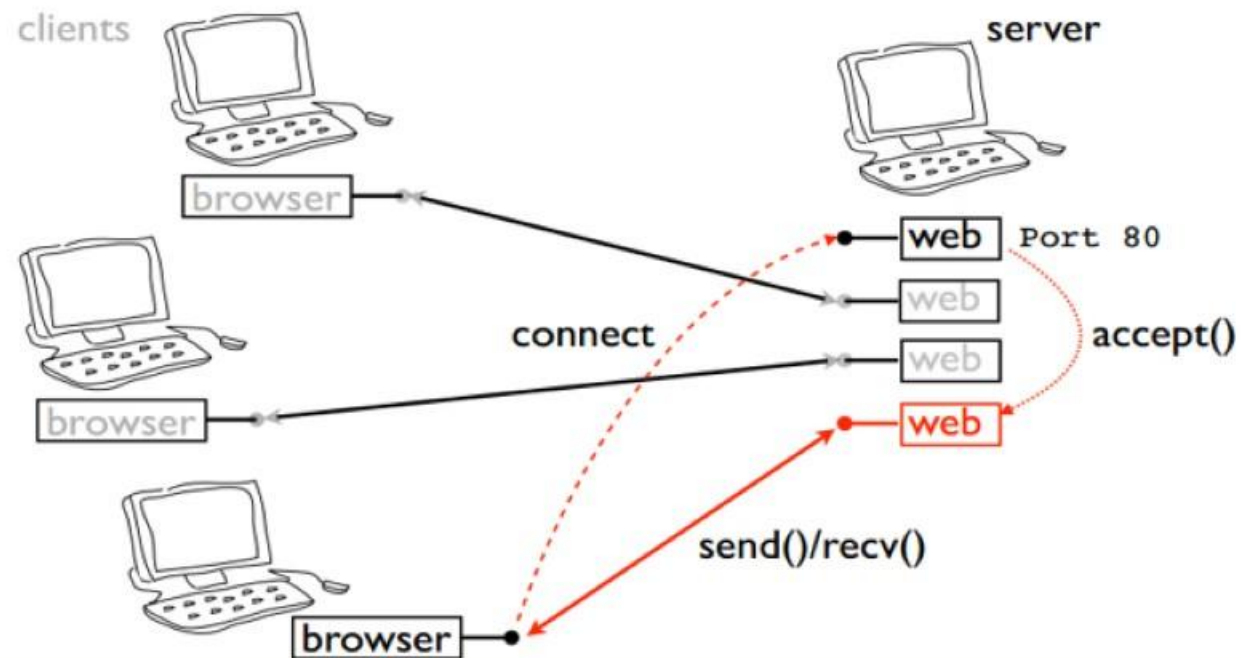
- Each client gets its own socket on server.



## SOCKETS AND CONCURRENCY

Cont'd

- New connections make a new socket.



## SOCKETS AND CONCURRENCY

Cont'd

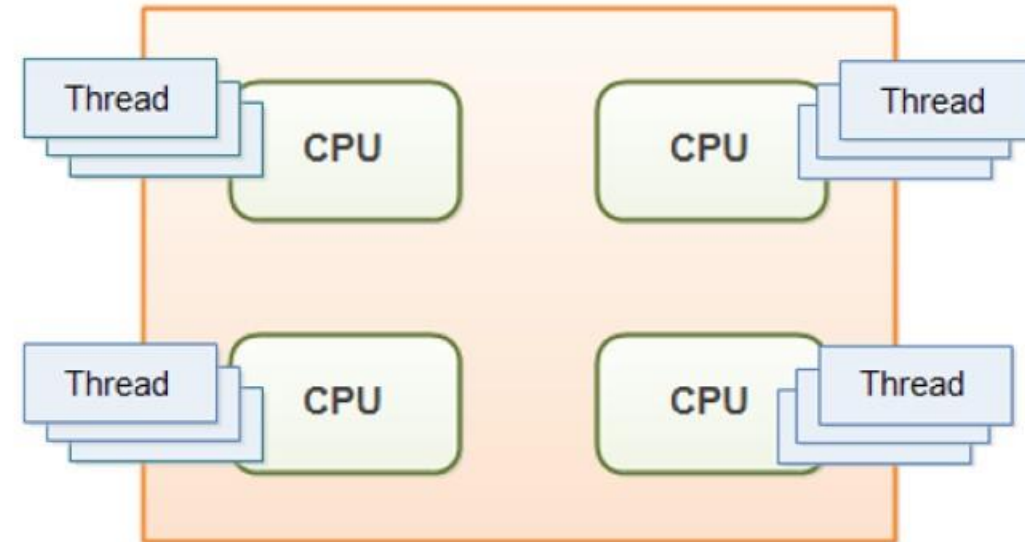
- **A concurrent server** accepts a client connection, delegates the connection to a child process of some kind, and immediately signals its availability to receive the next client connection.
- The following list describes the concurrent server process:
  - When a connection request arrives, it schedules a child process and forwards the connection to the child process.
  - The child process takes the connection from the main process.
  - The child process receives the client request and processes it.
  - The connection is closed, and the child process signals to the main process that it is available for a new connection.

## Lecture

## 05

## THE THREADING PROGRAMMING

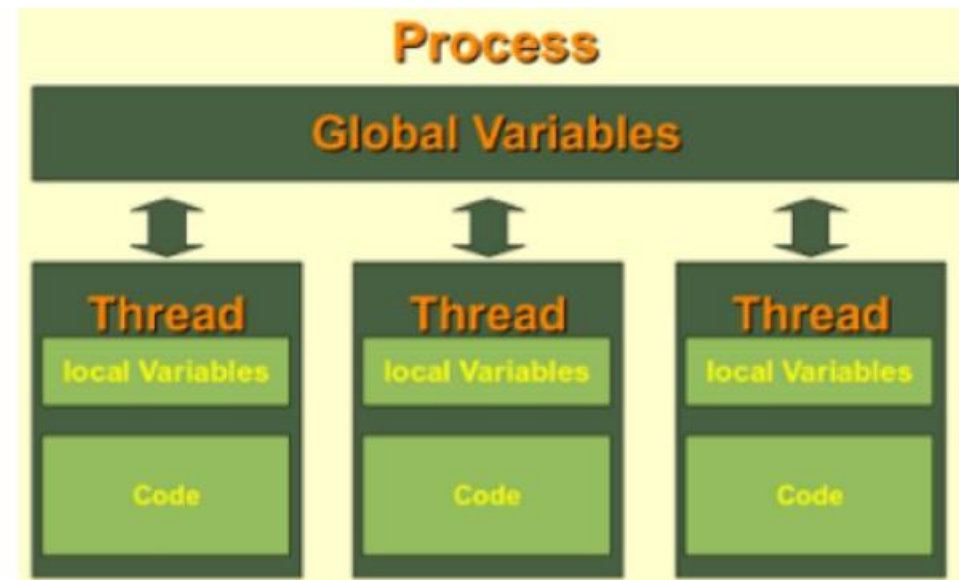
- A **thread** is defined in computer science as **the smallest unit** that can be scheduled in an operating system.
- The ability of a process to execute multiple threads parallelly is called **Multithreading**.



# THE THREADING PROGRAMMING

Cont'd

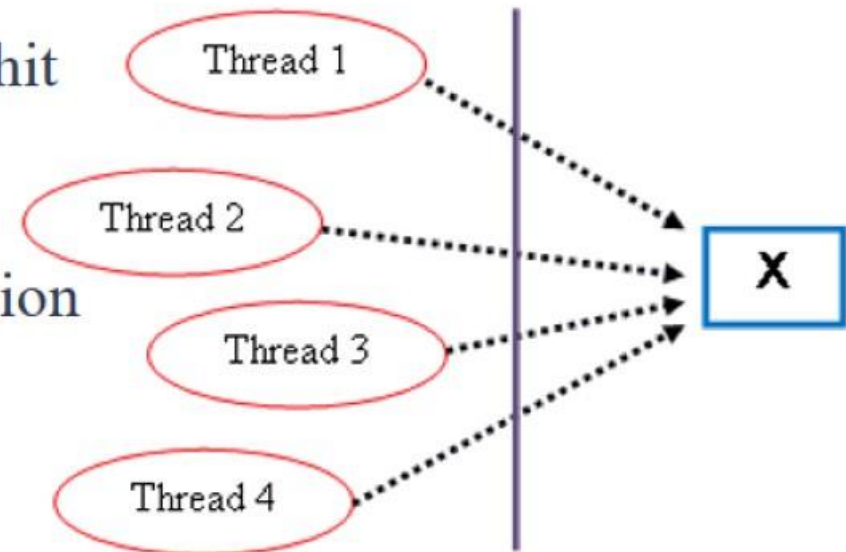
- Advantages:
  - Significantly improve the performance of any program.
  - A program can remain responsive while one thread waits for input.
  - All the threads of a process have access to its global variables.



# THE THREADING PROGRAMMING

Cont'd

- Disadvantages:
  - On 1-CPU system, multithreading would not hit the speed of computation.
  - Synchronization for preventing mutual exclusion of shared resources.
  - Increases the complexity of the program.
  - Thread safe problem.



## Lecture

## 05

THREADING IN PYTHON USING  
FUNCTIONS

- A function is a set of instructions that could be performed n times according to function calling in the code.

```
def myFun1(a, b):  
    c = a + b  
    print(c)
```

```
myFun1(1, 2)
```

# THREADING IN PYTHON USING FUNCTIONS

Cont'd

- Performing/calling a function using threads using the threading module in Python.

```
import threading
def myFun1(a, b):
    c = a + b
    print(c)
thread1 = threading.Thread(target = myFun1, args = (12, 10))
thread1.start()
```

# THREADING IN PYTHON USING FUNCTIONS

Cont'd

- Code:

```
import threading, time
def myFun1(a, b):
    time.sleep(1) #seconds
    c = a + b
    print(c)
def myFun2(a):
    c = a * 2
    print(c)
```

# THREADING IN PYTHON USING FUNCTIONS

Cont'd

- Code:

```
thread1 = threading.Thread(target = myFun1, args = (12, 10))  
thread1.start()
```

```
thread2 = threading.Thread(target = myFun2, args = (10,))  
thread2.start()  
print("Total number of threads", threading.active_count() - 1)
```

## THREADING METHODS

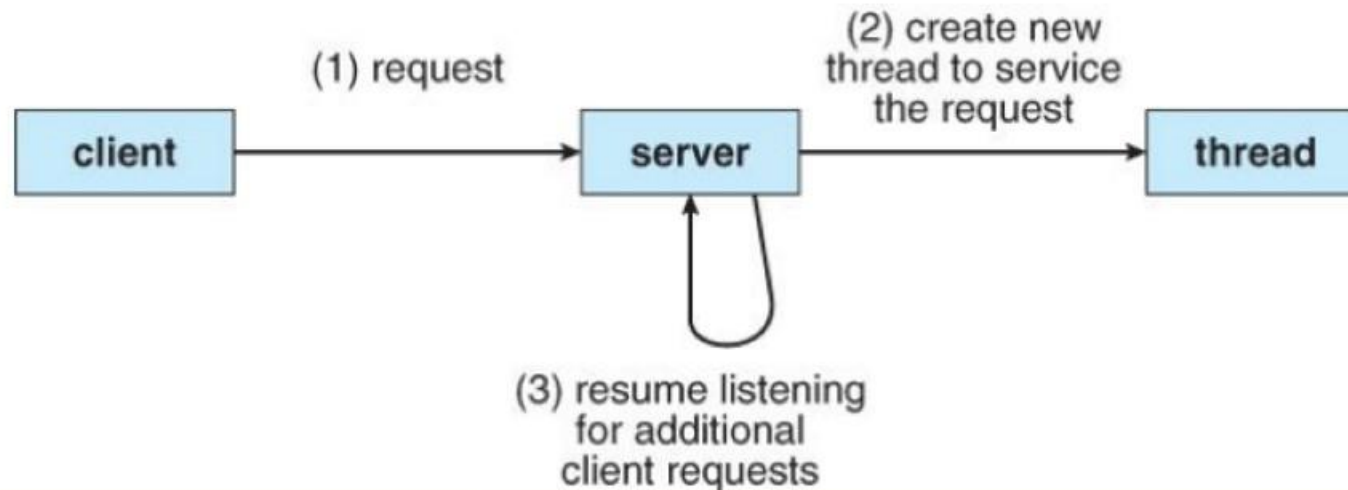
- **Thread()**: creates a thread object. It takes two argument as following:
  - **target**: refers to the function name assigned to the created thread.
  - **args**: a tuple represents the function inputs.
- **start()**: starts the execution of the thread in the operating system.
- **active\_count()**: counts the number of the threads in the whole program.
  - The program itself is counted as a thread, so we subtract 1 from the number returned from this method.

## Lecture

## 05

# THREADING FOR THE CLIENT/SERVER MODEL

- Thread-1: listen and accept connections from other clients.
- Thread-N: send commands to an already connected clients.



## Lecture

## 05

THREADING FOR THE  
CLIENT/SERVER MODEL

Cont'd

```
ng/My Lecs/W 6 - Done/codes/multiple connection/ser
ver.py"
127.0.1.1
The server is starting
The server is listening to 127.0.1.1
('127.0.0.1', 35204) is connected.
Number of active connection: 1
```

```
aming/My Lecs/W 6 - Done/codes/multiple connectio
n/client.py"
```

```
127.0.1.1
The server is starting
The server is listening to 127.0.1.1
('127.0.0.1', 35204) is connected.
Number of active connection: 1
('127.0.0.1', 35204) hi server
('127.0.0.1', 35204) Is there any one else?
```

```
hi server
Is there any one else?
```

## Lecture

## 05

THREADING FOR THE  
CLIENT/SERVER MODEL

Cont'd

```
127.0.1.1
The server is starting
The server is listening to 127.0.
1.1
('127.0.0.1', 35204) is connected
.
Number of active connection: 1
('127.0.0.1', 35204) hi server
('127.0.0.1', 35204) Is there any
one else?
('127.0.0.1', 35486) is connected
.
Number of active connection: 2
█
```

```
CI Courses/Online/Old/Network Pro
gramming/My Lecs/W 6 - Done/codes
/multiple connection/client.py"
█
```

```
rk Programming/My Lecs/W 6 - Don
e/codes/multiple connection/clie
nt.py"
hi server
Is there any one else?
█
```

## Lecture

## 05

THREADING FOR THE  
CLIENT/SERVER MODEL

Cont'd

```
127.0.1.1
The server is starting
The server is listening to 127.0.
1.1
('127.0.0.1', 35204) is connected
.
Number of active connection: 1
('127.0.0.1', 35204) hi server
('127.0.0.1', 35204) Is there any
one else?
('127.0.0.1', 35486) is connected
.
Number of active connection: 2
('127.0.0.1', 35486) hi from seco
nd client
█
```

```
n/p /bin/p
ython3 "/media/drhamada/FCI/FCI/F
CI Courses/Online/Old/Network Pro
gramming/My Lecs/W 6 - Done/codes
/multiple connection/client.py"
hi from second client
█
```

```
in/python3 "/media/drhamada/FCI/
FCI/FCI Courses/Online/Old/Netwo
rk Programming/My Lecs/W 6 - Don
e/codes/multiple connection/clie
nt.py"
hi server
Is there any one else?
Message received: hi from second
client
█
```

## THREADING FOR THE CLIENT/SERVER MODEL

Cont'd

- Code: 1- the server

```
import socket, threading
Port = 5050
Server = socket.gethostbyname(socket.gethostname())
Addr = (Server, Port)
server = socket.socket(socket.AF_INET, socket.SOCK_STREAM)
server.bind(Addr)

def send(msg, client):
    client.send(msg.encode("utf-8"))
```

## THREADING FOR THE CLIENT/SERVER MODEL

Cont'd

- Code: 1- the server

```
def client(conn, addr, clients):  
    print(addr, "is connected.")  
    connected = True  
    while connected:  
        msg = conn.recv(1024).decode("utf-8")  
        if msg == "DES":  
            connected = False  
            continue  
        print(addr, msg)
```

## THREADING FOR THE CLIENT/SERVER MODEL

Cont'd

- Code: 1- the server

```
def client(conn, addr, clients):  
    .....  
    while connected:  
        .....  
        msg = f"Message received: {msg}"  
        for client in clients:  
            if client != conn:  
                send(msg, client)  
conn.close()
```

## THREADING FOR THE CLIENT/SERVER MODEL

Cont'd

- Code: 1- the server

```
def start():  
    clients = []  
    server.listen()  
    print("The server is listening to", Server)  
    while True:  
        conn, addr = server.accept()  
        clients.append(conn)
```

## THREADING FOR THE CLIENT/SERVER MODEL

Cont'd

- Code: 1- the server

```
def start():  
    .....  
    while True:  
        .....  
        thread = threading.Thread(target=client, args=(conn, addr, clients))  
        thread.start()  
        print("Number of active connection:", threading.active_count() - 1)  
  
print("The server is starting")  
start()
```

## THREADING FOR THE CLIENT/SERVER MODEL

Cont'd

- Code: 2- the client

```
import threading, socket
Port = 5050
Server = socket.gethostbyname(socket.gethostname())
Addr = (Server, Port)
client = socket.socket(socket.AF_INET, socket.SOCK_STREAM)
client.connect(Addr)
def read():
    msg = client.recv(1024).decode("utf-8")
    print(msg)
```

## THREADING FOR THE CLIENT/SERVER MODEL

Cont'd

- Code: 2- the client

```
def send():  
    connect = True  
    while connect:  
        msg = input();    client.send(msg.encode("utf-8"))  
        if msg == "DES": connect = False  
        else:  
            thread = threading.Thread(target=read);    thread.start()  
            continue
```

**send()**

## Lecture

## 05

## CASTING

- Many applications transmit the same data at one time to multiple receivers:
  - Broadcasts of Radio or Video.
  - Video conferencing.
  - Shared applications.
- **The cast** term signifies some data (stream of packets) is being transmitted to the client(s) from a server over the communication channel.

Lecture

05

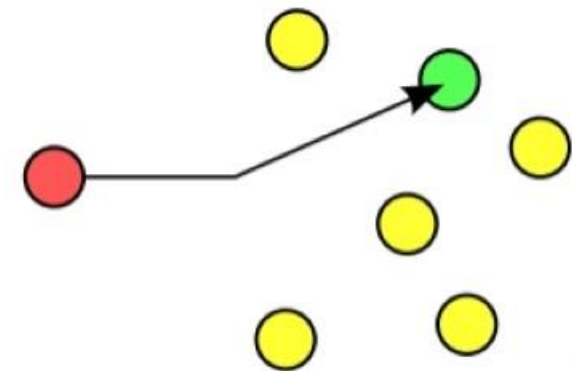
## CASTING/COMMUNICATION TYPES

- There are three casting types:
  - Unicasting.
  - Multicasting.
  - Broadcasting.

# CASTING/COMMUNICATION TYPES

Cont'd

- **Unicasting.**
  - A type of information transfer and it is used when there is a participation of single sender and single recipient.
  - Represents the one-to-one communication.
  - Example:
    - Browsing a website.
    - Downloading a file from a FTP server

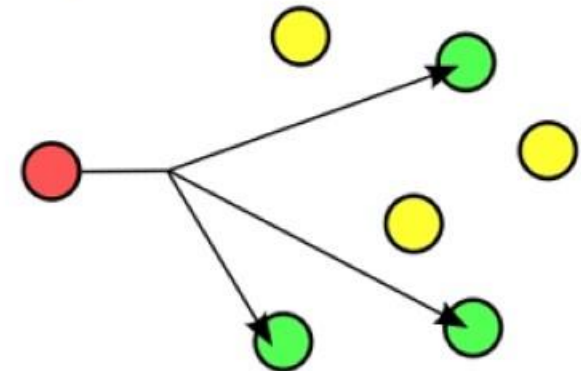


# CASTING/COMMUNICATION TYPES

Cont'd

- **Multicasting.**

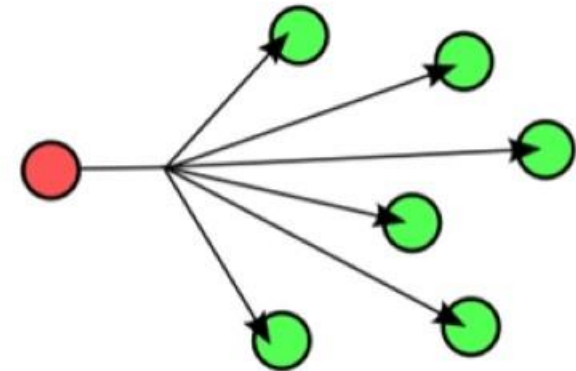
- A type of information transfer and it is used when there is a participation of single sender and multiple recipient.
- Represents the **one-to-many** communication.
- The multiple recipient must join to that a **multicast group** to receive data.
- Only supported by UDP.
- Example:
  - Sending an email to a mailing list.



# CASTING/COMMUNICATION TYPES

Cont'd

- **Broadcasting.**
  - A type of information transfer and it is used when there is a participation of single sender and all recipient.
  - Represents the **one-to-all** communication.
  - The recipients should be in a **local network**.
  - Example:
    - Video and audio broadcasting.



Lecture

05

## QUESTIONS FOR ONLINE DISCUSSION

- Write a program to simulate an arithmetic competition, where any client can ask other clients to solve a simple arithmetic equation, and they can send their results. The client who send the question then say the client's address with the correct answer.



**Thank You**

