Client-Server Model

- Server
  - well-known name
  - waits for contact
  - process requests, sends replies

- Client
  - initiate contact
  - wait for response
Socket Programming

• Socket
  – A method for communication between a server program and a client program
  – Bidirectional communication between processes
  – Processes could be in the same/different computer(s)
Client-Server Communications

Diagram showing the process of client-server communication with functions such as `socket()`, `bind()`, `listen()`, `accept()`, `recv()`, `send()`, and `close()`.
Server

• `socket(domain, type)`
  – create a socket

• `bind((host, port))`
  – associate the socket with a port in the host computer

• `listen(n)`
  – size of queue for waiting clients who want to get connection

• `accept()`
  – wait and accept client’s call
Server(2)

• `recv(bufsize)`
  – load the data from socket

• `send(string)`
  – write data(string) in socket and send to client

• `close()`
  – close socket
Client

• socket(domain, type)
  – create a socket
• connect((host,port))
  – Try to connect to defined address by socket
• send(string)
  – write data(string) in socket and send to server
• close()
  – close socket
import socket

HOST = ''
PORT = 50007
s = socket.socket(socket.AF_INET, socket.SOCK_STREAM)
s.bind((HOST, PORT))
s.listen(1)
conn, addr = s.accept()
print('Connected by', addr)
while True:
    data = conn.recv(1024).decode('UTF-8')
    if not data:
        break
    print(data)
    conn.send(bytes(data, 'UTF-8'))
conn.close()
import socket

HOST = 'localhost'
PORT = 50007

s = socket.socket(socket.AF_INET, socket.SOCK_STREAM)
s.connect((HOST, PORT))
s.send(bytes("Hello World","UTF-8"))
data = s.recv(1024).decode('UTF-8')
s.close()
print('Received', data)