Sistemas Distribuidos

Sockets en Python

Sara Balderas Díaz
Gabriel Guerrero Contreras

Versión 2.0

Grado en Ingeniería Informática
Departamento de Ingeniería Informática
Universidad de Cádiz

Índice

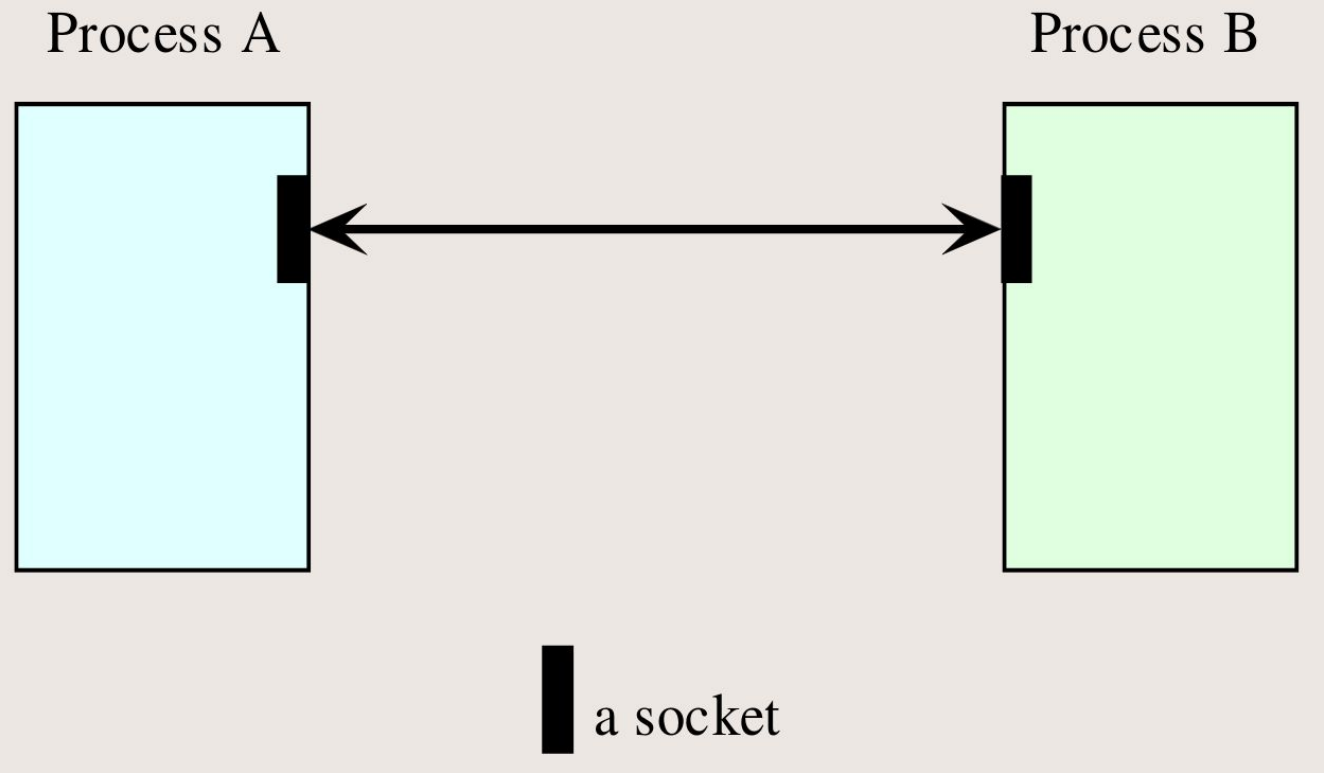
1. Sockets (I)

¿Qué son?

- Un socket es una interfaz de entrada-salida que permite la comunicación entre procesos.
- Application Programming Interface (API), publicada en 1983 en la Distribución de Software de Berkeley (Berkeley Software Distribution, BSD) y denominada Berkeley Sockets.
- Se utilizan en Internet Sockets y Unix Domain Sockets, para la comunicación entre procesos (Interprocess Communication, IPC).
- En la década de 1990 se generalizaron las aplicaciones cliente-servidor.
- Implementada en prácticamente todos los SO (es un estándar de facto).
- La API de bajo nivel se ha mantenido durante estos años.
- Es la base para construir cosas más complejas.

1. Sockets (II)

Modelo Conceptual



1. Sockets (III)

API Socket

- Si se actúa como servidor:
  - Dirección IP
  - Puerto (los puertos <1024 están reservados)
- Protocolos:
  - Transmission Control Protocol (TCP): Socket Stream. Orientado a conexión. Se garantiza que todos los paquetes llegan ordenados.
  - User Datagram Protocol (UDP): Socket Datagram. No orientado a conexión. Los paquetes pueden llegar desordenados o perderse.
- Listar puertos:
  - `Todos los puertos: $netstat -a`
  - `Puertos TCP: $netstat -at`
  - `Puertos UDP: $netstat -au`

1. Sockets (IV)

El módulo Socket de Python

```
import socket

s = socket.socket(socket_family, socket_type)
```

- socket_family: AF_UNIX o AF_INET
- socket_type: SOCK_STREAM (TCP) o SOCK_DGRAM (UDP)

2. Sockets UDP (I)

Métodos para los sockets UDP

- Enviar un mensaje, especificar anfitrión (host) y puerto:

  ```
  s.sendto(mensaje, (HOST, PUERTO))
  ```

  El mensaje puede perderse.

- Recibir un mensaje:

  ```
  mensaje = s.recvfrom(buffersize)
  ```

  Es bloqueante, se queda parado hasta recibir algo.

  Devuelve un mensaje recibido de cualquier máquina.

2. Sockets UDP (II)

Servidor UDP

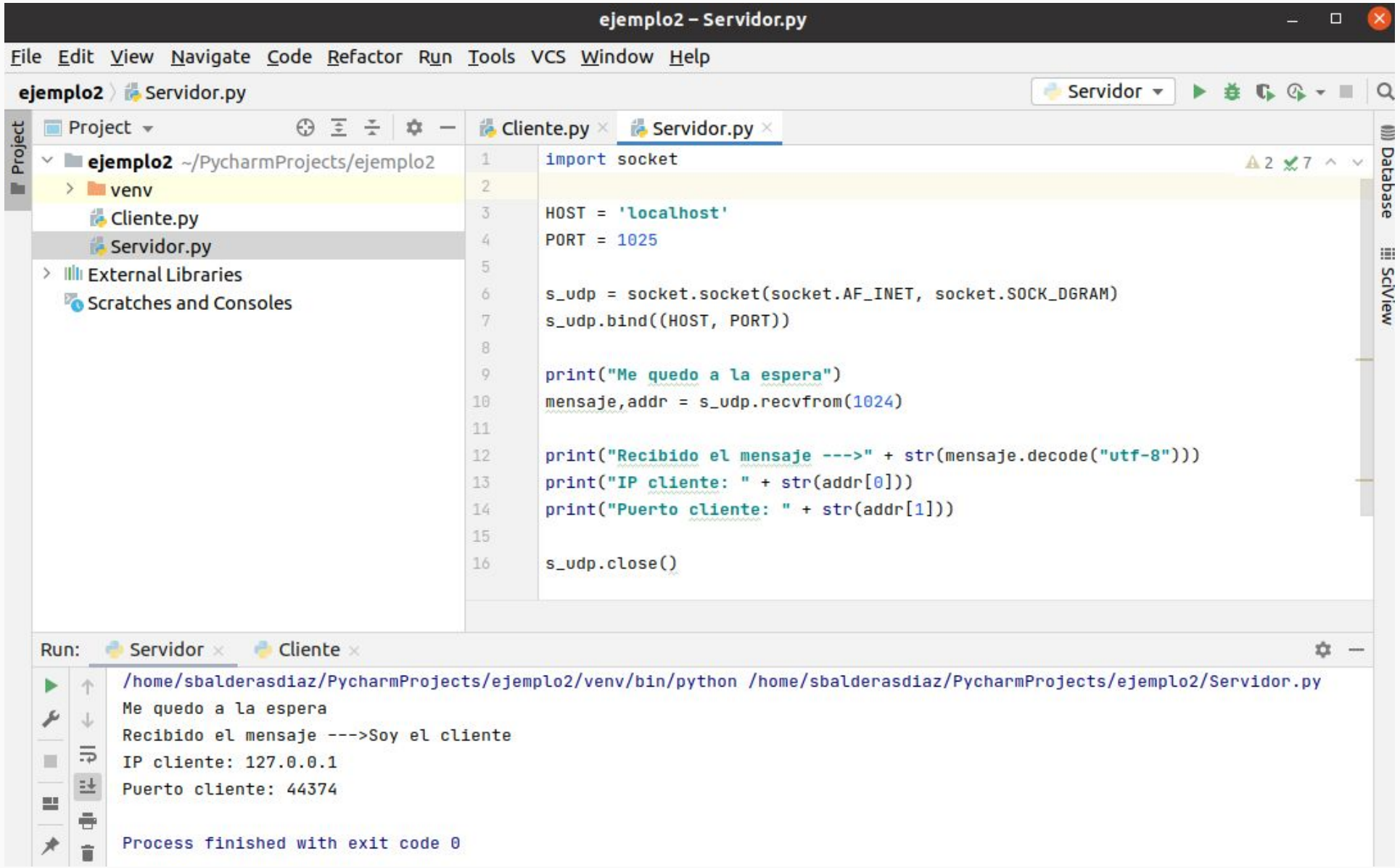
```
import socket

HOST = 'localhost'
PORT = 1025

s_udp = socket.socket(socket.AF_INET, socket.SOCK_DGRAM)
s_udp.bind((HOST, PORT))

print("Me quedo a la espera")
mensaje,addr = s_udp.recvfrom(1024)

print("Recibido el mensaje --->" + str(mensaje.decode("utf-8")))
print("IP cliente: " + str(addr[0]))
print("Puerto cliente: " + str(addr[1]))

s_udp.close()
```

2. Sockets UDP (III)

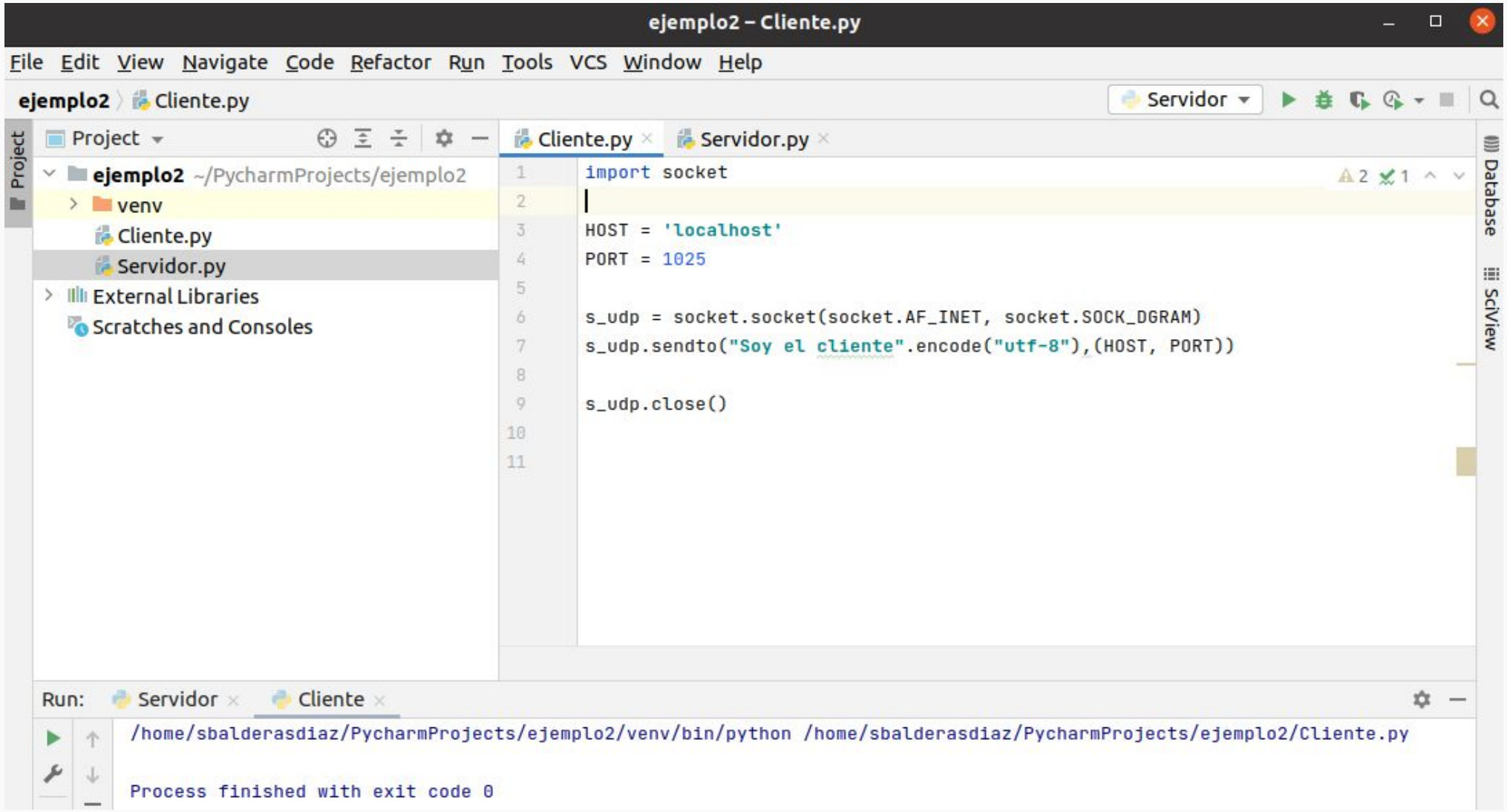
Cliente UDP

```
import socket

HOST = 'localhost'
PORT = 1025

s_udp = socket.socket(socket.AF_INET, socket.SOCK_DGRAM)
s_udp.sendto("Soy el cliente".encode("utf-8"),(HOST, PORT))

s_udp.close()
```

2. Sockets UDP (IV)

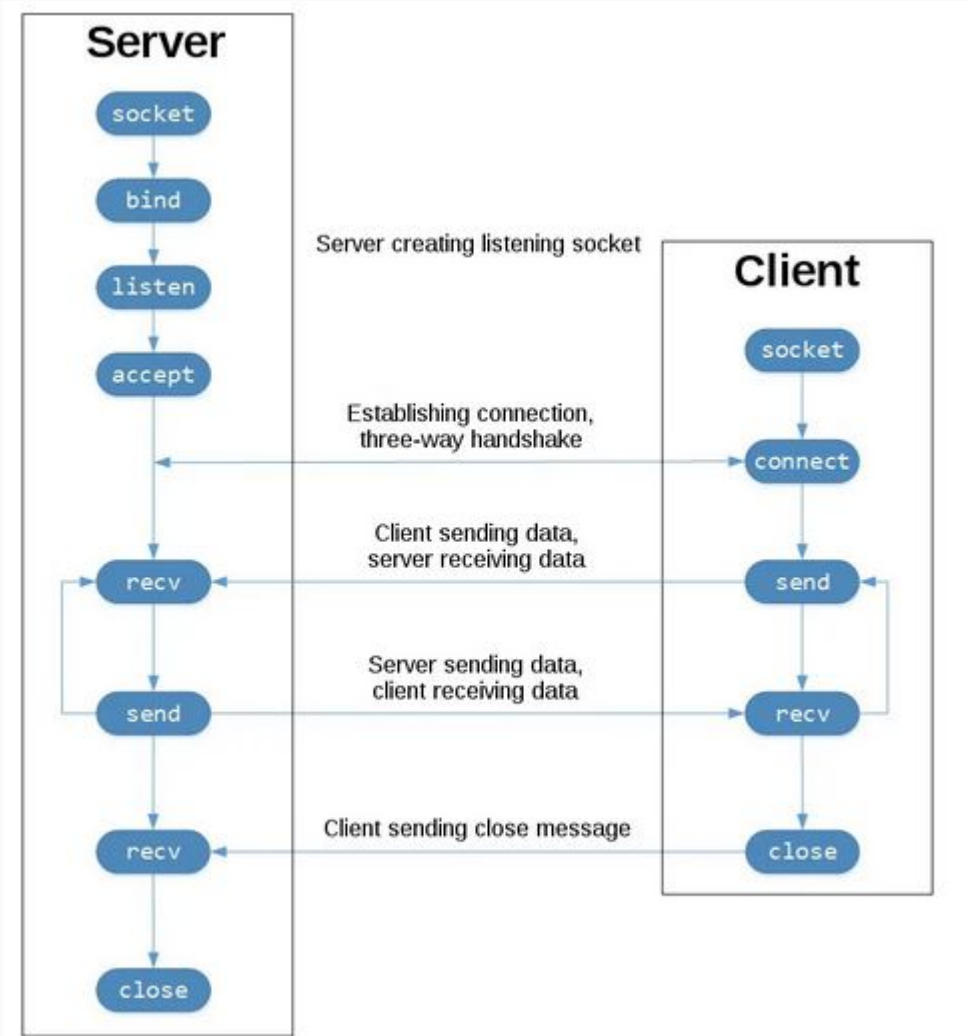


2. Sockets UDP (V)



3. Sockets TCP (I)

Flujo socket orientado a conexión



3. Sockets TCP (II)

Métodos para los Servidores TCP

- Se le asigna IP y puerto al socket que estará a la espera:

```
s.bind((HOST, PORT))
```

- Prepara el *listener* para aceptar clientes:

```
s.listen()
```

- Se queda bloqueado hasta que un cliente se conecta. Devuelve un nuevo socket para comunicarse con el cliente.

```
s_cliente,addr = s.accept()
```

3. Sockets TCP (III)

Métodos para los Clientes TCP

- Se conecta a un servidor que esté aceptando conexiones. Si no existe, da un error.

```
s.connect((HOST, PORT))
```

3. Sockets TCP (IV)

Métodos para los sockets TCP (servidor o cliente)

- Recibe un mensaje del socket al que está conectado (particionado en 1024 bytes):

```
mensaje = s.recv(1024)
```

- Envía un mensaje por el socket:

```
s.send(mensaje)
```

3. Sockets TCP (V)

Servidor TCP

```
import socket

HOST = 'localhost'
PORT = 1024

socketServidor = socket.socket(socket.AF_INET, socket.SOCK_STREAM)

socketServidor.bind((HOST, PORT))
socketServidor.listen(1)
print("Nos quedamos a la espera...")
s_cliente, addr = socketServidor.accept()

mensaje = s_cliente.recv(1024)
print("Recibo:["+mensaje.decode("utf-8")+"] del cliente con la direccion " + str(addr))

s_cliente.send("Hola, cliente, soy el servidor".encode("utf-8"))
s_cliente.close()

socketServidor.close()
```

3. Sockets TCP (VI)

Cliente TCP

```
import socket

HOST = 'localhost'
PORT = 1024

s = socket.socket(socket.AF_INET, socket.SOCK_STREAM)
s.connect((HOST, PORT))

s.send("Hola, servidor".encode("utf-8"))
mensaje = s.recv(1024)
print("Recibido:["+mensaje.decode("utf-8")+"] del servidor ")

s.close()
```

3. Sockets TCP (VII)

Encodings de Python 3

```
#ENVIAR MENSAJE
s.send(bytes("Hey there!!!", "utf-8"))
#IMPRIMIR MENSAJE RECIBIDO
print(msg.decode("utf-8"))
```

Bibliografía

- Socket - Low-level networking interface. https://docs.python.org/3.6/library/socket.html
- Socket Programming HOWTO. https://docs.python.org/es/3.6/howto/sockets.html
- Python Tutorial. https://www.tutorialspoint.com/python/

Agradecimientos

Por la elaboración de la versión original (versión 1.0) de este seminario:

- Pablo García

Preguntas