Branch: **master** **hackerU_python** / lessonPlan / **lesson5_sockets.md** Find file Copy path

t-0-m-1-3 added folders 190da8f on Jul 30, 2018

1 contributor

247 lines (185 sloc) 7.19 KB Raw Blame History

The Socket Module

Communication is a key component of being a human being. In the modern environment we have a large amount of communication methods at our disposal. We've come a long way from email and IRC channels, but the core components of sending messages to your friends remains generally the same.

Computers were originally isolated from each other, and were later connected together to share information between each other. We will learn how to lay that framework here.

Server-Client:

There are a large number of forms of communication, but the main and most common form is called **"Server-Client"**. With this form, the computer `server` that provides a service to a requester called a `client`

To implment client-server communication, we need to know what a **socket** is, a socket is the end point of connecting two components. I we want to transfer information between two components, we need sockets to link them.

In order to construct this linked data pipeline we need a few things: 1. **Build the pipieline** - done by both the client and the server 2. **Start-and-end OR Source-and-destination** - defined by address and port 3. **Size** - each pipe has a diameter and a length, same as with sockets. How much information can be transmitted at any given time is a crucial piece of information.

Build a client:

First take a look at the socket library using `help('socket')`

You can see stuff and things

The first thing to do is to create a socket object and apply a variable to it. This way we can call the methods attached to the object using the variable name. The example below is a general form.

```
import socket
SocketVariable = socket.socket()
```

The function needs to send a tuple with the `ip address` and the `port` as parameters `SocketVariable = socket.socket("ip", port)`

If the connection is confirmed, a complet socket will be created that can transfer the information between the server and the client.

To Send information through the connection `SocketVariable.recv(buffer)` is used.

The buffer represents the diameter of the pipeline we create, it's the size of information we can send in a single pulse. This data can be saved into variables and manipulated.

Just like with files, it is prudent to close the socket when it is not being used. `SocketVariable.close()`

```
import socket

S = socket.socket()
S.connect(("1.1.1.1", 80))
S.send("hi")
Data = S.recv(2048)
print Data
S.close()
```

This will send information, receive information, and manipulate it at will.

How to Build a Server:

```
import socket

SocketVariable = socket.socket()
```

This is the starting point for our server. We've imported and instantiated a socket. Next we need to link out socket to an object that represents out local address. This is the address people will "call" when requesting information from us on a port.

`SocketVariable.bind("ip",port))`

If you want to connect to anyone who wants information from you, use the address `0.0.0.0` , the representation for the general network name.

Next, the server needs to listen for anyone requesting information from it. `SocketVariable.listen(socketNumberOfConnection)`

The line above lets us listen for incoming connections, the next step is to handle them as the come.

`clientSocket, ClientAddress = SocketVariable.accpet()`

the accept command accepts tuple that will freeze execution until a new connection is received on the server. The two variables are broken down from a general connection into a speicific connection. The ClientSocket variable will represent the specific connection to the client.

Any information that related to a specific client will be made passible through the client's object rather that the servers general object

```
Data = ClientSocket.recv(2048)

#I can send info to you
ClientSocket.send("some information")
#After sending and receiving information, close the conne
ClientSocket.close()
# Close the servers connection tool
ServerSocket.close()
```

putting all the code together for the server

```
S = socket.socket()
S.bind((0.0.0.0,1234)
S.listen(1)
C, ADDRESS = S.accept()
Data = C.recv(2048)
print "the server send the data: ", Data
```

```
C.Send("hi")
C.close()
S.close()
```

Exercise

1. Create an echo server. Any information sent to it, the server will echo back

```
import socket
S = socket.socket()
S.bind(("0.0.0.0", 80))
S.listen(1)
C, Address = S.accept()
Name = C.recv(2048)
C.close()
S.close()
```

```
import socket

sock = socket.socket(socket.AF_INET, socket.SOCK_DGRAM)

server_address = '0.0.0.0'
server_port = 31337

server = (server_address, server_port)
sock.bind(server)
print("Listening on " + server_address + ":" + str(server

while True:
	payload, client_address = sock.recvfrom(1)
	print("Echoing data back to " + str(client_address
sent = sock.sendto(payload, client_address)
```

2. Create a server that will take in the users name and send the user a message "Hi User".

```
import socket
S = socket.socket()
S.bind(("0.0.0.0",80))
S.listen(1)
C, Address = S.accept()
Name = C.recv(2048)
C.send("hi " + Name)
C.close()
S.cLose()
```

3. Create an echo server that receives a user name and a password from the user. If they match a predefined name and password, send the user a personalized greeting.

```
import socket
S = socket.socket()
S.bind(("0.0.0.0",80))
S.listen(1)
C,Address = S.Accept()
USER = C.recv(2048)
PASS = C.recv(2048)
if USER == "admin" and PASS == "12345":
    C.send("hello admin, welcome to the server")
C.close()
S.close()
```

4. Create a basic command server that received four commands from the user and sends a custom response to the request.

- **Random** if the input is `R` send back a random number between 1 and 10
- **Time** if the input is `T` send back the system time
- **Language** if the input is `L` the server will send the version info for python
- **Exit** if the input is `E` close the connection and quit the program

```
import socket
import random
import datetime
import sys

def rand():
    number = random.randint(1,10)
    return number

def timeof():
    return str(datetime.datetime.now())

def lang():
    return "English"

def exitfrom():
    sys.exit()
S = socket.socket()
S.bind("0.0.0.0", 80)
S.listen(1)
C, Address = S.accept()
Choice = C.recv(2048)
if Choice == "r"
    C.send(rand())
elif choice == "T":
    C.send(timeof())
elif choice == "L":
    C.send(lang())
elif choice == "E":
    C.sen(exitfrom())
else:
    print "wrong option selected"
    sys.exit()
C.close()
S.close()
```

5. Create a UDP Server.

```
import logging
import socket

log = logging.getLogger('udp_server')
```

```
def udp_server(host='127.0.0.1', port=1234):
    s = socket.socket(socket.AF_INET, socket.SOCK_DGRAM)
    s.setsockopt(socket.SOL_SOCKET, socket.SO_REUSEADDR,

    log.info("Listening on udp %s:%s" % (host, port))
    s.bind((host, port))
    while True:
        (data, addr) = s.recvfrom(128*1024)
        yield data


FORMAT_CONS = '%(asctime)s %(name)-12s %(levelname)8s\t%(
logging.basicConfig(level=logging.DEBUG, format=FORMAT_CO

for data in udp_server():
log.debug("%r" % (data,))
```