Python Networking

All Versions

Examples

Creating a Simple Http Server

To share files or to host simple websites(http and javascript) in your local network, you can use Python's builtin SimpleHTTPServer module. Python should be in your Path variable. Go to the folder where your files are and type:

For `python 2` :

```
$ python -m SimpleHTTPServer <portnumber>
```

For `python 3` :

```
$ python3 -m http.server <portnumber>
```

If port number is not given `8000` is the default port. So the output will be:

> Serving HTTP on 0.0.0.0 port 8000 ...

You can access to your files through any device connected to the local network by typing `http://hostipaddress:8000/` .

`hostipaddress` is your local ip address which probably starts with `192.168.x.x` .

To finish the module simply press `ctrl+c`.

Creating a TCP server

You can create a TCP server using the `socketserver` library. Here's a simple echo server.

Server side

```
from sockerserver import BaseRequestHandler, TCPServer

class EchoHandler(BaseRequestHandler):
    def handle(self):
        print('connection from:', self.client_address)
        while True:
            msg = self.request.recv(8192)
            if not msg:
                break
            self.request.send(msg)

if __name__ == '__main__':
    server = TCPServer(('', 5000), EchoHandler)
    server.serve_forever()
```

Client side

```
from socket import socket, AF_INET, SOCK_STREAM
sock = socket(AF_INET, SOCK_STREAM)
sock.connect(('localhost', 5000))
sock.send(b'Monty Python')
sock.recv(8192)  # returns b'Monty Python'
```

`socketserver` makes it relatively easy to create simple TCP servers. However, you should be aware that, by default, the servers are single threaded and can only serve one client at a time. If you want to handle multiple clients, either instantiate a `ThreadingTCPServer` instead.

```
from socketserver import ThreadingTCPServer
...
if __name__ == '__main__':
    server = ThreadingTCPServer(('', 5000), EchoHandler)
    server.serve_forever()
```

Creating a UDP Server

A UDP server is easily created using the `socketserver` library

A UDP server is easily created using the socketserver library.

a simple time server:

```
import time
from socketserver import BaseRequestHandler, UDPServer

class CtimeHandler(BaseRequestHandler):
    def handle(self):
    print('connection from: ', self.client_address)
    # Get message and client socket
    msg, sock = self.request
    resp = time.ctime()
    sock.sendto(resp.encode('ascii'), self.client_address)

if __name__ == '__main__':
    server = UDPServer(('', 5000), CtimeHandler)
    server.serve_forever()
```

Testing:

```
>>> from socket import socket, AF_INET, SOCK_DGRAM
>>> sock = socket(AF_INET, SOCK_DGRAM)
>>> sick.sendto(b'', ('localhost', 5000))
0
>>> sock.recvfrom(8192)
(b'Wed Aug 15 20:35:08 2012', ('127.0.0.1', 5000))
```

Start Simple HttpServer in a thread and open the browser

Useful if your program is outputting web pages along the way.

```
from http.server import HTTPServer, CGIHTTPRequestHandler
import webbrowser
import threading

def start_server(path, port=8000):
    '''Start a simple webserver serving path on port'''
    os.chdir(path)
    httpd = HTTPServer(('', port), CGIHTTPRequestHandler)
    httpd.serve_forever()

# Start the server in a new thread
port = 8000
daemon = threading.Thread(name='daemon_server',
                          target=start_server,
                          args=('.', port)
daemon.setDaemon(True) # Set as a daemon so it will be killed once the main thread is dead.
daemon.start()

# Open the web browser
webbrowser.open('http://localhost:{}'.format(port))
```

The simplest Python socket client-server example

Server side:

```
import socket

serversocket = socket.socket(socket.AF_INET, socket.SOCK_STREAM)
serversocket.bind(('localhost', 8089))
serversocket.listen(5) # become a server socket, maximum 5 connections

while True:
    connection, address = serversocket.accept()
    buf = connection.recv(64)
    if len(buf) > 0:
        print(buf)
    break
```

Client Side:

```
import socket

clientsocket = socket.socket(socket.AF_INET, socket.SOCK_STREAM)
clientsocket.connect(('localhost', 8089))
clientsocket.send('hello')
```

First run the SocketServer.py, and make sure the server is ready to listen/receive sth Then the client send

info to the server; After the server received sth, it terminates

Syntax

Parameters

Remarks

(Very) basic Python client socket example