'''

Copyright (C) 2014-2016 ddurdle

This program is free software: you can redistribute it and/or modify

it under the terms of the GNU General Public License as published by

the Free Software Foundation, either version 3 of the License, or

(at your option) any later version.

This program is distributed in the hope that it will be useful,

but WITHOUT ANY WARRANTY; without even the implied warranty of

MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE. See the

GNU General Public License for more details.

You should have received a copy of the GNU General Public License

along with this program. If not, see <http://www.gnu.org/licenses/>.

'''

import socket

import select

import time

import sys

buffer\_size = 4096

delay = 0.0001

forward\_to = ('dmdsoftware.net', 80)

class Forward:

def \_\_init\_\_(self):

self.forward = socket.socket(socket.AF\_INET, socket.SOCK\_STREAM)

def start(self, host, port):

try:

self.forward.connect((host, port))

return self.forward

except Exception, e:

print e

return False

#

# This class is contains code contributed from http://voorloopnul.com/blog/a-python-proxy-in-less-than-100-lines-of-code/

#

class proxy:

input\_list = []

channel = {}

def \_\_init\_\_(self, host, port):

self.server = socket.socket(socket.AF\_INET, socket.SOCK\_STREAM)

self.server.setsockopt(socket.SOL\_SOCKET, socket.SO\_REUSEADDR, 1)

self.server.bind((host, port))

self.server.listen(200)

def main\_loop(self):

self.input\_list.append(self.server)

while 1:

time.sleep(delay)

ss = select.select

inputready, outputready, exceptready = ss(self.input\_list, [], [])

for self.s in inputready:

if self.s == self.server:

self.on\_accept()

break

self.data = self.s.recv(buffer\_size)

if len(self.data) == 0:

self.on\_close()

break

else:

self.on\_recv()

def on\_accept(self):

forward = Forward().start(forward\_to[0], forward\_to[1])

clientsock, clientaddr = self.server.accept()

if forward:

print clientaddr, "has connected"

self.input\_list.append(clientsock)

self.input\_list.append(forward)

self.channel[clientsock] = forward

self.channel[forward] = clientsock

else:

print "Can't establish connection with remote server.",

print "Closing connection with client side", clientaddr

clientsock.close()

def on\_close(self):

print self.s.getpeername(), "has disconnected"

#remove objects from input\_list

self.input\_list.remove(self.s)

self.input\_list.remove(self.channel[self.s])

out = self.channel[self.s]

# close the connection with client

self.channel[out].close() # equivalent to do self.s.close()

# close the connection with remote server

self.channel[self.s].close()

# delete both objects from channel dict

del self.channel[out]

del self.channel[self.s]

def on\_recv(self):

data = self.data

# here we can parse and/or modify the data before send forward

print data

self.channel[self.s].send(data)