**Lab Report No: 02**

**Lab Report Name: Socket Programming**

**Name: S.M.SANOWER HOSSAIN**

**ID: IT-16047**

**Objective**

Socket programming shows how to use socket APIs to establish communication links between remote and local processes. The processes that use a socket can reside on the same system or different systems on different networks. Sockets are useful for both stand-alone and  
network applications.  
The processes that use a socket can reside on the same system or different systems on different networks. Sockets are useful for both stand-alone and network applications. Sockets allow you to exchange information between processes on the same machine or across a network, distribute work to the most efficient machine, and they easily allow access to centralized data. Socket application program interfaces (APIs) are the network standard for TCP/IP. A wide range of  
operating systems support socket APIs. i5/OS™ sockets support multiple transport and networking protocols. Socket system functions and the socket network functions are threadsafe.

**Connecting to a server :**

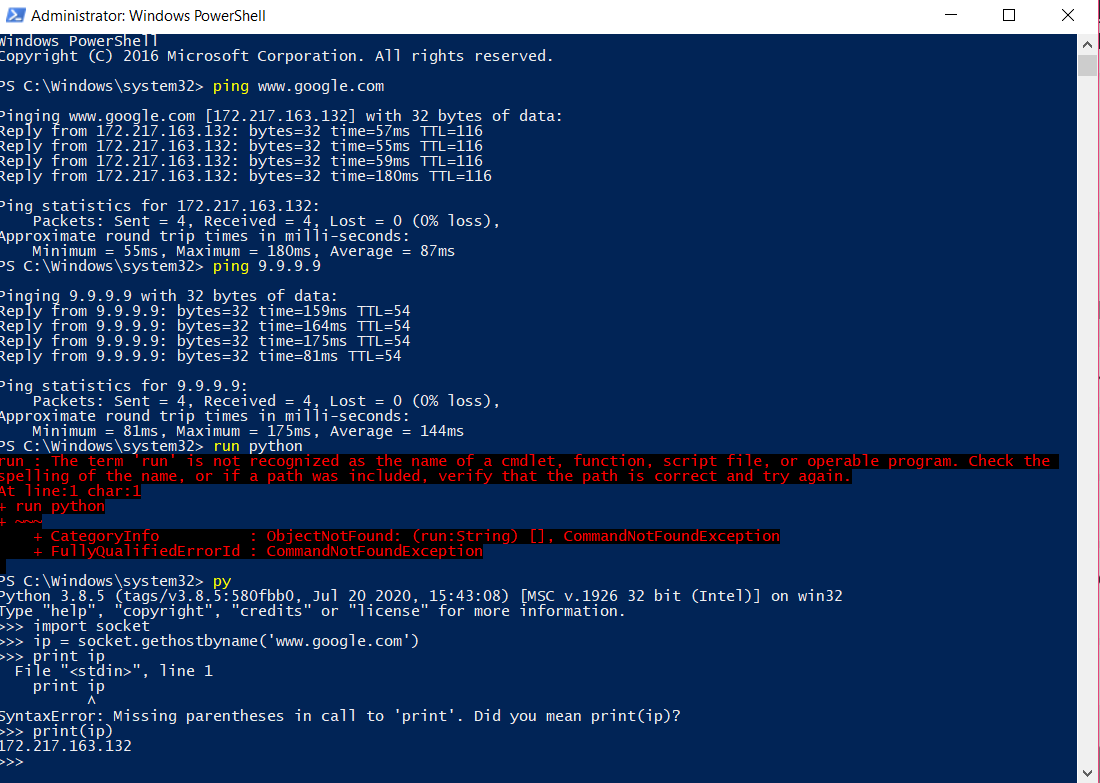
We can find ip address using ping command or using python. To find ip for google from command we use ‘ping [www.google.com](http://www.google.com)’

And for python

import socket

ip = socket.gethostbyname('www.google.com')

print ip



**Server:**

A server has a bind() method which binds it to a specific ip and port so that it can listen to incoming requests on that ip and port.A server has a listen() method which puts the server into listen mode. This allows the server to listen to incoming connections. And last a server has an accept() and close() method. The accept method initiates a connection with the client and the close method closes the connection with the client.

The below script is used for connecting to google. Python has a robust library for socket programming.

I named the file socket.py for interpreting it using command prompt.

**Socket.py**

import socket # for socket

import sys

try:

s = socket.socket(socket.AF\_INET, socket.SOCK\_STREAM)

print "Socket successfully created"

except socket.error as err:

print "socket creation failed with error %s" %(err)

# default port for socket

port = 80

try:

host\_ip = socket.gethostbyname('www.google.com')

except socket.gaierror:

# this means could not resolve the host

print "there was an error resolving the host"

sys.exit()

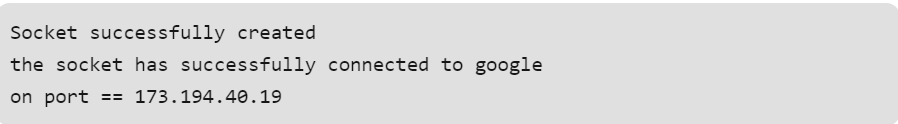
# connecting to the server

s.connect((host\_ip, port))

print "the socket has successfully connected to google \

on port == %s" %(host\_ip)

**Output:**



**Client:**

Now we need something with which a server can interact. We could tenet to the server like this just to know that our server is working. Type these commands in the terminal:

**Client.py**

import socket

# Create a socket object

s = socket.socket()

# Define the port on which you want to connect

port = 12345

# connect to the server on local computer

s.connect(('127.0.0.1', port))

# receive data from the server

print s.recv(1024)

# close the connection

s.close()

**Conclusion:** Sockets are the endpoints of a bidirectional communications channel. Sockets may communicate within a process, between processes on the same machine, or between processes on  
different continents.  
Sockets may be implemented over a number of different channel types: Unix domain sockets, TCP, UDP, and so on. The socket library provides specific classes for handling the common transports as well as a generic interface for handling the rest.