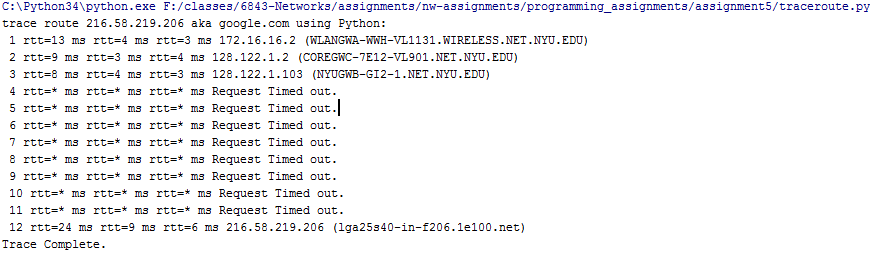
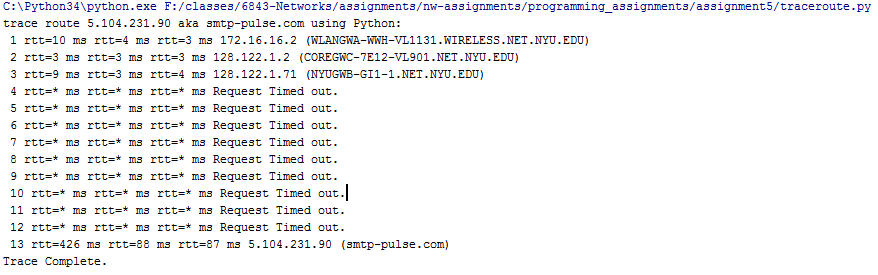
**Assignment 5 – Socket Programming Assignment – ICMP Traceroute and Pinger**

**ICMP Trace Route**

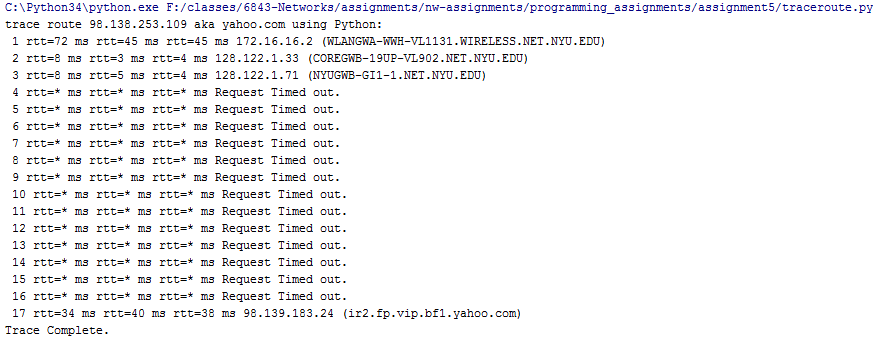
Output:



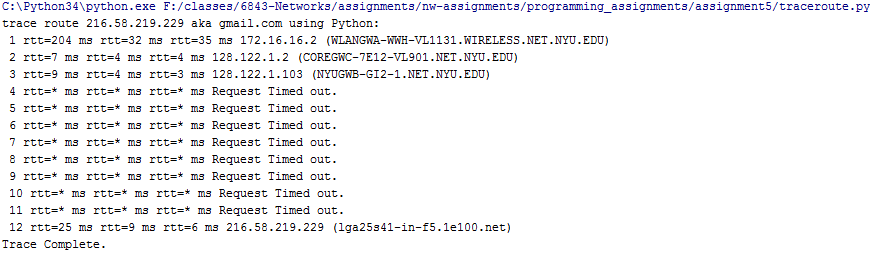
**Traceroute “google.com”**



**Traceroute “smtp-pulse.com”**

****

**Traceroute “yahoo.com”**

****

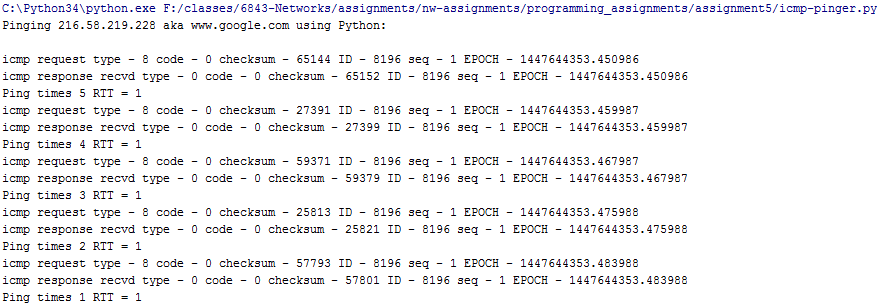
**Traceroute “gmail.com”**

Code – **traceroute.py**

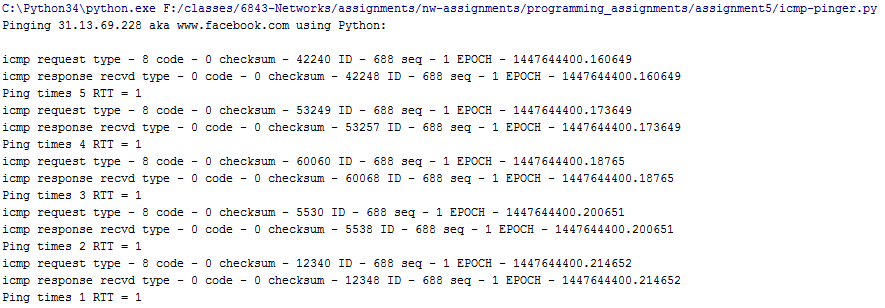
**from** socket **import** \*  
**import** os  
**import** sys  
**import** struct  
**import** time  
**import** select  
**import** binascii  
**import** socket  
**import** ctypes  
  
MAX\_HOPS = 30  
TIMEOUT = 2.0  
TRIES = 2  
ICMP\_ECHO\_REQUEST = 8  
ICMP\_ECHO\_REPLY = 0  
ICMP\_ECHO\_REQUEST\_CODE = 0  
ICMP\_ECHO\_REPLY\_CODE = 0  
  
**def** checksum(str):  
 *#print(str)* csum = 0  
 countTo = (len(str) / 2) \* 2  
 count = 0  
 **while** count < countTo:  
 thisVal = str[count+1] \* 256 + str[count]  
 csum = csum + thisVal  
 csum = csum & 0xffffffff  
 count = count + 2  
 **if** countTo < len(str):  
 csum = csum + ord(str[len(str) - 1])  
 csum = csum & 0xffffffff  
 csum = (csum >> 16) + (csum & 0xffff)  
 csum = csum + (csum >> 16)  
 answer = ~csum  
 answer = answer & 0xffff  
 answer = answer >> 8 | (answer << 8 & 0xff00)  
 **return** answer  
  
**def** build\_packet():  
 *# Header is type (8), code (8), checksum (16), id (16), sequence (16)* myChecksum = 0  
 ID = os.getpid() & 0xFFFF  
 *# Make a dummy header with a 0 checksum.  
 # struct -- Interpret strings as packed binary data* header = struct.pack(**"bbHHh"**, ICMP\_ECHO\_REQUEST, ICMP\_ECHO\_REQUEST\_CODE, myChecksum, ID, 1)  
 data = struct.pack(**"d"**,time.time() )  
 *# Calculate the checksum on the data and the dummy header.* myChecksum = checksum(header + data)  
  
 *# Get the right checksum, and put in the header* **if** sys.platform == **'darwin'**:  
 myChecksum = socket.htons(myChecksum) & 0xffff  
 *#Convert 16-bit integers from host to network byte order.* **else**:  
 myChecksum = socket.htons(myChecksum)  
 header = struct.pack(**"bbHHh"**, ICMP\_ECHO\_REQUEST, ICMP\_ECHO\_REQUEST\_CODE, myChecksum, ID, 1)  
 packet = header + data  
 **return** packet  
  
**def** get\_route(hostname):  
 timeLeft = TIMEOUT  
 done =0  
 print (**"trace route "** + socket.gethostbyname(hostname) + **' aka '**+hostname+**" using Python:"**)  
 **for** ttl **in** range(1,MAX\_HOPS+1):*#xrange is removed from python 3.0* timeTaken =[-1,-1,-1]  
 name=**''  
 for** tries **in** range(0,TRIES+1):*#xrange is removed from python 3.0  
 #Fill in start  
 # Make a raw socket named mySocket* destAddr = socket.gethostbyname(hostname)  
 icmp = socket.getprotobyname(**"icmp"**)  
 mySocket =socket.socket(socket.AF\_INET,socket.SOCK\_RAW,icmp)  
 mySocket.setsockopt(IPPROTO\_IP, IP\_TTL, struct.pack(**'I'**, ttl))  
 mySocket.settimeout(TIMEOUT)  
 *#Fill in end* **try**:  
 d = build\_packet()  
 mySocket.sendto(d, (hostname, 0))  
 t= time.time()  
 startedSelect = time.time()  
 whatReady = select.select([mySocket], [], [], timeLeft)  
 howLongInSelect = (time.time() - startedSelect)  
 recvPacket, addr = mySocket.recvfrom(1024)  
 **try**:  
 host = socket.gethostbyaddr(addr[0])  
 name = **'{0} ({1})'**.format(addr[0] , host[0])  
 **except** Exception:  
 name = **'{0} (host name/IP not found)'**.format(addr[0])  
 timeReceived = time.time()  
 timeLeft = timeLeft - howLongInSelect  
 **except** socket.timeout:  
 **continue  
 else**:  
 *#Fill in start  
 # Fetch the icmp type from the IP packet* icmpHeaderContent = recvPacket[20:28]  
 type, code, checksum, packetID, sequence = struct.unpack(**"bbHHh"**, icmpHeaderContent)  
 *#Fill in end* **if** type == 11:  
 bytes = struct.calcsize(**"d"**)  
 timeSent = struct.unpack(**"d"**, recvPacket[28:28 + bytes])[0]  
 timeTaken[tries] = (timeReceived -t)\*1000  
 *#print (" %d rtt=%.0f ms %s" %(ttl, (timeReceived -t)\*1000, addr[0]),socket.gethostbyaddr(addr[0])[0])* **elif** type == 3:  
 bytes = struct.calcsize(**"d"**)  
 timeSent = struct.unpack(**"d"**, recvPacket[28:28 + bytes])[0]  
 timeTaken[tries] =(timeReceived-t)\*1000  
 *#print (" %d rtt=%.0f ms %s" %(ttl, (timeReceived-t)\*1000, addr[0]),socket.gethostbyaddr(addr[0])[0])* **elif** type == 0:  
 bytes = struct.calcsize(**"d"**)  
 timeSent = struct.unpack(**"d"**, recvPacket[28:28 + bytes])[0]  
 timeTaken[tries]=(timeReceived - timeSent)\*1000  
 *#print (" %d rtt=%.0f ms %s" %(ttl, (timeReceived - timeSent)\*1000, addr[0]),socket.gethostbyaddr(addr[0])[0])* done = 1  
 **else**:  
 print (**"error"**)  
 **break  
 finally**:  
 mySocket.close()  
 i=0  
 a= [**'\*'**,**'\*'**,**'\*'**]  
 **if** timeTaken[i] != -1:  
 a[i] = int(timeTaken[i])  
 **if** timeTaken[i+1] != -1:  
 a[i+1] = int(timeTaken[i+1])  
 **if** timeTaken[i+2] != -1:  
 a[i+2] = int(timeTaken[i+2])  
 **if** timeTaken[i] == timeTaken[i+1] == timeTaken[i+2] == -1:  
 print(**" %d rtt=%s ms rtt=%s ms rtt=%s ms %s"** %(ttl, a[i],a[i+1],a[i+2] ,**'Request Timed out.'**))  
 **else**:  
 print(**" %d rtt=%s ms rtt=%s ms rtt=%s ms %s"** %(ttl, a[i],a[i+1],a[i+2] ,name))  
 **if** done == 1:  
 print(**'Trace Complete.'**)  
 **return**get\_route(**"google.com"**)  
get\_route(**"smtp-pulse.com"**)  
get\_route(**"yahoo.com"**)  
get\_route(**"gmail.com"**)

**ICMP Pinger**

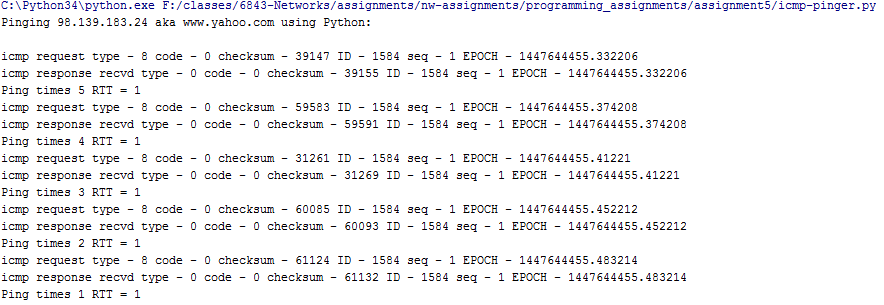
Output:



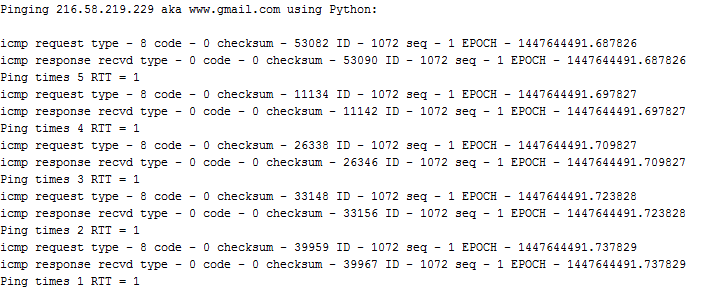
**ping “google.com”**



**ping “facebook.com”**



**ping “yahoo.com”**



**ping “gmail.com”**

Code- **icmp-pinger.py**

**from** socket **import** \*  
**import** os  
**import** sys  
**import** struct  
**import** time  
**import** select  
**import** binascii  
**import** socket  
**import** ctypes  
**import** math  
  
ICMP\_ECHO\_REQUEST = 8  
ICMP\_ECHO\_REPLY = 0  
ICMP\_ECHO\_REQUEST\_CODE = 0  
ICMP\_ECHO\_REPLY\_CODE = 0  
  
**def** checksum(str):  
 *#print(str)* csum = 0  
 countTo = (len(str) / 2) \* 2  
 count = 0  
 **while** count < countTo:  
 thisVal = str[count+1] \* 256 + str[count]  
 csum = csum + thisVal  
 csum = csum & 0xffffffff  
 count = count + 2  
 **if** countTo < len(str):  
 csum = csum + ord(str[len(str) - 1])  
 csum = csum & 0xffffffff  
 csum = (csum >> 16) + (csum & 0xffff)  
 csum = csum + (csum >> 16)  
 answer = ~csum  
 answer = answer & 0xffff  
 answer = answer >> 8 | (answer << 8 & 0xff00)  
 **return** answer  
  
**def** receiveOnePing(mySocket, ID, timeout, destAddr):  
 timeLeft = timeout  
 **while** 1:  
 timeLeft = timeout  
 startedSelect = time.time()  
 whatReady = select.select([mySocket], [], [], timeLeft)  
 howLongInSelect = (time.time() - startedSelect)  
 *#if whatReady[0] == []: # Timeout  
 #return "#Request timed out."* timeReceived = time.time()  
 recPacket, addr = mySocket.recvfrom(1024)  
  
 *#Fill in start  
 #Fetch the ICMP header from the IP packet* psize= len(recPacket)  
 *#advance to 128th byte coz thats where ICMP starts* icmp\_header = struct.unpack(**"bbHHhd"**,recPacket[psize-16:])  
 print(**'icmp response recvd type - '**+str(icmp\_header[0])+**' code - '**+str(icmp\_header[1])+  
 **' checksum - '**+str(icmp\_header[2])+**' ID - '**+str(icmp\_header[3])+**' seq - '**+str(icmp\_header[4])+**' EPOCH - '**+str(icmp\_header[5]))  
 *#Fill in end* timeLeft = timeLeft - howLongInSelect  
 timeLeft = math.ceil(timeLeft)  
 **if** timeLeft <= 0:  
 **return "#Request timed out."  
 else**:  
 **return** timeLeft  
  
**def** sendOnePing(mySocket, destAddr, ID):  
 *# Header is type (8), code (8), checksum (16), id (16), sequence (16)* myChecksum = 0  
 *# Make a dummy header with a 0 checksum.  
 # struct -- Interpret strings as packed binary data* header = struct.pack(**"bbHHh"**, ICMP\_ECHO\_REQUEST, ICMP\_ECHO\_REQUEST\_CODE, myChecksum, ID, 1)  
 tim = time.time()  
 data = struct.pack(**"d"**,tim )  
 *# Calculate the checksum on the data and the dummy header.* myChecksum = checksum(header + data)  
  
 *# Get the right checksum, and put in the header* **if** sys.platform == **'darwin'**:  
 myChecksum = socket.htons(myChecksum) & 0xffff  
 *#Convert 16-bit integers from host to network byte order.* **else**:  
 myChecksum = socket.htons(myChecksum)  
 header = struct.pack(**"bbHHh"**, ICMP\_ECHO\_REQUEST, ICMP\_ECHO\_REQUEST\_CODE, myChecksum, ID, 1)  
 packet = header + data  
 print(**'icmp request type - '**+str(ICMP\_ECHO\_REQUEST)+**' code - '**+str(ICMP\_ECHO\_REQUEST\_CODE)+  
 **' checksum - '**+str(myChecksum)+**' ID - '**+str(ID)+**' seq - '**+str(1)+**' EPOCH - '**+str(tim))  
 mySocket.sendto(packet, (destAddr, 1)) *# AF\_INET address must be tuple, not str  
 #Both LISTS and TUPLES consist of a number of objects  
 #which can be referenced by their position number within the object***def** doOnePing(destAddr, timeout):  
 icmp = socket.getprotobyname(**"icmp"**)  
 *#SOCK\_RAW is a powerful socket type. For more details see: http://sock-raw.org/papers/sock\_raw  
 #Fill in start  
 #Create Socket here* mySocket = socket.socket(socket.AF\_INET,socket.SOCK\_RAW,icmp)  
 *#Fill in end* myID = os.getpid() & 0xFFFF *#Return the current process i* sendOnePing(mySocket, destAddr, myID)  
 delay = receiveOnePing(mySocket, myID, timeout, destAddr)  
 mySocket.close()  
 **return** delay  
  
**def** ping(host, timeout=1):  
 *#timeout=1 means: If one second goes by without a reply from the server,  
 #the client assumes that either the client’s ping or the server’s pong is lost* dest = socket.gethostbyname(host)  
 print (**"Pinging "** + dest + **' aka '**+host+**" using Python:"**)  
 print (**""**)  
 *#Send ping requests to a server separated by approximately one second* i = 5  
 **while** i :  
 delay = doOnePing(dest, timeout)  
 print(**'Ping times '**+str(i)+**' RTT = '**+str(delay))  
 i-=1  
 **return** delay  
  
ping(**"www.google.com"**)  
ping(**"www.facebook.com"**)  
ping(**"www.yahoo.com"**)  
ping(**"www.gmail.com"**)