```
#!/usr/bin/env python

import telnetlib
import threading
import os.path
import subprocess
import time
import sys



#Checking IP address validity
def ip_is_valid():
    check = False
    global ip_list

    while True:
        #Prompting user for input
        ip_file = raw_input("Enter IP file name and extension: ")

        #Changing exception message
        try:
            #Open user selected file for reading (IP addresses file)
            selected_ip_file = open(ip_file, 'r')

            #Starting from the beginning of the file
```

```
        selected_ip_file.seek(0)

        #Reading each line (IP address) in the file
        ip_list = selected_ip_file.readlines()

        #Closing the file
        selected_ip_file.close()

    except IOError:
        print "\nFile %s does not exist! Please check and try again!\n" % ip_file

    #Checking octets
    for ip in ip_list:
        a = ip.split('.')

        if (len(a) == 4) and (1 <= int(a[0]) <= 223) and (int(a[0]) != 127) and (int(a[0]) != 169 or int(a[1]) !=
254) and (0 <= int(a[1]) <= 255 and 0 <= int(a[2]) <= 255 and 0 <= int(a[3]) <= 255):
            check = True
            break

        else:
            print '\n* There was an INVALID IP address! Please check and try again!\n'
            check = False
            continue

    #Evaluating the 'check' flag
```

```
    if check == False:
        continue

    elif check == True:
        break

#Checking IP reachability
print "\nChecking IP reachability...\n"

check2 = False

while True:
    for ip in ip_list:
        ping_reply = subprocess.call(['ping', '-c', '3', '-w', '3', '-q', '-n', ip])

        if ping_reply == 0:
            check2 = True
            continue

        elif ping_reply == 2:
            print "\nNo response from device %s." % ip
            check2 = False
            break

        else:
```

```
            print "\nPing to the following device has FAILED:", ip
            check2 = False
            break

        #Evaluating the 'check' flag
        if check2 == False:
            print "Please re-check IP address list or device.\n"
            ip_is_valid()

        elif check2 == True:
            print '\nAll devices are reachable. Waiting for command file...\n'
            break

#Checking command file validity
def cmd_is_valid():
    global cmd_file

    while True:
        cmd_file = raw_input("Enter command file name and extension: ")

        #Changing exception message
        if os.path.isfile(cmd_file) == True:
            print "\nSending command(s) to device(s)...\n"
            break
```

```
        else:
            print "\nFile %s does not exist! Please check and try again!\n" % cmd_file
            continue

#Change exception message
try:
    #Calling IP validity function
    ip_is_valid()

except KeyboardInterrupt:
    print "\n\nProgram aborted by user. Exiting...\n"
    sys.exit()

#Change exception message
try:
    #Calling command file validity function
    cmd_is_valid()

except KeyboardInterrupt:
    print "\n\nProgram aborted by user. Exiting...\n"
    sys.exit()


#Open telnet connection to devices
def open_telnet_conn(ip):
```

```
#Change exception message
try:
    #Define telnet parameters
    username = 'teopy'
    password = 'python'

    #Specify the Telnet port (default is 23, anyway)
    port = 23

    #Specify the connection timeout in seconds for blocking operations, like the connection attempt
    connection_timeout = 5

    #Specify a timeout in seconds. Read until the string is found or until the timout has passed
    reading_timeout = 5

    #Logging into device
    connection = telnetlib.Telnet(ip, port, connection_timeout)

    #Waiting to be asked for an username
    router_output = connection.read_until("Username:", reading_timeout)
    #Enter the username when asked and a "\n" for Enter
    connection.write(username + "\n")

    #Waiting to be asked for a password
    router_output = connection.read_until("Password:", reading_timeout)
```

```
#Enter the password when asked and a "\n" for Enter
connection.write(password + "\n")
time.sleep(1)

#Setting terminal length for the entire output - disabling pagination
connection.write("terminal length 0\n")
time.sleep(1)

#Entering global config mode
connection.write("\n")
connection.write("configure terminal\n")
time.sleep(1)

#Open user selected file for reading
selected_cmd_file = open(cmd_file, 'r')

#Starting from the beginning of the file
selected_cmd_file.seek(0)

#Writing each line in the file to the device
for each_line in selected_cmd_file.readlines():
   connection.write(each_line + '\n')
   time.sleep(1)

#Closing the file
```

```
        selected_cmd_file.close()


        #Test for reading command output
        #router_output = connection.read_very_eager()
        #print router_output


        #Closing the connection
        connection.close()


    except IOError:
        print "Input parameter error! Please check username, password and file name."



#Creating threads
def create_threads():
    threads = []
    for ip in ip_list:
        th = threading.Thread(target = open_telnet_conn, args = (ip,))   #args is a tuple with a single element
        th.start()
        threads.append(th)


    for th in threads:
        th.join()


#Calling threads creation function
```

```
create_threads()


#End of program
```