Introduction:

What is a port?

A port is a virtual point where network connections start and end. Ports are software-based and managed by a computer's operating system. Each port is associated with a specific process or service. Ports allow computers to easily differentiate between different kinds of traffic: emails go to a different port than webpages, for instance, even though both reach a computer over the same Internet connection.

What is a port number?

Ports are standardized across all network-connected devices, with each port assigned a number. Most ports are reserved for certain protocols — for example, all Hypertext Transfer Protocol (HTTP) messages go to port 80. While IP addresses enable messages to go to and from specific devices, port numbers allow targeting of specific services or applications within those devices.

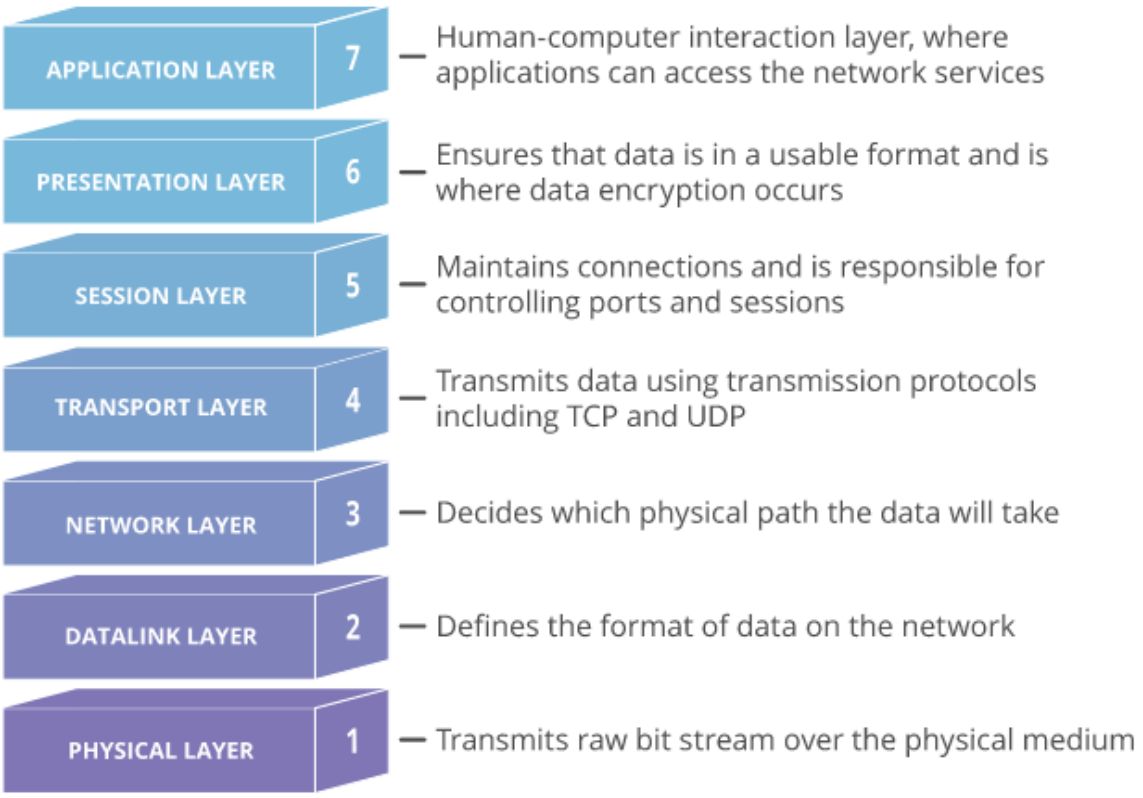
Vastly different types of data flow to and from a computer over the same network connection. The use of ports helps computers understand what to do with the data they receive.

Suppose Bob transfers an MP3 audio recording to Alice using the File Transfer Protocol (FTP). If Alice's computer passed the MP3 file data to Alice's email application, the email application would not know how to interpret it. But because Bob's file transfer uses the port designated for FTP (port 21), Alice's computer is able to receive and store the file.

Meanwhile, Alice's computer can simultaneously load HTTP webpages using port 80, even though both the webpage files and the MP3 sound file flow to Alice's computer over the same WiFi connection.

Are ports part of the network layer?

The OSI model is a conceptual model of how the Internet works. It divides different Internet services and processes into 7 layers. These layers are



Ports are a transport layer (layer 4) concept. Only a transport protocol such as the Transmission Control Protocol (TCP) or User Datagram Protocol (UDP) can indicate which port a packet should go to. TCP and UDP headers have a section for indicating port numbers. Network layer protocols — for instance, the Internet Protocol (IP) — are unaware of what port is in use in a given network connection. In a standard IP header, there is no place to indicate which port the data packet should go to. IP headers only indicate the destination IP address, not the port number at that IP address.

Usually, the inability to indicate the port at the network layer has no impact on networking processes, since network layer protocols are almost always used in conjunction with a transport layer protocol. However, this does impact the functionality of testing software, which is software that "pings" IP addresses using Internet Control Message Protocol (ICMP) packets. ICMP is a network layer protocol that can ping networked devices — but without the ability to ping specific ports, network administrators cannot test specific services within those devices.

Some ping software, such as My Traceroute, offers the option to send UDP packets. UDP is a transport layer protocol that can specify a particular port, as opposed to ICMP, which cannot specify a port. By adding a UDP header to ICMP packets, network administrators can test specific ports within a networked device.

Why do firewalls sometimes block specific ports?

A firewall is a security system that blocks or allows network traffic based on a set of security rules. Firewalls usually sit between a trusted network and an untrusted network; often the untrusted network is the Internet. For example, office networks often use a firewall to protect their network from online threats.

Some attackers try to send malicious traffic to random ports in the hopes that those ports have been left "open," meaning they are able to receive traffic. This action is somewhat like a car thief walking down the street and trying the doors of parked vehicles, hoping one of them is unlocked. For this reason, firewalls should be configured to block network traffic directed at most of the available ports. There is no legitimate reason for the vast majority of the available ports to receive traffic.

Properly configured firewalls block traffic to all ports by default except for a few predetermined ports known to be in common use. For instance, a corporate firewall could only leave open ports 25 (email), 80 (web traffic), 443 (web traffic), and a few others, allowing internal employees to use these essential services, then block the rest of the 65,000+ ports.

As a more specific example, attackers sometimes attempt to exploit vulnerabilities in the RDP protocol by sending attack traffic to port 3389. To stop these attacks, a firewall may block port 3389 by default. Since this port is only used for remote desktop connections, such a rule has little impact on day-to-day business operations unless employees need to work remotely.

What are the different port numbers?

There are 65,535 possible port numbers, although not all are in common use. Some of the most commonly used ports, along with their associated networking protocol, are:

- **Ports 20 and 21:** File Transfer Protocol (FTP). FTP is for transferring files between a client and a server.
- **Port 22:** Secure Shell (SSH). SSH is one of many tunneling protocols that create secure network connections.
- **Port 25:** Simple Mail Transfer Protocol (SMTP). SMTP is used for email.
- **Port 53:** Domain Name System (DNS). DNS is an essential process for the modern Internet; it matches human-readable domain names to machine-readable IP addresses, enabling users to load websites and applications without memorizing a long list of IP addresses.
- **Port 80:** Hypertext Transfer Protocol (HTTP). HTTP is the protocol that makes the World Wide Web possible.
- **Port 123:** Network Time Protocol (NTP). NTP allows computer clocks to sync with each other, a process that is essential for encryption.
- **Port 179:** Border Gateway Protocol (BGP). BGP is essential for establishing efficient routes between the large networks that make up the Internet (these large networks are called autonomous systems). Autonomous systems use BGP to broadcast which IP addresses they control.

- **Port 443:** HTTP Secure (HTTPS). HTTPS is the secure and encrypted version of HTTP. All HTTPS web traffic goes to port 443. Network services that use HTTPS for encryption, such as DNS over HTTPS, also connect at this port.
- **Port 500:** Internet Security Association and Key Management Protocol (ISAKMP), which is part of the process of setting up secure IPsec connections.
- **Port 3389:** Remote Desktop Protocol (RDP). RDP enables users to remotely connect to their desktop computers from another device.

The Internet Assigned Numbers Authority (IANA) maintains the full list of port numbers and protocols assigned to them.

What is Port Scanner:

A port scanner is a computer program that checks network ports for one of three possible statuses – open, closed, or filtered.

Port scanners are valuable tools in diagnosing network and connectivity issues. However, attackers use port scanners to detect possible access points for infiltration and to identify what kinds of devices you are running on the network, like firewalls, proxy servers or VPN servers.

Implementation

IDE Used:IDLE

Python-version:3.8

Modules used:

1)Socket

To install:

1)install python3-pip

$sudo apt install python3-pip

2)pip3 install socket or sudo apt install python3-socket

This module provides access to the BSD *socket* interface. It is available on all modern Unix systems, Windows, MacOS, and probably additional platforms.

Creating a socket

For creating a ipv4 family socket: AF_INET

For creating ipv6 family socket: AF_INET6

For tcp Socket: SOCK_STREAM

For udp Socket   :SOCK_DGRAM

For connecting:

socket.connect(*address*)

Connect to a remote socket at *address*. (The format of *address* depends on the address family — see above.)

socket.connect_ex(*address*)

Like connect(address), but return an error indicator instead of raising an exception for errors returned by the C-level connect() call (other problems, such as “host not found,” can still raise exceptions). The error indicator is 0 if the operation succeeded, otherwise the value of the errno variable. This is useful to support, for example, asynchronous connections.

2)Thread

Multithreading is a threading technique in Python programming to run multiple threads concurrently by rapidly switching between threads with a CPU help (called context switching). Besides, it allows sharing of its data space with the main threads inside a process that share information and communication with other threads easier than individual processes. Multithreading aims to perform multiple tasks simultaneously, which increases performance, speed and improves the rendering of the application.

A **threading** module is made up of a **Thread** class, which is instantiated to create a Python thread.

To start a thread in Python multithreading, call the thread class's object. The start() method can be called once for each thread object; otherwise, it throws an exception error.

To import :

import threading

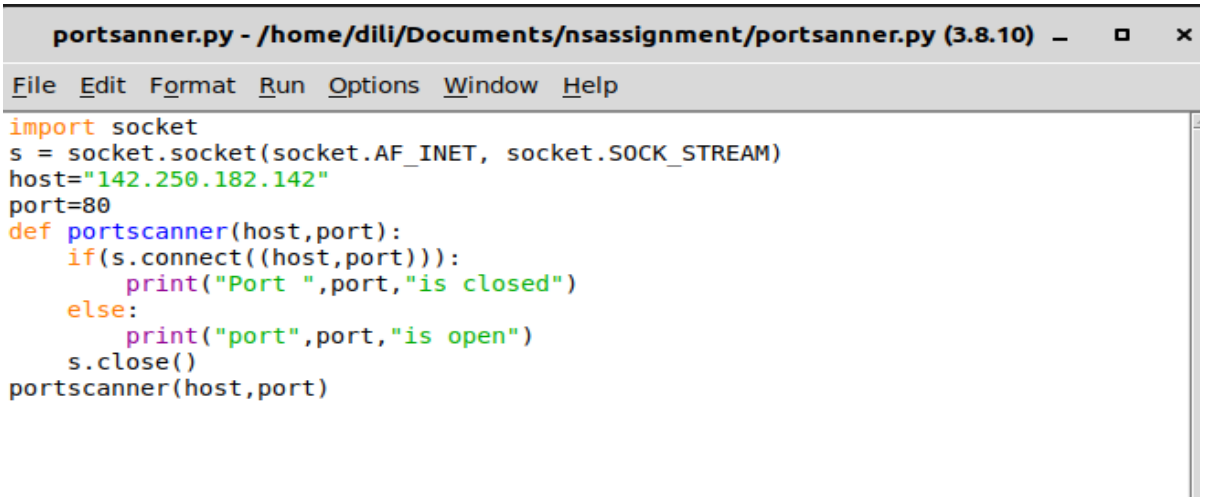
To Declare Thread Parameters:

threading.Thread(target = functonanme, args =(arguments to be passed)

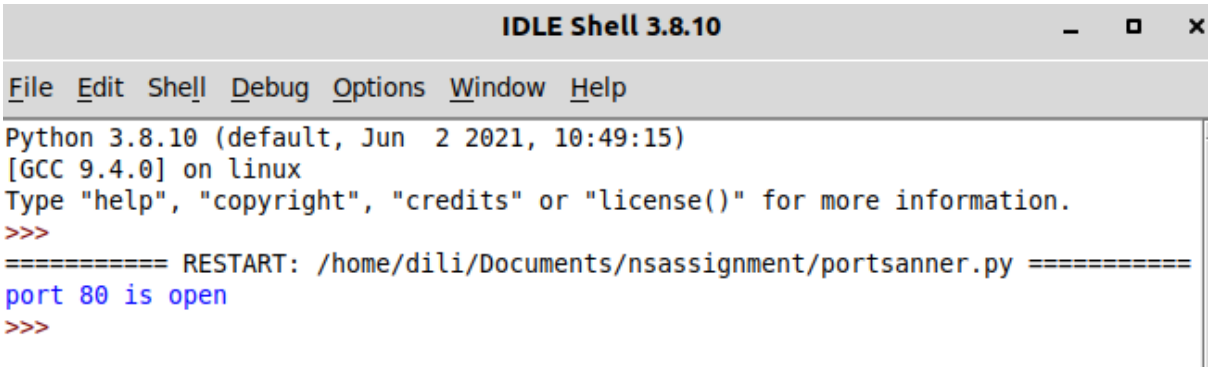
To start a thread:

threading.Thread(target = functonanme, args =(arguments to be passed).start()

Sample Script:



Output:



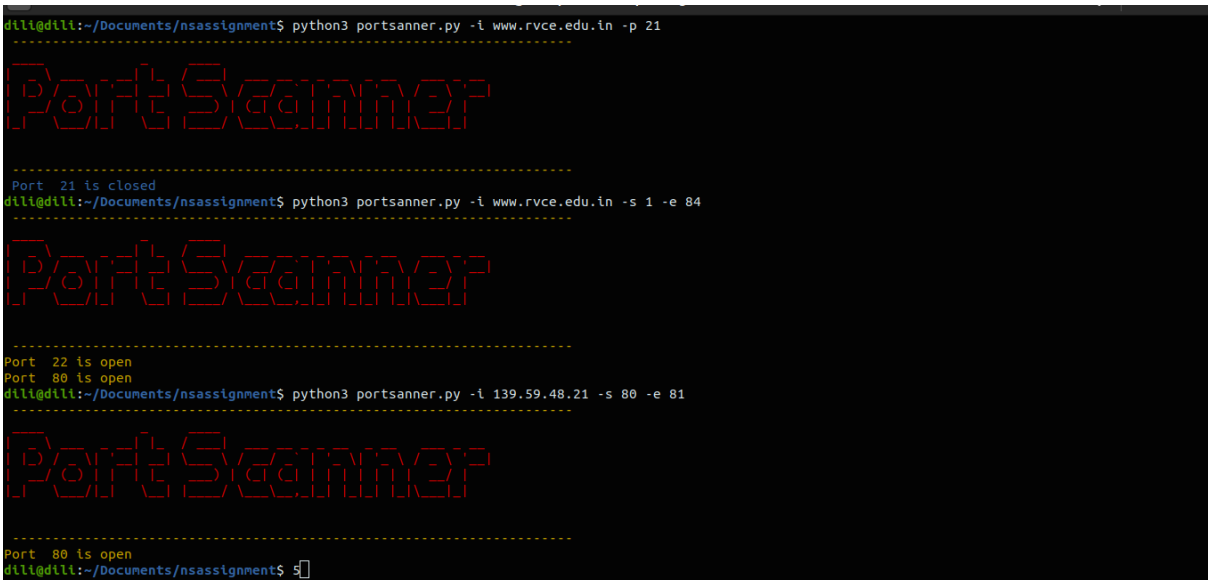
Complete Code:

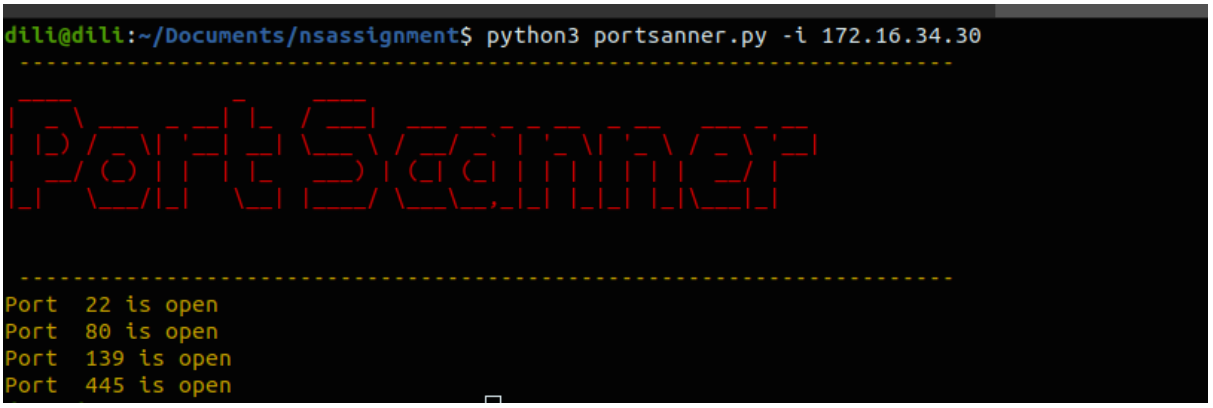
```
import socket
import optparse
import threading
from colorama import Fore, Style
from pyfiglet import Figlet
from termcolor import colored
f=Figlet(font="standard")
print(Fore.YELLOW,"-"*70)
print(colored(f.renderText(f'Port Scanner'), "red"))
print(Fore.YELLOW,"-"*70)
parser = optparse.OptionParser()
parser.add_option("-s","--sport",dest="sport",help="Add starting port
```

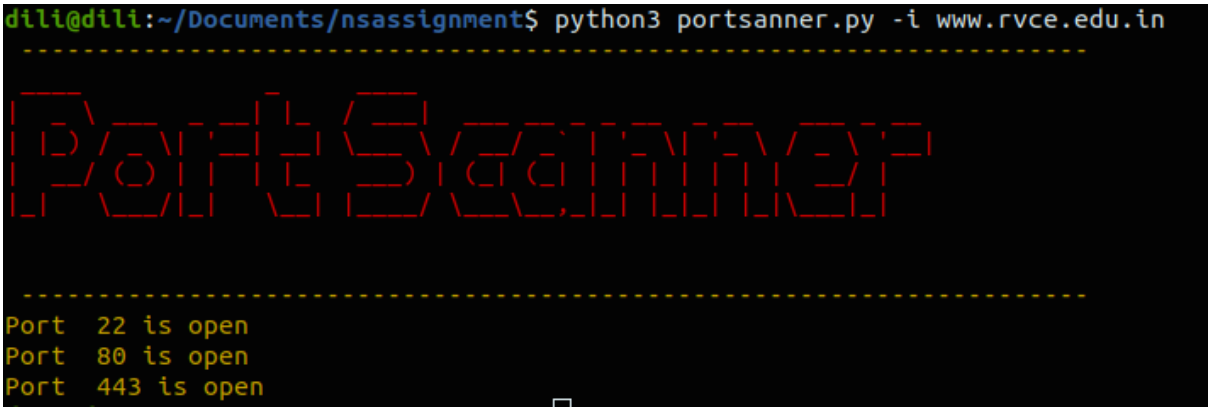
```
number to scan")
parser.add_option("-e","--eport",dest="eport",help="Add ending port
number to scan")
parser.add_option("-i","--host",dest="host",help="Add host")
parser.add_option("-p","--port",dest="port",help="Add single port number
to scan")
(options,arguments)=parser.parse_args()
if not options.host:
      parser.error(Fore.RED+"[-] Please Enter hostname or ip address")
elif options.port:
    s = socket.socket(socket.AF_INET, socket.SOCK_STREAM)
    if not (s.connect_ex((options.host,int(options.port)))):
       print(Fore.GREEN,"Port ",options.port,"is open")
    else:
       print(Fore.BLUE,"Port ",options.port,"is closed")
    s.close()
    exit()

options.sport= 0 if not (options.sport) else options.sport
options.eport=1024 if not (options.eport) else options.sport

try:
    sport=int(options.sport)
    eport=int(options.eport)
except:
    print(Fore.RED+"[-] Please Enter the correct port number")
    exit()
if options.port:
    s = socket.socket(socket.AF_INET, socket.SOCK_STREAM)
    if not (s.connect_ex((host,options.port))):
       print(Fore.GREEN,"Port ",options.port,"is open")
    s.close()

try:
    host=socket.gethostbyname(options.host)
except:
    print(Fore.RED,"[-] Please Enter the correct hostname or
ip-address")
    exit()

def loo():
      for i in range(sport,eport):
      s = socket.socket(socket.AF_INET, socket.SOCK_STREAM)
      if not (s.connect_ex((host,i))):
            print("Port ",i,"is open")
      s.close()
```

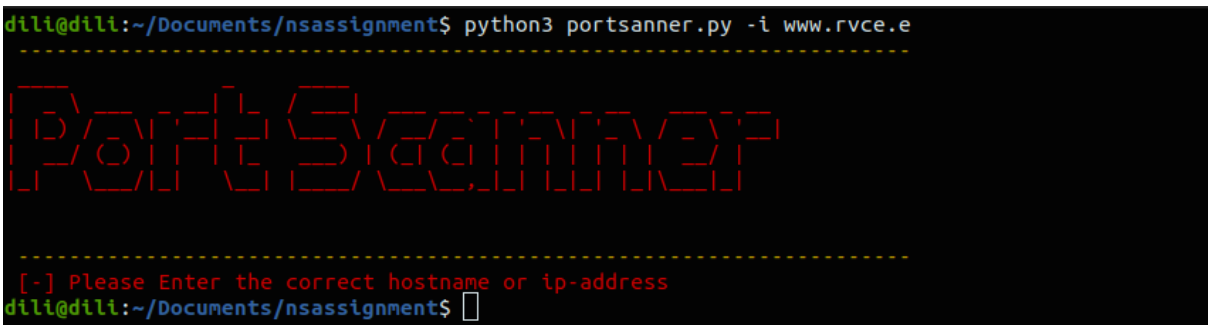
```
threading.Thread(target=loo).start()
```

OUTPUT:




dili@dili:~/Documents/nsassignment$ python3 portsanner.py -i 172.16.34.30
Port 22 is open
Port 80 is open
Port 139 is open
Port 445 is open


dili@dili:~/Documents/nsassignment$ python3 portsanner.py -i www.rvce.edu.in
Port 22 is open
Port 80 is open
Port 443 is open


dili@dili:~/Documents/nsassignment$ python3 portsanner.py -i www.rvce.e
[-] Please Enter the correct hostname or ip-address
dili@dili:~/Documents/nsassignment$