**Command Line**

**basics**

cls – clears the command line interface of perhaps previously used commands and previously used lines

dir -

mkdir <foldername> -

cd <directory> -

cd.. – moves up to directory once

cd/ - moves to topmost directory

c: or d: or e: - transfers to different drives in desktop

exit – exits the command line

chkdsk -

rmdir <foldertodeleteincurrdir> - deletes the folder in the directory currently in

ren <\*substringtoreplace> <\*newsubstring> - replaces a substring all files that have the same substring with new substring ex. ren \*.jpg \*.png

del <filenameincurrdirectory> - deletes file in current directory

echo > <filename>.<fileextension> - creates a file in current directory

**network**

ipconfig –

ftp -

ftype -

netsh -

netstat -

nslookup -

pathping -

ping <ip-address> – a command that

Example 1:

C:\Users\Cueva>ping 192.168.10.5

Pinging 192.168.10.5 with 32 bytes of data:

Request timed out.

Request timed out.

Request timed out.

Request timed out.

Ping statistics for 192.168.10.5:

Packets: Sent = 4, Received = 0, Lost = 4 (100% loss),

Example 2:

C:\Users\Cueva>ipconfig

Windows IP Configuration

Ethernet adapter Local Area Connection:

Connection-specific DNS Suffix . : bbrouter

IPv6 Address. . . . . . . . . . . : 2001:4451:8262:8d00:69:b02a:7e88:1d9

Temporary IPv6 Address. . . . . . : 2001:4451:8262:8d00:1417:e42e:c810:750

Link-local IPv6 Address . . . . . : fe80::69:b02a:7e88:1d9%11

IPv4 Address. . . . . . . . . . . : 192.168.1.3

Subnet Mask . . . . . . . . . . . : 255.255.255.0

Default Gateway . . . . . . . . . : fe80::9261:cff:fe67:92dd%11

192.168.1.1

Tunnel adapter isatap.bbrouter:

Media State . . . . . . . . . . . : Media disconnected

Connection-specific DNS Suffix . : bbrouter

C:\Users\Cueva>ping 192.168.1.3

Pinging 192.168.1.3 with 32 bytes of data:

Reply from 192.168.1.3: bytes=32 time<1ms TTL=128

Reply from 192.168.1.3: bytes=32 time<1ms TTL=128

Reply from 192.168.1.3: bytes=32 time<1ms TTL=128

Reply from 192.168.1.3: bytes=32 time<1ms TTL=128

Ping statistics for 192.168.1.3:

Packets: Sent = 4, Received = 4, Lost = 0 (0% loss),

Approximate round trip times in milli-seconds:

Minimum = 0ms, Maximum = 0ms, Average = 0ms

route -

systeminfo -

telnet -

tftp -

tracert <ip-address> -

Enter the command ipconfig to see ip address, this will display the following.

Windows IP Configuration

Ethernet adapter Local Area Connection:

Connection-specific DNS Suffix . : bbrouter

IPv6 Address. . . . . . . . . . . : 2001:4451:81f6:2000:69:b02a:7e88:1d9

Temporary IPv6 Address. . . . . . : 2001:4451:81f6:2000:50b0:d8:8c5f:c353

Link-local IPv6 Address . . . . . : fe80::69:b02a:7e88:1d9%11

IPv4 Address. . . . . . . . . . . : 192.168.1.4

Subnet Mask . . . . . . . . . . . : 255.255.255.0

Default Gateway . . . . . . . . . : fe80::9261:cff:fe67:92dd%11

192.168.1.1

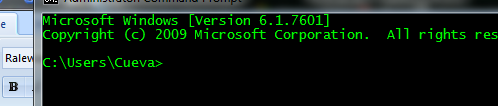
Tunnel adapter isatap.bbrouter:

Media State . . . . . . . . . . . : Media disconnected

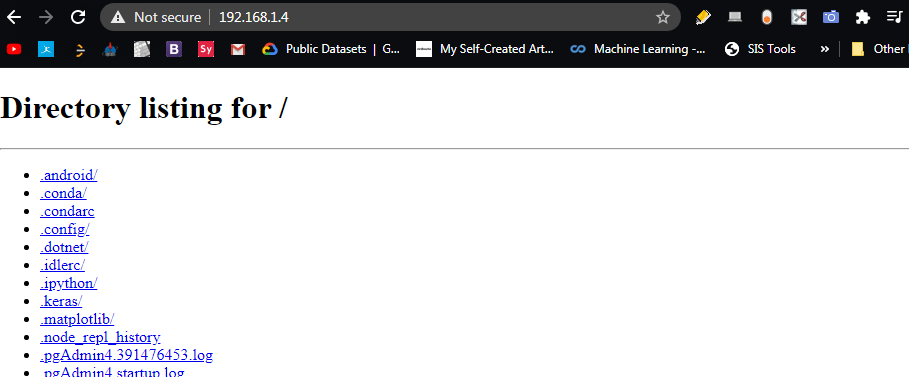
Connection-specific DNS Suffix . : bbrouter

What we should type in the url is the IPv4 address. At first it will load a page that is unable to connect, but if we enter the command python –m http.server <optionalportnumberarg>

From here we can now load the page that was previously unable to connect to the local server. The page will display the files that are in the default path. To confirm this just go to the terminal and notice the first file path it is in right after opening the terminal window

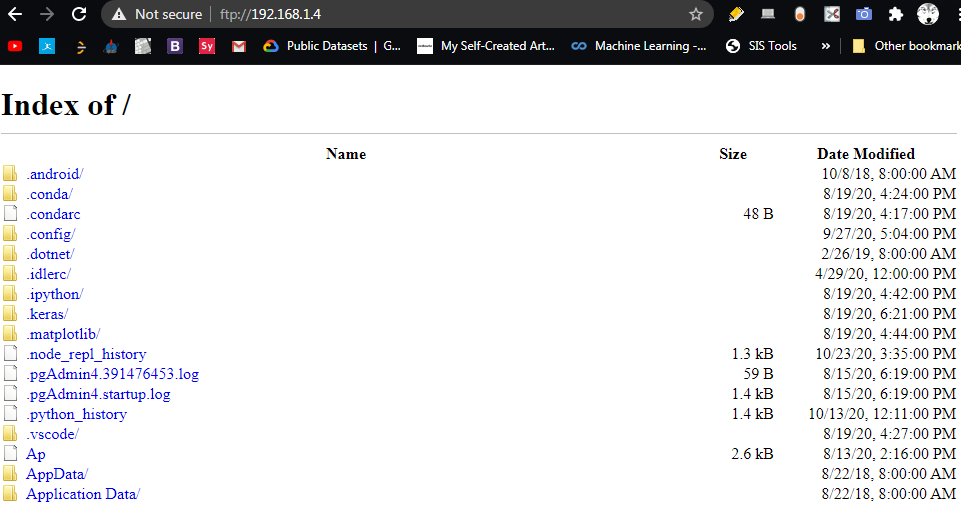


The files in this path will be displayed on the page we have requested using a local server



Another command is python –m pyftpdlib –p 21 w, dependencies will be to first install the pyftpdlib module by using the pip install command to install this module.

When this command is ran this will display a file manager like page like the command we previously ran which was python -m http.server <optionalportnumgerarg>. To display the page we just type ftp://<ipaddress> on the url box



**Networking**

to give an overview a socket in networking is generally one endpoint of a two way communication link between two programs running on the network.

A **server** normally runs on a specific computer and has a socket that is **bound to a specific port number**, a port is where two connections meet, e.g. usb port, headphone ports, etc. The server generally **just waits listening to the socket for a client to make a connection request**

The client or **user knows the hostname of the machine on which the server is running** and the port number on which the server is listening

client

port

listener

server

port

Connection request

To make a connection request the client tries to meet with the server on the server machine and port and most importantly the client needs to identify itself to the server so it binds to a local port number that it will use during this connection. **This assignment is usually done by the system**.

If successful the port of the client side connects to the port of the server side, upon acceptance the server gets a new socket bound to the same local port, e.g. 127.0.0.1 to port 8000. The server also has its remote endpoint set to the address and port of the client

And endpoint is a combination of an IP address and a port number.

**socket**

socket.socket() – returns a socket object, whose methods implement various socket system calls

args:

methods:

self.connect\_ex() -

self.close() -

socket.gaierror() -

socket.error() -

socket.AF\_INET -

socket.SOCK\_STREAM -

socket.setdefaulttimeout -

**sys**

sys.argv – is an attribute that holds the arguments in array when the python file is ran. Like the c language’s argument vector, the keyword that indicates the file to be ran is the first argument “<filename>.py” and the succeeding arguments separated by whitespaces will be treated as the nth argument

['sample.py', 'test']

Sys.exit(status) – is a function that will exit the interpreter by raising SystemExit(status). If the status argument is omitted or None, it defaults to zero (i.e., success). If the status is an integer, it will be used as the system exit status. If it is another kind of object, it will be printed and the system exit status will be one (i.e., failure).

This is also why when we use the python interpreter in the command line when we use the exit() function we also execute the same kind of exit() function in the sys module

**OS**

os.remove(string path and directory of file to be deleted) –

os.listdir(string directory of different files) – will list all files in directory

os.mkdir(string parentdir/nameoffolder, mode=0o7777, \*, dir\_fd=None) - creates a folder named path with numeric mode 'mode' if the directory already exists FileExistsError is raised

os.rmdir(string parentdir/nameoffolder,\*,dir\_fd=None) – removes the folder in the parent directory, if the directory does not exist or is not empty, an FileNotFoundError or an OSError is raised respectively

os.rename(path directory of file or directory/folder, newnameoffile or directory/folder) - raises an exception FileNotFoundError if source arg is a file or folder that cannot be found or if folder or file already exists with the new name you specified

**shutil**

shutil.move(path of file to be moved, destination path of file to be moved) - Recursively moves a file or directory (source) to another location (destination) and returns the destination. If the destination directory already exists then src is moved inside that directory. Returns a string which is the path of the file that was moved

**watchdogs.events**

for event handlers and file changes in the system we use the watchdog.events module to monitor and act on these events or changes in our directories

1. create event handler - the object that will be notified when something happen on the filesystem you are monitoring.

method overriding occurs simply defining in the child class a method with the same name of a method in the parent class. When you define a method in the object you make the latter able to satisfy that method call, so the implementations of its ancestors do not come in play.

The following are some of the useful methods that can be overridden in the FileSystemEventHandler class

* on\_any\_event: Catch-all event handler.
* on\_created: Executed when a file or a directory is created
* on\_modified: Called when a file or directory is modified.
* on\_moved: Called when a file or a directory is moved or renamed.
* on\_deleted: Executed when a file or directory is deleted.

list of errors

KeyboardInterrupt

A function, class, and its methods can be the only ones to be imported and used

automation ideas for faster work

- writing on notepad then transferring it to word

- when flash drive is plugged in automatically copy or move

files from a specific folder to the flash drive

- on startup open youtube in google and play youtube

created playlist

- on startup open vs code, notepad, and command line

**Vague roadmap for skills needed**

Linux

Computer networking

Forensics

System administration

Personal digital security

Command line

Using virtual machines

Reverse engineering

Cyber operations

Foreign terminologies:

Recon – finding/observation information, OSINT (open source intelligence, hacker finding open source information)

Tcp/ip – transmissional control protocol/internet protocol, is a set of rules that governs the connection of computer systems to the internet.

Lan – local area network that interconnects hosts that are up to a few or maybe a few tens of km apart

Man – metroplotan area network interconnects devices that are up to a few hundred km apart

Wan – wide area network interconnects devices that can be located anywhere on earth

start with what you want to protect first

cloud, database, web server, website etc.