NSA

1 Execute basic network commands ipconfig, ping, traceroute, Nslookup 15-01-2024 CO1

ipconfig

```
import subprocess
def get_ipconfig():
    result = subprocess.run(['ipconfig'], capture_output= True, text= True)
    return result.stdout
print(get_ipconfig())
```

```
PS C:\Users\ajcemca\Documents\nsa lab> & C:/Users/ajcemca/AppData/Local/Microsoft/WindowsA
nsa lab/ipconfig.py"

Windows IP Configuration


Ethernet adapter Ethernet 2:

   Connection-specific DNS Suffix  . :
   Link-local IPv6 Address . . . . . : fe80::23ac:b3bb:c943:5781%4
   IPv4 Address. . . . . . . . . . . : 192.168.6.28
   Subnet Mask . . . . . . . . . . . : 255.255.255.0
   Default Gateway . . . . . . . . . : 192.168.6.100

PS C:\Users\ajcemca\Documents\nsa lab>
```

ping

```
import subprocess

def ping(host):
    result = subprocess.run(['ping', host], capture_output=True, text=True)
    return result.stdout

host_to_ping = 'google.com'
print(pathping(host_to_ping))
```

traceroute

```
import subprocess

def traceroute(host):
    result = subprocess.run(['tracert', host], capture_output=True, text=True)
    return result.stdout
```

```
host_to_traceroute = 'google.com'
print(traceroute(host_to_traceroute))
```

```
PS C:\Users\ajcemca\Documents\nsa lab> py traceroute.py

Tracing route to google.com [142.250.196.78]
over a maximum of 30 hops:

  1    <1 ms    <1 ms    <1 ms  192.168.6.100
  2     1 ms     1 ms     1 ms  136.232.57.109
  3    15 ms    15 ms    15 ms  172.20.97.57
  4    17 ms    17 ms    17 ms  172.27.9.126
  5    18 ms    18 ms    18 ms  172.27.9.125
  6    17 ms    17 ms    17 ms  172.27.109.51
  7    17 ms    17 ms    17 ms  172.16.5.90
  8    18 ms    18 ms    18 ms  142.250.236.157
  9    16 ms    16 ms    16 ms  maa03s46-in-f14.1e100.net [142.250.196.78]

Trace complete.
```

<u>nslookup</u>

```
import subprocess

def nslookup(host):
    result = subprocess.run(['nslookup', host],capture_output= True, text= True)
    return result.stdout
host_to_nslookup = 'google.com'
print(nslookup(host_to_nslookup))
```

```
PS C:\Users\ajcemca\Documents\nsa lab> py nslookup.py
Server:  UnKnown
Address:  192.168.6.254

Name:    google.com
Addresses:  2404:6800:4007:82b::200e
          142.250.77.174
```

2 Apply command pathping 15-01-2024 CO1

```
import subprocess

def pathping(host):
    result = subprocess.run(['pathping', host], capture_output=True, text=True)
    return result.stdout
```

```
host_to_pathping = 'google.com'
print(pathping(host_to_pathping))
```

```
Tracing route to google.com [142.250.77.174]
over a maximum of 30 hops:
  0  S28.mca.com [192.168.6.28]
  1  192.168.6.100
  2  136.232.57.109
  3  172.20.97.57
  4  172.27.9.126
  5  172.27.9.125
  6  172.27.109.51
  7  172.16.5.90
  8  209.85.247.229
  9  maa05s17-in-f14.1e100.net [142.250.77.174]

Computing statistics for 225 seconds...
            Source to Here   This Node/Link
Hop  RTT    Lost/Sent = Pct  Lost/Sent = Pct  Address
  0                                           S28.mca.com [192.168.6.28]
                                 0/ 100 =  0%   |
  1    0ms     0/ 100 =  0%     0/ 100 =  0%  192.168.6.100
                                 0/ 100 =  0%   |
  2    1ms     0/ 100 =  0%     0/ 100 =  0%  136.232.57.109
                                 0/ 100 =  0%   |
  3  ---     100/ 100 =100%   100/ 100 =100%  172.20.97.57
                                 0/ 100 =  0%   |
  4  ---     100/ 100 =100%   100/ 100 =100%  172.27.9.126
                                 0/ 100 =  0%   |
  5  ---     100/ 100 =100%   100/ 100 =100%  172.27.9.125
                                 0/ 100 =  0%   |
  7  ---     100/ 100 =100%   100/ 100 =100%  172.16.5.90
                                 0/ 100 =  0%   |
  8   18ms     0/ 100 =  0%     0/ 100 =  0%  209.85.247.229
                                 0/ 100 =  0%   |
  9   18ms     0/ 100 =  0%     0/ 100 =  0%  maa05s17-in-f14.1e100.net [142.250.77.174]

Trace complete.
```

3 Design a network using Distance vector routing protocol - 19-01-2024 CO1

```
import copy
class Router:
    def __init__(self, name):
        self.name = name
        self.routing_table = {}

    def update_routing_table(self, destination, cost):
        self.routing_table[destination] = cost

    def get_routing_table(self):
        return copy.deepcopy(self.routing_table)

def distance_vector_algorithm(routers, max_iterations=10):
```

```
    for _ in range(max_iterations):
        for router in routers:
            for destination, cost in router.get_routing_table().items():
                for neighbor in routers:
                    if neighbor != router:
                        if destination not in neighbor.routing_table or \
                            router.routing_table[destination] + neighbor.routing_table[router.name] <
neighbor.routing_table[destination]:
                                neighbor.update_routing_table(destination, router.routing_table[destination]
+ neighbor.routing_table[router.name])

def print_routing_tables(routers):
    for router in routers:
        print("Router", (router.name), "Routing Table:")
        for destination, cost in router.get_routing_table().items():
            print(f"\tDestination: {destination} Cost: {cost}")
        print()

if __name__ == "__main__":
    router_A = Router("A")
    router_B = Router("B")
    router_C = Router("C")

    router_A.update_routing_table("A",0)
    router_A.update_routing_table("B",1)
    router_A.update_routing_table("C",float('inf'))

    router_B.update_routing_table("A",1)
    router_B.update_routing_table("B",0)
    router_B.update_routing_table("C",1)

    router_C.update_routing_table('A', float('inf'))
    router_C.update_routing_table('B', 1)
    router_C.update_routing_table('C', 0)

    routers = [router_A, router_B, router_C]

    distance_vector_algorithm(routers)

    print_routing_tables(routers)
```

```
PS C:\Users\ajcemca\Documents\nsa lab> & C:/Users/ajcemca/AppData/Local/Micros

Router A Routing Table:
        Destination: A Cost: 0
        Destination: B Cost: 1
        Destination: C Cost: 2

Router B Routing Table:
        Destination: A Cost: 1
        Destination: B Cost: 0
        Destination: C Cost: 1

Router C Routing Table:
        Destination: A Cost: 2
        Destination: B Cost: 1
        Destination: C Cost: 0
```

4 Design a network using Link State routing protocol 22-01-2024 CO1

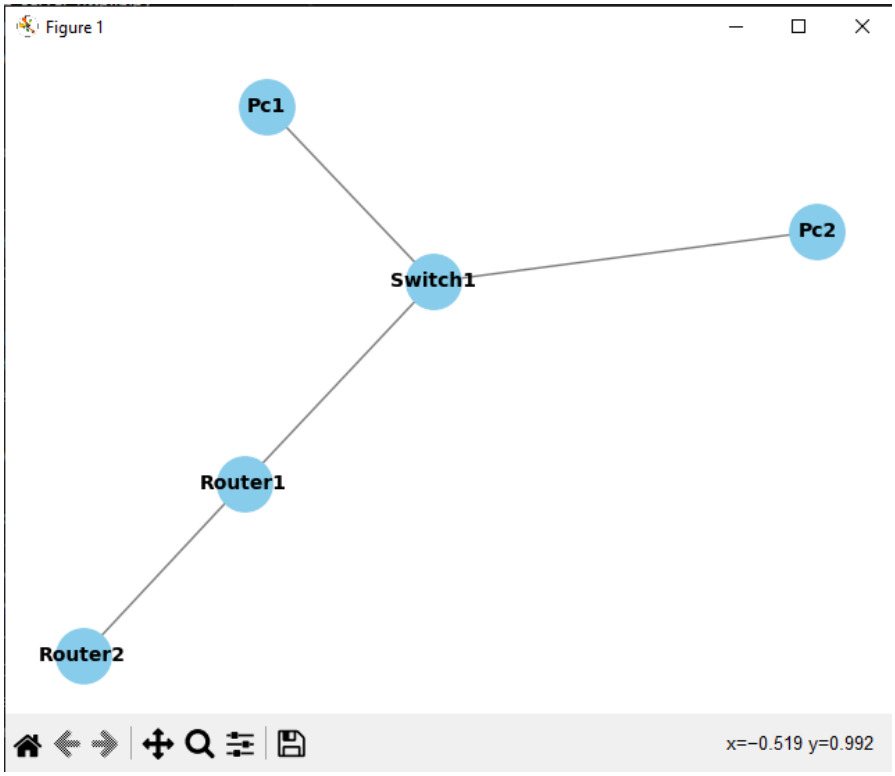
```
import copy
import heapq

class Router:
    def __init__(self, name):
        self.name = name
        self.links = {}
        self.routing_table = {}

    def add_link(self, neighbor, cost):
        self.links[neighbor] = cost

    def update_routing_table(self, destination, cost):
        self.routing_table[destination] = cost

    def get_links(self):
        return copy.deepcopy(self.links)

    def get_routing_table(self):
        return copy.deepcopy(self.routing_table)

def dijkstra_algorithm(routers, source):
    heap = [(0, source)]
    visited = set()

    while heap:
```

```
        (cost, current_router) = heapq.heappop(heap)

        if current_router in visited:
            continue

        visited.add(current_router)

        for neighbor, link_cost in routers[current_router].get_links().items():
            new_cost = cost + link_cost
            if neighbor not in visited:
                heapq.heappush(heap, (new_cost, neighbor))
                routers[neighbor].update_routing_table(neighbor, new_cost)

def link_state_algorithm(routers):
    for router in routers.values():
        for neighbor, link_cost in router.get_links().items():
            dijkstra_algorithm(routers, router.name)

def print_routing_table(routers):
    for router in routers:
        print(f"Router {router.name} routing table:")
        for destination, cost in router.get_routing_table().items():
            print(f"Destination: {destination} Cost: {cost}")
        print()

if __name__ == "__main__":
    router_A = Router("A")
    router_B = Router("B")
    router_C = Router("C")

    router_A.add_link("B", 1)
    router_B.add_link("A", 1)
    router_B.add_link("C", 1)
    router_C.add_link("B", 1)

    routers = {"A": router_A, "B": router_B, "C": router_C}

    link_state_algorithm(routers)
    print_routing_table(routers.values())
```

```
PS D:\S8\404_ACN> py LS.py
Router A routing table:
Destination: A Cost: 2

Router B routing table:
Destination: B Cost: 1

Router C routing table:
Destination: C Cost: 1
```

5 Connect the computers in local area network - Simulation 29-01-2024 CO1

```
import networkx as nx
import matplotlib.pyplot as plt
G = nx.Graph()
G.add_node("Router1")
G.add_node("Router2")
G.add_node("Switch1")
G.add_node("Pc1")
G.add_node("Pc2")

G.add_edge("Router1","Switch1")
G.add_edge("Switch1","Pc1")
G.add_edge("Switch1","Pc2")
G.add_edge("Router1","Router2")

pos=nx.spring_layout(G)
nx.draw(G,pos,with_labels=True,node_size
=800,node_color="skyblue",font_size=10,font_weight="bold",edge_color="grey",linewidths=1,arr
ows=True)
plt.title("Network Topology")
plt.axis("off")
plt.show()
```



6 Write a program for implementing an echo server in Python 29-01-2024 CO2

Client

```
import socket

client_socket = socket.socket(socket.AF_INET, socket.SOCK_STREAM)
client_socket.connect(('192.168.10.36', 12345))

message = "Hello Server"
client_socket.sendall(message.encode())

data = client_socket.recv(1024)
print("Received from server: ", data.decode())

client_socket.close()
```

Server

```
import socket

server_socket = socket.socket(socket.AF_INET, socket.SOCK_STREAM)
server_socket.bind(('192.168.10.36', 12345))
server_socket.listen(1)

while True:
```

```
    client_socket, client_address = server_socket.accept()
    print(f"Connection from {client_address}")

    data = client_socket.recv(1024)
    if not data:
        break
    data = data.upper()
    client_socket.sendall(data)
    client_socket.close()

server_socket.close()
```

```
PS D:\S8\404_ACN> py server.py
Connection from ('192.168.10.36', 55381)
```

```
PS D:\S8\404_ACN> py client.py
Received from server:  HELLO SERVER
```

7 Write a client server application program in Python using UDP 02-02-2024 CO2

<u>Client.py</u>

```
import socket
SERVER_HOST = '192.168.6.214'
SERVER_PORT = 12345
client_socket = socket.socket(socket.AF_INET, socket.SOCK_DGRAM)
while True:
        message = input("Enter message to send (type 'quit' to quit):")
        if message.lower() == 'quit':
        break

        client_socket.sendto(message.encode(), (SERVER_HOST, SERVER_PORT))

        data, _ = client_socket.recvfrom(1024)
        print('Received from server:', data.decode())

client_socket.close()
```

<u>Server.py</u>

```
import socket
SERVER_HOST = '192.168.6.214'
SERVER_PORT = 12345

server_socket =  socket.socket(socket.AF_INET, socket.SOCK_DGRAM)
server_socket.bind((SERVER_HOST, SERVER_PORT))
```

```
print(f"UDP server listening on {SERVER_HOST}:{SERVER_PORT}")
while True:
        data, client_address = server_socket.recvfrom(1024)
        print(f"Received from client at {client_address}:{data.decode()}")
        server_socket.sendto(data, client_address)
server_socket.close()
```

```
PS C:\NSA\blah> & "C:/Program Files/Python310/python.exe" c:/NSA/blah/udpserver.py
UDP server listening on 127.0.0.1:12345
Received data from ('127.0.0.1', 54329): hi
```

```
PS C:\NSA\blah> & "C:/Program Files/Python310/python.exe" c:/NSA/blah/udpclient.py
Enter the msg to send(type 'quit' to exit): hi
Received from server:hi
Enter the msg to send(type 'quit' to exit): quit
PS C:\NSA\blah>
```

8 Write a client server application program in Python using TCP 02-02-2024 CO2

Client.py

```
import socket

client_socket = socket.socket(socket.AF_INET, socket.SOCK_STREAM)
client_socket.connect(('192.168.6.214', 12345))

message = "Hello Server"
client_socket.sendall(message.encode())

data = client_socket.recv(1024)
print("Received from server:  ", data.decode())

client_socket.close()
```

Server.py

```
import socket

server_socket = socket.socket(socket.AF_INET, socket.SOCK_STREAM)
server_socket.bind(('192.168.6.214', 12345))
```

```
server_socket.listen(1)

while True:
    client_socket, client_address = server_socket.accept()
    print(f"Connection from {client_address}")

    data = client_socket.recv(1024)
    if not data:
        break
    client_socket.sendall(data)
    client_socket.close()

server_socket.close()
```

```
PS C:\NSA\blah> & "C:/Program Files/Python310/python.exe" c:/NSA/blah/tcpserevr.py
Connection from ('127.0.0.1', 50900)
```

```
PS C:\NSA\blah> & "C:/Program Files/Python310/python.exe" c:/NSA/blah/tcpclient.py
Enter message: HI HELLO
Received from Server hi hello
PS C:\NSA\blah>
```

9 Write a client server application program in Python for two – way chat using TCP
12-02-2024 CO2

Client-2Way.py

```
import socket

SERVER_HOST = '192.168.6.214'
SERVER_PORT = 12345

client_socket = socket.socket(socket.AF_INET, socket.SOCK_STREAM)

client_socket.connect((SERVER_HOST,SERVER_PORT))
print(f'Connected to server at {SERVER_HOST}:{SERVER_PORT}')
while True:
    message = input("Enter message to send (type 'quit' to exit): ")
    client_socket.sendall(message.encode())
```

```
    if message.lower() == 'quit':
        break
    response = client_socket.recv(1024).decode()
    print(f'Received from server: {response}')
client_socket.close()
```

Server-2Way.py

```
import socket
import threading

SERVER_HOST = '192.168.6.214'
SERVER_PORT = 12345

server_socket = socket.socket(socket.AF_INET, socket.SOCK_STREAM)
server_socket.bind((SERVER_HOST,SERVER_PORT))

server_socket.listen(5)
print(f'Server is listening on {SERVER_HOST}:{SERVER_PORT}')

def handle_client(client_socket, client_address):
    print(f"Connected to client at {client_address}")
    while True:
        data = client_socket.recv(1024).decode()
        if not data:
            break
        print(f'Received from client: {data}')
        response = input("Enter response to client: ")
        client_socket.sendall(response.encode())

    client_socket.close()
    print(f"Connection with client at {client_address} closed")

while True:
    client_socket, client_address = server_socket.accept()
    client_thread = threading.Thread(target=handle_client, args=(client_socket,client_address))
    client_thread.start()
```

```
PS C:\NSA\blah> & "C:/Program Files/Python310/python.exe" c:/NSA/blah/tcpserver-2way.py
Server listening on 127.0.0.1:12345
Connected to client at ('127.0.0.1', 64945)
Received from client :hi
Enter response to client:bye
Received from client :how are you
Enter response to client:fine, thankyou
```

```
PS C:\NSA\blah> & "C:/Program Files/Python310/python.exe" c:/NSA/blah/tcpclient-2way.py
Connected to server at 127.0.0.1:12345
Enter the msg to send(type 'quit' to exit): hi
Received from server:bye
Enter the msg to send(type 'quit' to exit): how are you
Received from server:fine, thankyou
Enter the msg to send(type 'quit' to exit):
```

10 Write a client server application program in Python for two – way chat using UDP 16-02-2024 CO2

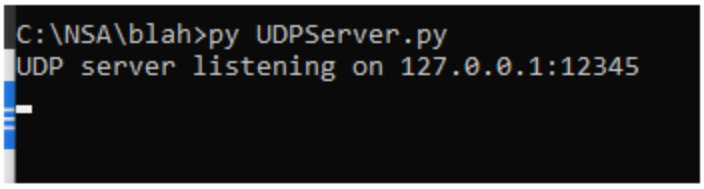
<u>UDPServer.py</u>

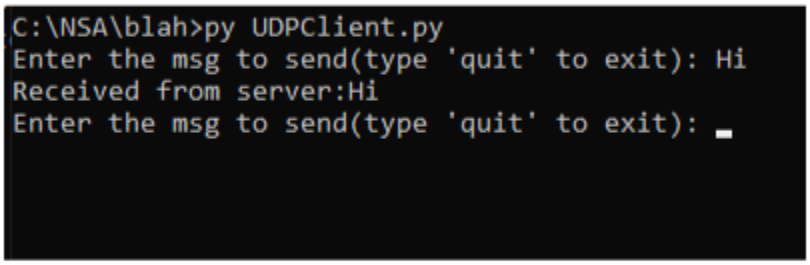
```
import socket
SERVER_HOST = '127.0.0.1'
SERVER_PORT = 12345
server_socket = socket.socket(socket.AF_INET, socket.SOCK_DGRAM)
server_socket.bind((SERVER_HOST, SERVER_PORT))
print('Server is running at {}:{}'.format(SERVER_HOST, SERVER_PORT))
while True:
data, client_address = server_socket.recvfrom(1024)
print(f"Received from Client at {client_address}: {data.decode()}")
if data.decode().lower() == "quit":
server_socket.sendto("quit".encode(), client_address)
break
response = input("Enter Response To Client: ")
server_socket.sendto(response.encode(), client_address)
server_socket.close()
print(f"Client at {client_address} disconnected")
```

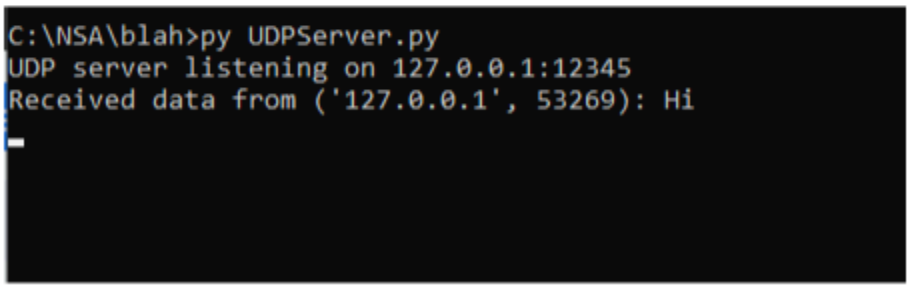
<u>UDPClient.py</u>

```
import socket
SERVER_HOST = '127.0.0.1'
SERVER_PORT = 12345
client_socket = socket.socket(socket.AF_INET, socket.SOCK_DGRAM)
print(f"Connected to server at {SERVER_HOST}:{SERVER_PORT}")
```

```
while True:
message = input("Enter a message to send (type 'quit' to exit): ")
client_socket.sendto(message.encode(), (SERVER_HOST, SERVER_PORT))
if message.lower() == "quit":
break
response, server_address = client_socket.recvfrom(1024)
print(f"Received From Server at {server_address}: {response.decode()}\n")
client_socket.close()
```







11 Implement TCP port checker in Python 19-02-2024 CO3

```
import socket
def check_tcp_port(host, port):
    try:
        with socket.socket(socket.AF_INET, socket.SOCK_STREAM) as s:
            s.settimeout(5)
            s.connect((host, port))
            print(f"Port {port} on {host} is open")
    except socket.error:
        print(f"Port {port} on {host} is closed")

if __name__=="__main__":
```

```
    host = input("Enter the host to check: ")
    port = int(input("Enter the port to check: "))
    check_tcp_port(host, port)
```

```
PS C:\Users\ajcemca\Documents\44\nsa> py TCP-portChk.py
Enter the host to check: 192.168.6.214
Enter the port to check: 80
Port 80 on 192.168.6.214 is open
```

12 Implement socket-based web server checker 19-02-2024 CO3

```
import socket
def check_web_server(host, port=80):
    client_socket = socket.socket(socket.AF_INET, socket.SOCK_STREAM)
    client_socket.settimeout(10)
    try:
        client_socket.connect((host, port))
        request = "GET / HTTP/1.1\r\nHost: " + host + "\r\n\r\n"
        client_socket.sendall(request.encode())

        response = client_socket.recv(4096)
        status_code = response.decode().split('\r\n')[0].split(' ')[1]
        print(f'Response from {host}:{port} - Status code: {status_code}')
    except socket.error as e:
        print(f'Error: {e}')
    finally:
        client_socket.close()

if __name__=="__main__":
    host = input("Enter the host or IP address: ")
    port = input("Enter the port number (default is 80 for HTTP and 443 for HTTPS): ")
    if port:
        port = int(port)
    else:
        port = 80
    check_web_server(host, port)
```

```
PS C:\NSA\blah> & "C:/Program Files/Python310/python.exe" c:/NSA/blah/check_web_server.py
Enter the host or IP address:192.168.56.1
Enter the port number(default is 80):80
Response from 192.168.56.1:80 - Status code: 302
```

13 Implement httplib-based web server checker 23-02-2024 CO3

```
import http.client
def check_web_server_httplib(host, port=80):
    try:
        conn = http.client.HTTPConnection(host, port, timeout=10)
        conn.request("GET", "/")
        response = conn.getresponse()
        print(f"Reponse from {host}:{port} - Status code: {response.status}")
        conn.close()
    except ConnectionRefusedError:
        print(f"Connection to {host}:{port} refused")
    except Exception as e:
        print(f"An error occured: {str(e)}")

if __name__=="__main__":
    host = input("Enter the host to check: ")
    port = input("Enter the port number (default is 80): ")
    if port:
        port = int(port)
    else:
        port = 80
    check_web_server_httplib(host, port)
```

```
PS C:\Users\ajcemca\Documents\44\nsa> py check_web_server_httplib.py
Enter the host to check: www.google.com
Enter the port number (default is 80): 80
Reponse from www.google.com:80 - Status code: 200
```

14 Implement a simple Pyro server 23-02-2024 CO3

PyroServer.py

```
# py -m Pyro4.naming

import Pyro4
class MyServer(object):
    @Pyro4.expose
    def greet(self, name):
        return "Hello, {}".format(name)

daemon = Pyro4.Daemon()

uri = daemon.register(MyServer)
print(f"Server URI: {uri}")

ns = Pyro4.locateNS()
```

```
ns.register("myserver",uri)
print("Server is ready")

daemon.requestLoop()
```

PyroClient.py

```
import Pyro4
ns = Pyro4.locateNS()
uri = ns.lookup("myserver")
server = Pyro4.Proxy(uri)
name = input(f"Enter your name: ")
message = server.greet(name)
print("Message from server: ", message)
```

```
PS C:\Users\ajcemca\Documents\44\nsa> py -m Pyro4.naming
Not starting broadcast server for localhost.
NS running on localhost:9090 (127.0.0.1)
Warning: HMAC key not set. Anyone can connect to this server!
URI = PYRO:Pyro.NameServer@localhost:9090
```

```
PS C:\Users\ajcemca\Documents\44\nsa> py PyroServer.py
Server URI: PYRO:obj_a7b3dc4262fb4ef28a479f9fd7d2c9d8@localhost:62426
Server is ready
```

```
PS C:\Users\ajcemca\Documents\44\nsa> py PyroClient.py
Enter your name: Navya
Message from server:  Hello, Navya
```

15 Using Pyro Server implement a simple calculator 26-02-2024 CO3

PyroServer.py

```
# py -m Pyro4.naming

import Pyro4
class MyServer(object):
    @Pyro4.expose
    def greet(self, name):
        return "Hello, {}".format(name)
    @Pyro4.expose
    def calc(self,a,b):
        add = a+b
        sub = a-b
        pdt = a*b
        quo = a/b
        return f"Sum: {add}, Difference: {sub}, Product: {pdt}, Quotient: {quo}"
```

```
daemon = Pyro4.Daemon()

uri = daemon.register(MyServer)
print(f"Server URI: {uri}")

ns = Pyro4.locateNS()
ns.register("myserver",uri)
print("Server is ready")

daemon.requestLoop()
```

PyroClient.py

```
import Pyro4
ns = Pyro4.locateNS()
uri = ns.lookup("myserver")
server = Pyro4.Proxy(uri)
name = input(f"Enter your name: ")
message = server.greet(name)
print("Message from server: ", message)

a = int(input("Enter a number: "))
b = int(input("Enter another number: "))
result = server.calc(a,b)
print(result)
```

```
PS C:\Users\ajcemca\Documents\44\nsa> py PyroClient.py
Enter your name: Navya
Message from server:  Hello, Navya
Enter a number: 10
Enter another number: 2
Sum: 12, Difference: 8, Product: 20, Quotient: 5.0
PS C:\Users\ajcemca\Documents\44\nsa>
```

16 Using Pyro Server check whether the number is Prime or Not 26-02-2024 CO3

PyroServer.py

```
# py -m Pyro4.naming

import Pyro4

class MyServer(object):
    @Pyro4.expose
    def prime(self, num):
```

```
        if num < 2:
            return False
        for i in range(2, int(num**0.5) + 1):
            if num % i == 0:
                return False
        return True

daemon = Pyro4.Daemon()

uri = daemon.register(MyServer)
print("Server URI : ", uri)
ns = Pyro4.locateNS()
ns.register("myserver", uri)
print("Server is Ready")
daemon.requestLoop()
```

PyroClient.py

```
import Pyro4
ns = Pyro4.locateNS()
uri = ns.lookup("myserver")
server = Pyro4.Proxy(uri)
num = int(input("Enter the Number to check: "))
message = server.prime(num)
if message:
    print("The entered number is prime")
else:
    print("The entered number is not prime")
```

```
C:\Program Files\Python310\Scripts>py -m Pyro4.naming
Not starting broadcast server for localhost.
NS running on localhost:9090 (127.0.0.1)
Warning: HMAC key not set. Anyone can connect to this server!
URI = PYRO:Pyro.NameServer@localhost:9090
```

```
C:\NSA\blah>py PyroServerPrime.py
Server URI :  PYRO:obj_9fb94035d20b44b09e799892c43be5c6@localhost:54644
Server is Ready
```

```
C:\NSA\blah>py PyroClientPrime.py
Enter the Number to check: 7
The entered number is prime

C:\NSA\blah>
```

17 Simple XML-RPC server 01-03-2024 CO3

SimpleXMLRPCServer.py

```
SimpleXMLRPCServer.py > ...
from xmlrpc.server import SimpleXMLRPCServer
def add(x,y):
    return x+y
server= SimpleXMLRPCServer(("localhost",8000))
print("XML-RPC Server is running on port 8000.......")

server.register_function(add, "add")
server.serve_forever()
```

SimpleXMLRPCClient.py

```
SimpleXMLRPCClient.py > ...
import xmlrpc.client
server=xmlrpc.client.ServerProxy("http://localhost:8000")
result=server.add(3,5)
print("Result of remote function call: " , result)
```

output

```
PS C:\NSA\blah> & "C:/Program Files/Python310/python.exe" c:/NSA/blah/SimpleXMLRPCServer.py
XML-RPC Server is running on port 8000.......
127.0.0.1 - - [16/Feb/2024 12:18:31] "POST /RPC2 HTTP/1.1" 200 -
```

```
PS C:\NSA\blah> & "C:/Program Files/Python310/python.exe" c:/NSA/blah/SimpleXMLRPCClient.py
Result of remote function call:  8
PS C:\NSA\blah>
```

18 XML-RPC server to implement a simple calculator 01-03-2024 CO3

SimpleXMLRPCServer.py

```
from xmlrpc.server import SimpleXMLRPCServer
def add(x, y):
    return x + y
def sub(x, y):
    return x - y
def mul(x, y):
    return x * y
server = SimpleXMLRPCServer(("localhost",8000))
print("XML-RPC server is running on port 8000...")
server.register_function(add, "add")
server.register_function(sub, "sub")
server.register_function(mul, "mul")
server.serve_forever()
```

SimpleXMLRPCServer_Client.py

```
import xmlrpc.client
server = xmlrpc.client.ServerProxy("http://localhost:8000")
result1 = server.add(3,5)
print("Result of remote function call: ", result1)
result2 = server.sub(3,5)
print("Result of remote function call: ", result2)
result3 = server.mul(3,5)
print("Result of remote function call: ", result3)
```

```
PS C:\Users\ajcemca\Documents\44\nsa> py SimpleXMLRPCServer.py
XML-RPC server is running on port 8000...
127.0.0.1 - - [16/Feb/2024 12:32:12] "POST /RPC2 HTTP/1.1" 200 -
127.0.0.1 - - [16/Feb/2024 12:32:14] "POST /RPC2 HTTP/1.1" 200 -
127.0.0.1 - - [16/Feb/2024 12:32:16] "POST /RPC2 HTTP/1.1" 200 -
```

```
PS C:\Users\ajcemca\Documents\44\nsa> py SimpleXMLRPCServer_Client.py
Result of remote function call:  8
Result of remote function call:  -2
Result of remote function call:  15
```

19 Connecting to an SSH server and remotely executing a command 04-03-2024 CO3

```
import paramiko
ssh_client=paramiko.SSHClient()
ssh_client.set_missing_host_key_policy(paramiko.AutoAddPolicy())
ssh_client.connect(hostname='192.168.6.214',username='student',password='student')
stdin,stdout,stderr=ssh_client.exec_command('ls -l')
output=stdout.read().decode()
print(output)
ssh_client.close()
```

```
student@u45:~$ /bin/python3 /home/student/Documents/ssh_command.py
total 44
drwxr-xr-x 3 student student 4096 Feb 12 14:35 Desktop
drwxr-xr-x 2 student student 4096 May  5  2023 Documents
drwxr-xr-x 2 student student 4096 May  5  2023 Downloads
drwxr-xr-x 2 student student 4096 May  5  2023 Music
drwxr-xr-x 2 student student 4096 May  5  2023 Pictures
drwxr-xr-x 2 student student 4096 May  5  2023 Public
drwxrwxr-x 3 student student 4096 Feb 12 15:00 PycharmProjects
drwx------ 4 student student 4096 Feb 19 14:53 snap
-rw-rw-r-- 1 student student  312 Feb 19 15:02 ssh_command.py
drwxr-xr-x 2 student student 4096 May  5  2023 Templates
drwxr-xr-x 2 student student 4096 May  5  2023 Videos
```

20 Using the shutil module to copy a data tree 11-03-2024 CO4

```
import shutil
src = 'shutil_source'
dest = 'shutil_dest'
shutil.copytree(src, dest)
print(f"Contents of {src} successfully copied to {dest}.")
```

```
PS C:\Users\ajcemca\Documents\44\nsa> py shutil_copy_datatree.py
Contents of shutil_source successfully copied to shutil_dest.
```

21 Moving a data tree with shutil 11-03-2024 CO4

```
import shutil
src = 'shutil_source'
dest = 'shutil_dest'
shutil.move(src, dest, copy_function=shutil.copy2)
print(f"Contents of {src} successfully moved to {dest}.")
```

```
PS C:\Users\ajcemca\Documents\44\nsa> py shutil_move_datatree.py
Contents of shutil_source successfully moved to shutil_dest.
```

22 Performing an MD5 checksum on files. 15-03-2024 CO4

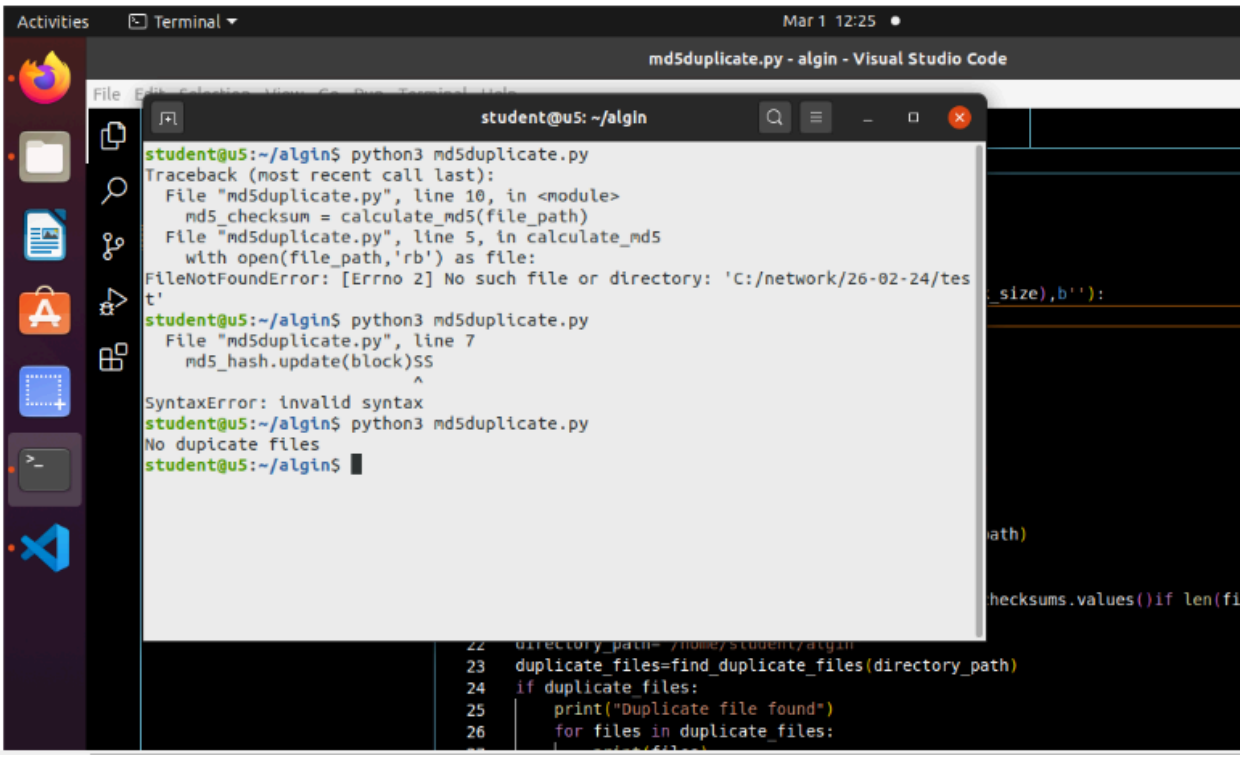
```
import hashlib
def calculate_md5(file_path, block_size = 65536):
    md5_hash = hashlib.md5()
    with open(file_path, 'rb') as file:
        for block in iter (lambda: file.read(block_size), b''):
            md5_hash.update(block)
    return md5_hash.hexdigest()

file_path = 'md5_hash.py'
md5_checksum = calculate_md5(file_path)
print("MD5 checksum of {} : {}".format(file_path, md5_checksum))
```

```
PS C:\Users\ajcemca\Documents\44\nsa> py md5_hash.py
MD5 checksum of md5_hash.py : c8eba47654f349ecc2f0e059817a9568
```

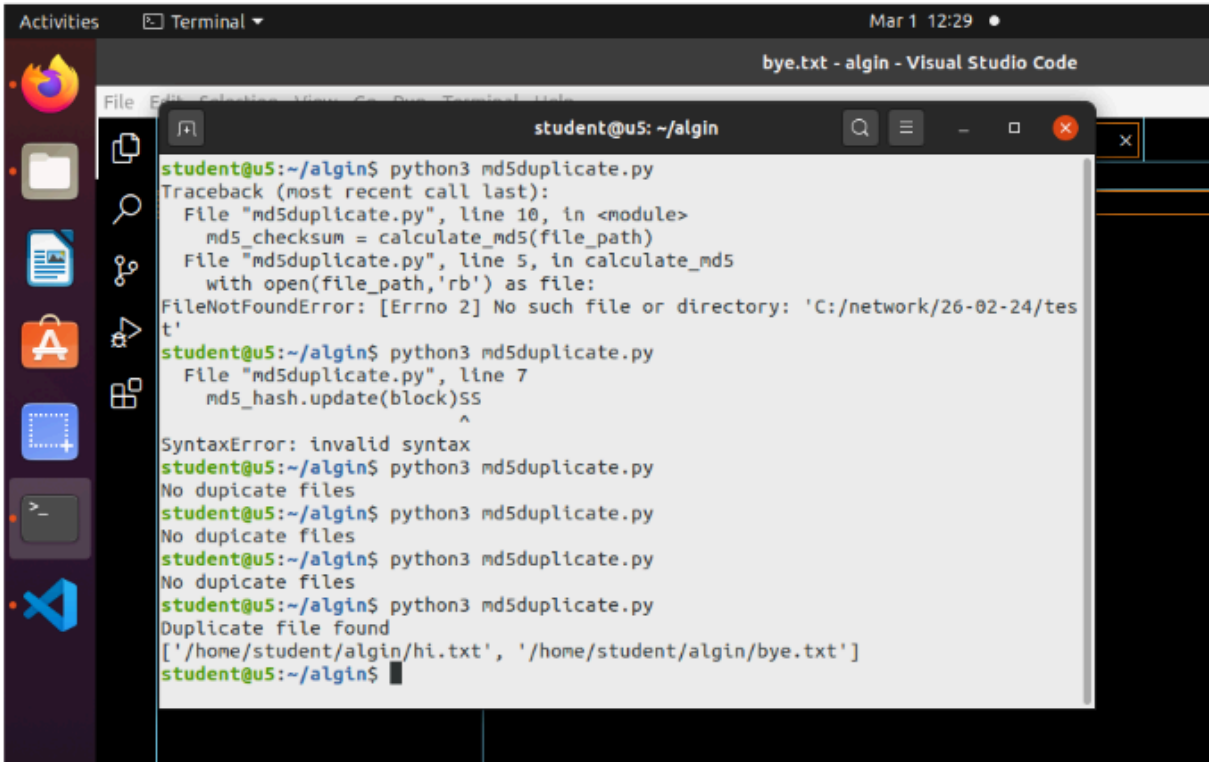
23 Performing an MD5 checksum on a directory tree to find Duplicates 15-03-2024 CO4

```
import os
import hashlib
def calculate_md5(file_path,block_size=65536):
md5_hash = hashlib.md5()
with open(file_path,'rb') as file:
for block in iter(lambda: file.read(block_size),b''):
md5_hash.update(block)
return md5_hash.hexdigest()
file_path = '/home/student/algin/hi.txt'
def find_duplicate_files(directory):
checksums={}
for root,_,files in os.walk(directory):
for filename in files:
file_path=os.path.join(root,filename)
checksum=calculate_md5(file_path)
if checksum in checksums:
checksums[checksum].append(file_path)
else:
checksums[checksum]=[file_path]
duplicate_files=[file_list for file_list in checksums.values()if
len(file_list)>1]
return duplicate_files
```

```
directory_path='/home/student/algin'
duplicate_files=find_duplicate_files(directory_path)
if duplicate_files:
print("Duplicate file found")
for files in duplicate_files:
print(files)
else:
print("No duplicate files")
```



when bye.txt and hi.txt had same contents



24 Basic data center discovery - Simulation 18-03-2024 CO5

```
class Server:
    def __init__(self,name,cpu_capacity,memory_capacity):
        self.name = name
        self.cpu_capacity = cpu_capacity
        self.memory_capacity = memory_capacity
        self.cpu_usage = 0
        self.memory_usage = 0

    def allocate_resource(self,cpu,memory):
        if self.cpu_capacity - self.cpu_usage >= cpu and self.memory_capacity -
self.memory_usage >= memory:
            self.cpu_usage += cpu
            self.memory_usage += memory
            return True
        else:
            return False

    def release_resource(self,cpu,memory):
        self.cpu_usage -= cpu
        self.memory_usage -= memory
```

```
class DataCenter:
    def __init__(self):
        self.servers = []
    def add_server(self, server):
        self.servers.append(server)
    def allocate_resource(self,cpu,memory):
        for server in self.servers:
            if server.allocate_resource(cpu,memory):
                return server.name
        return None
dc = DataCenter()

dc.add_server(Server("Server1",4,16))
dc.add_server(Server("Server2",8,32))
dc.add_server(Server("Server3",2,8))

workloads = [(2, 8), (4, 16), (1, 4)]
for cpu, memory in workloads:
    server_name = dc.allocate_resource(cpu, memory)
    if server_name:
        print(f"Workload ({cpu} CPU, {memory} Memory) allocated to {server_name}")
    else:
        print(f"Workload ({cpu} CPU, {memory} Memory) could not be allocated.")
```

```
Workload (2 CPU, 8 Memory) allocated to Server1
Workload (4 CPU, 16 Memory) allocated to Server2
Workload (1 CPU, 4 Memory) allocated to Server1

[Done] exited with code=0 in 0.29 seconds
```

25 Basic Net-SNMP operations 08-04-2024 CO5
26 SNMP configuration file with Hello World 08-04-2024 CO5