# Kanani Njaramba EBU Python Project Port Scanner Documentation

## Introduction

This Python script serves as a basic port scanner. It reads a list of hosts and corresponding ports from a CSV file, attempts to establish a TCP connection to each host on the specified port, and records whether the connection was successful or not. Additionally, it generates a pie chart to visualize the distribution of open and closed ports.

## Usage

1. Ensure you have the required libraries installed: `csv`, `socket`, and `matplotlib`.

2. Prepare a CSV file with a list of hosts and ports. The CSV file should have headers 'host' and 'port' respectively.

3. Modify the file path in the `csv\_file` variable to point to your CSV file.

4. Execute the script in a Python environment.

## Code Description

### Importing Libraries

```python

import csv

import socket

import matplotlib.pyplot as plt

```

### Initializing Counters and Lists

```python

open\_ports = 0

closed\_ports = 0

open\_port\_details = []

closed\_port\_details = []

```

These variables are used to keep track of the number of open and closed ports, as well as to store details about each.

### Reading CSV File

```python

with open('server\_list.csv', 'r') as csv\_file:

csv\_reader = csv.DictReader(csv\_file)

# ...

```

This block of code opens the CSV file containing the list of hosts and ports. It then iterates through each row in the CSV file.

### Port Scanning Loop

```python

for row in csv\_reader:

host = row['host']

port = int(row['port'])

s = socket.socket(socket.AF\_INET, socket.SOCK\_STREAM)

result = s.connect\_ex((host, port))

# ...

```

For each row in the CSV file, this loop extracts the host and port information. It then attempts to establish a TCP connection to the host on the specified port.

### Handling Connection Results

```python

if result == 0:

print(f"Connected to {host}:{port}")

open\_ports += 1

open\_port\_details.append((host, port))

else:

print(f"Failed to connect to {host}:{port}")

closed\_ports += 1

closed\_port\_details.append((host, port))

s.close()

```

Based on the result of the connection attempt, the script updates the counters and appends details to the respective lists.

### Exporting Results to CSV Files

```python

with open('open\_ports.csv', 'w', newline='') as open\_file:

open\_writer = csv.writer(open\_file)

open\_writer.writerow(['Host', 'Port'])

open\_writer.writerows(open\_port\_details)

with open('closed\_ports.csv', 'w', newline='') as closed\_file:

closed\_writer = csv.writer(closed\_file)

closed\_writer.writerow(['Host', 'Port'])

closed\_writer.writerows(closed\_port\_details)

```

This section exports the open and closed port details to separate CSV files.

### Generating Pie Chart

```python

labels = 'Open Ports', 'Closed Ports'

sizes = [open\_ports, closed\_ports]

colors = ['green', 'red']

explode = (0.1, 0)

plt.pie(sizes, explode=explode, labels=labels, colors=colors, autopct='%1.1f%%', shadow=True, startangle=140)

plt.axis('equal')

```

This block prepares data for the pie chart and sets its appearance.

### Displaying Pie Chart

```python

plt.show()

```

Finally, this command displays the pie chart.

## Conclusion

This script provides a basic example of a port scanner in Python. It can be used to scan a list of hosts for open ports and visualize the results using a pie chart.

```