Loop PY - Line by Line Breakdown

```
import socket
```

Loads the socket module for network connections (IP/Port level).

```
import time
```

Allows tracking start and end times for the scan duration.

```
import threading
```

Enables multithreading so you can scan multiple ports simultaneously.

```
def scan_port(target, port):
```

Defines a function to scan a single port on a target host.

```
    try:
```

Starts a try block to catch any connection errors safely.

```
        sock = socket.socket(socket.AF_INET, socket.SOCK_STREAM)
```

Creates a TCP socket using IPv4.

```
        sock.settimeout(0.5)
```

Limits wait time for the port response to 0.5 seconds.

```
        result = sock.connect_ex((target, port))
```

Attempts connection to the port. Result 0 = open.

```
        if result == 0:
```

If the port is open, print that its open.

```
        print(f"Port {port} is OPEN ")
```

```
    sock.close()
```

Closes the socket to clean up the connection.

```
except:
```

Prevents crashes by catching errors silently.

```
    pass
```

```
target = input("Enter an IP address or hostname to scan: ").strip()
```

Takes target IP input and removes any leading/trailing spaces.

```
if not target:
```

Checks if the user submitted an empty string.

```
    print(" No IP or hostname entered. Exiting...")
```

Message if the target input is empty.

```
    exit()
```

Stops the script if no valid target is given.

```
start_port = int(input("Enter the starting port number: "))
```

Asks for the starting port number and converts it to an integer.

```
end_port = int(input("Enter the ending port number: "))
```

Asks for the ending port number.

```
start_time = time.time()
```

Captures the current time before the scan starts.

```
for port in range(start_port, end_port + 1):
```

Loops over each port in the specified range.

```
    thread = threading.Thread(target=scan_port, args=(target, port))
```

Creates a new thread for scanning a port.

```
    thread.start()
```

Starts the thread to perform the scan on that port.

```
end_time = time.time()
```

Captures the time after all scan threads have started.

```
duration = end_time - start_time
```

Calculates the total scan time.

```
print(f"\nScan completed in {duration:.2f} seconds.")
```

Prints how long the scan took.