



**School of Engineering and Technology**

**Department of AIML & DS**

# **Serve a Directory using Python**

**Name: Darain Brit A**

**Registration Number: 2462060**

**Subject: Ethical Hacking**

**Subject Code: CSHO331CSP**

**Academic Year: 2025-2026**

## Title

SL no.	Content	Page Number
1.	Aim	3
2.	Methodology	3-4
3.	Findings	5
4.	Conclusion	5

# Assignment 12: Serve a Directory using Python

## Aim:

To turn a computer into a simple web server to share files over the internet using Python's built-in HTTP server module.

## Methodology:

1. Create a directory named webserver using the mkdir command:

```
mkdir webserver
```

2. Navigate into the directory using the cd command:

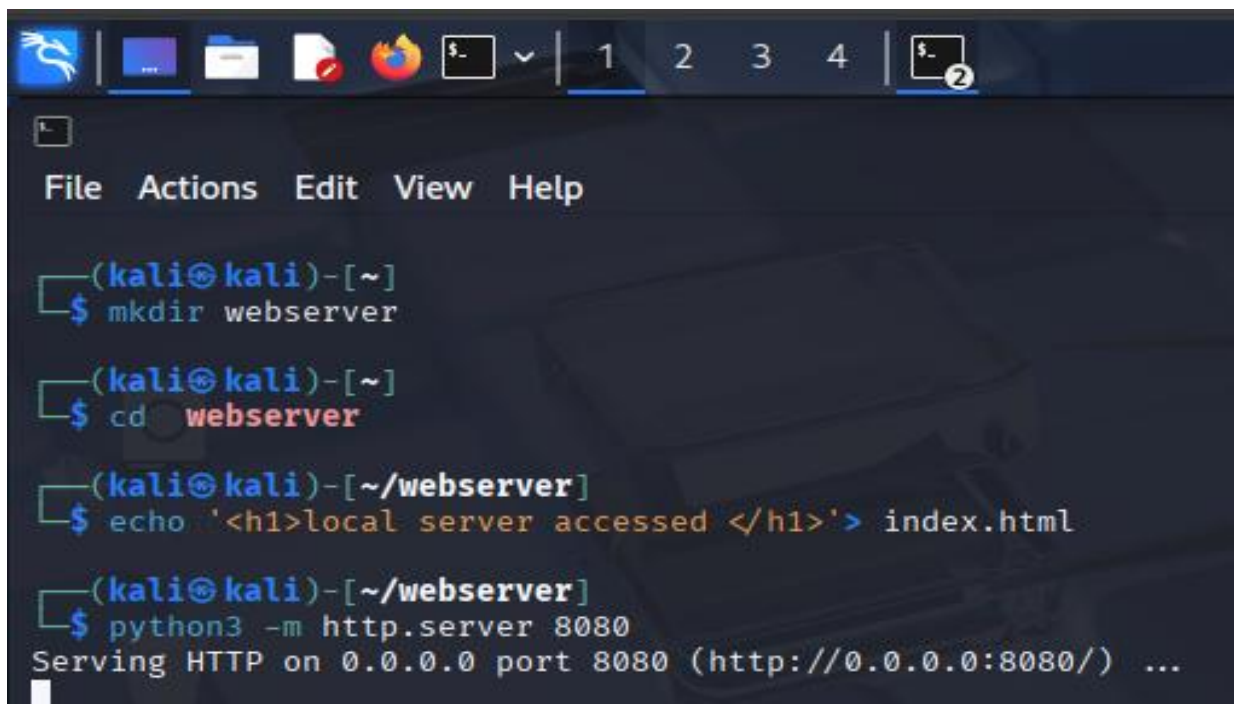
```
cd webserver
```

3. Create an HTML file named index.html so that it loads automatically when accessed in a browser:

```
echo "<h1>local server accessed</h1>" > index.html
```

4. Run the Python HTTP server on port 8080:

```
python3 -m http.server 8080
```

A screenshot of a Kali Linux terminal window. The terminal has a dark background with a menu bar at the top containing 'File', 'Actions', 'Edit', 'View', and 'Help'. The terminal shows the following commands and output:  
1. Prompt: (kali@kali)-[~]  
Command: \$ mkdir webserver  
2. Prompt: (kali@kali)-[~]  
Command: \$ cd webserver  
3. Prompt: (kali@kali)-[~/webserver]  
Command: \$ echo '<h1>local server accessed </h1>'> index.html  
4. Prompt: (kali@kali)-[~/webserver]  
Command: \$ python3 -m http.server 8080  
Output: Serving HTTP on 0.0.0.0 port 8080 (http://0.0.0.0:8080/) ...  
The terminal window has a taskbar at the top with icons for a web browser, file manager, and other applications. The terminal output is in a monospaced font with syntax highlighting.

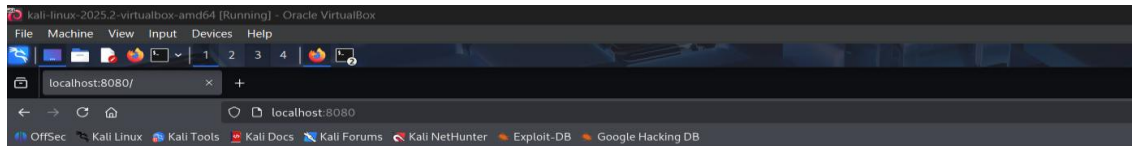
5. Open a web browser and enter:

`http://localhost:8080`

6. If the file is not named `index.html`, it must be accessed with its name in the URL,

`http://localhost:8080/filename.html`

The `index.html` file will be displayed.



**local server accessed**

7. Observe the terminal output, which will show the web server's log entries for each request made.

```
(kali㉿kali)-[~/webserver]
$ mkdir webserver

(kali㉿kali)-[~/]
$ cd webserver

(kali㉿kali)-[~/webserver]
$ echo '<h1>local server accessed </h1>'> index.html

(kali㉿kali)-[~/webserver]
$ python3 -m http.server 8080
Serving HTTP on 0.0.0.0 port 8080 (http://0.0.0.0:8080/) ...
127.0.0.1 - - [30/Jul/2025 10:52:53] "GET / HTTP/1.1" 200 -
```

## Findings:

- Python's built-in HTTP server is useful for quickly testing or sharing files locally.
- The server lacks authentication and encryption, making it unsuitable for public use.
- Without an index.html file, the server displays a directory listing, which can expose sensitive file names.

## Conclusion:

Through this experiment, I learned how to:

- Use Python's built-in HTTP server to expose files over a network.
- Understand how clients send HTTP GET requests and how servers respond.
- Read and interpret server log entries.

This practical exercise demonstrated the basics of serving static files over HTTP and highlighted potential security risks.