

var

Variables in Python**

A **variable** is a **name** that stores a value in memory. You can think of it as a **box** where you store information.

```
variable_name = value
```

```
name = "Trishant" age = 22 height = 5.9 is_hacker = True
```

👉 Here, `name` , `age` , `height` , and `is_hacker` are variables storing different types of values.

Data Types in Python

Python automatically assigns a **data type** to a variable based on the value stored. Here are the main types:

Data Type	Example	Use Case
<code>str</code> (String)	<code>"hello"</code>	Storing text (names, messages)
<code>int</code> (Integer)	<code>42</code>	Counting, loops, indexing
<code>float</code> (Floating Point)	<code>3.14</code>	Storing decimal numbers (height, weight)
<code>bool</code> (Boolean)	<code>True</code> , <code>False</code>	Conditional checks (<code>if</code> statements)

List + Dictionary

Lists: Ordered Collection of Items

Definition:

List ek **ordered collection** hoti hai jo multiple values store kar sakti hai. Ye **mutable** hoti hai (iska data change ho sakta hai).

hack

```
list_name = [item1, item2, item3, ...]
```

```
hack_tools = ["Nmap", "Burp Suite", "Metasploit", "Wireshark"] print(hack_tools)
```

List Operations

```
# Accessing Elements print(hack_tools[0])    # 'Nmap'   (First element)

print(hack_tools[-1])  # 'Wireshark' (Last element)

# Modifying List

hack_tools.append("SQLmap")  # Add element

hack_tools.remove("Burp Suite")

# Remove element

hack_tools[1] = "Aircrack-ng"  # Modify element  print(hack_tools)

['Nmap', 'Aircrack-ng', 'Metasploit', 'Wireshark', 'SQLmap']
```

Use Case:

Jab ek **list of items** store karni ho, jaise **ports, usernames, IPs**.

Dictionaries: Key-Value Pairs

Definition:

Dictionaries **key-value pairs** store karti hain. Iska use **structured data** rakhne ke liye hota hai, jaise **user info, settings, or response data**.

```
dict_name = {"key1": value1, "key2": value2, ...}
```

```
`hacker_info = {

"name": "CyberGhost",
"skills": ["Pentesting", "Scripting", "Exploit Dev"],
"age": 22,      "certified": True

}

print(hacker_info)
`

#output
`{'name': 'CyberGhost', 'skills': ['Pentesting', 'Scripting', 'Exploit Dev'],
'age': 22, 'certified': True}`
```

Dictionary Operations

python

Copy code

```
# Accessing Values print(hacker_info["name"]) # 'CyberGhost'
print(hacker_info["skills"]) # ['Pentesting', 'Scripting', 'Exploit Dev'] #
Modifying Dictionary hacker_info["experience"] = "3 Years" # Add new key
hacker_info["age"] = 23 # Update value del hacker_info["certified"] # Delete key
print(hacker_info)
```

Output:

```
{'name': 'CyberGhost', 'skills': ['Pentesting', 'Scripting', 'Exploit Dev'],
'age': 23, 'experience': '3 Years'}
```

Use Case:

Jab kisi **entity ke details** store karni ho, jaise **user credentials, config files, JSON API responses**.

Difference Between List & Dictionary

Feature	List	Dictionary
Data Type	Ordered Collection	Key-Value Pairs
Indexing	Numeric (<code>list[0]</code>)	Key-based (<code>dict["key"]</code>)
Mutability	Mutable	Mutable
Best For	Storing multiple values	Storing structured data

Practice Challenge

```
`ports = [80, 443, 22, 8080]
```

```
network_info = {"IP": "192.168.1.1", "Open_Ports": ports}

print(network_info["IP"])

print(network_info["Open_Ports"][1])`
```

arithmetic

1. Arithmetic Operators

Use hota hai maths calculations me

```
a = 10
b = 3

print(a + b) # 13 (Addition)
print(a - b) # 7 (Subtraction)
print(a * b) # 30 (Multiplication)
print(a / b) # 3.333 (Division)
print(a // b) # 3 (Floor Division - integer result)--->returns without . like
2.46 =2
print(a % b) # 1 (Modulo - remainder)
print(a ** b) # 1000 (Exponent - Power)
```

Use Case:

Password hashing me **exponents** ka use hota hai ($x ** y$).

Modulo use hota hai **even-odd check** me ($x \% 2$).

logical

3. Logical Operators

Boolean values (True/False) ke sath kaam karte hain

```
a = True  
b = False
```

`print(a and b)` # False (Dono true hone chahiye)

```
print(a or b)    # True   (Koi ek true ho)  
print(not a)     # False  (Negation)
```

Use Case:

Multiple conditions check karne ke liye jaise `if user == "admin" and password_correct` .

ID operator

Identity Operators

Check karta hai ki do variables same memory location point kar rahe hain ya nahi

```
`a = [1, 2, 3] b = a # Both point to same list print(a is b) # True (Same object) print(a is not b) # False`
```

Use Case:

Object comparison me use hota hai, jaise **cache data validation**.

comparision

```
x = 10
y = 5

print(x == y)  # False (Equal)
print(x != y)  # True  (Not Equal)
print(x > y)   # True  (Greater than)
print(x < y)   # False (Less than)
print(x >= y)  # True  (Greater or Equal)
print(x <= y)  # False (Less or Equal)
```

value compare karne k liye

Login System me **password check** karne ke liye (`entered_pass == stored_pass`).

BITWISE OPERATORS

Bitwise Operators (For Advanced Users)

Binary level pe operations perform karta hai

```
x = 5 # 101 in binary y = 3 # 011 in binary print(x & y) # 1 (AND)
print(x | y) # 7 (OR)
print(x ^ y) # 6 (XOR)
print(~x) # -6 (NOT) |
print(x << 1) # 10 (Left Shift)
print(x >> 1) # 2 (Right Shift)`
```

Use Case:

Encryption & compression algorithms me use hota hai.

MEMBERSHIP

Check karta hai ki koi value list ya string me hai ya nahi

python

CopyEdit

```
`hack_tools = ["Nmap", "Burp Suite", "Metasploit"] print("Nmap" in  
hack_tools)    # True print("Wireshark" not in hack_tools) # True`
```

Use Case:

Dictionary me key check karne ke liye (if "username" in user_data:).

PRACTICE

```
x = 8
y = 3

result = (x % y == 2) and (x // y > 2)
print(result)
```

FALSE

basics

- **Sequence = Ek ordered collection** jisme multiple values hoti hain. (e.g., `list` , `string` , `tuple` , `range`)
- **Iterate = Ek-ek karke har element par kaam karna.** (Loop ke andar sequence ke elements ko access karna)

MINI-PROJECT

```
pin = 1234
entered_pin = int(input("Enter your PIN: "))

if entered_pin == pin:
    print("    Access Granted! Welcome.")
else:
    print("    Incorrect PIN! Try again.")
```

BASIC

if	Jab sirf ek condition check karni ho
if-else	Jab ek True aur ek False case ho
if-elif-else	Jab multiple conditions check karni ho
Nested if	Jab ek if ke andar aur if lagana ho
Conditional Statement	Use Case

```
age = 20

if age >= 18:
    print("You are eligible to vote!")
```

IF-ELSE

```
age = 16

if age >= 18:
    print("You are eligible to vote!")
else:
    print("You are NOT eligible to vote!")
```

IF-ELIF-ELSE

```
marks = 75

if marks >= 90:
    print("Grade: A")
elif marks >= 70:
    print("Grade: B")
elif marks >= 50:
    print("Grade: C")
else:
```

```
print("Grade: F")
```

NESTED IF-ELSE

```
age = 20
has_voter_id = True

if age >= 18:
    if has_voter_id:
        print("You can vote!")
    else:
        print("Get your voter ID first!")
else:
    print("You are too young to vote!")
```

CONTROL STATEMENTS

Statement	Explanation
break	Loop ko turant stop kar deta hai
continue	Current iteration skip karta hai, loop continue rahta hai
pass	Kuch nahi karta, bas syntax error se bachne ke liye

BREAK

```
for i in range(1, 6):  
    if i == 3:  
        break # Loop yaha ruk jayega  
    print(i)
```

CONTINUE

```
for i in range(1, 6):  
    if i == 3:  
        continue # 3 skip ho jayega  
    print(i)
```

PASS

```
for i in range(1, 6):  
    if i == 3:  
        pass # Yeh kuch nahi karega, bas syntax error se bachne ke liye  
    print(i)
```

Quick Recap

Loop Type	Use Case
for loop	Jab kisi sequence (list, range, string) pe iterate karna ho
while loop	Jab condition true hone tak loop chalana ho
Nested loops	Jab ek loop ke andar doosra loop chalana ho
break	Loop ko turant exit karne ke liye
continue	Current iteration skip karne ke liye
pass	Placeholder statement, kuch nahi karta

PRO TIPS ---->>>>>>

Loop Type	Use Case (Hacking)
for loop	Jab fixed repetitions pata ho, jaise brute force attack, wordlist processing, port scanning
while loop	Jab condition-based repetition ho, jaise infinite reverse shell, continuous listener, login attempts

for loop

```
for variable in sequence:  
    # Loop body (yeh har iteration pe chalega)
```

with list

```
fruits = ["apple", "banana", "mango"]  
  
for fruit in fruits:  
    print(fruit)
```

with string

```
for char in "HACKER":  
    print(char)
```

H
A
C
K
E
R

WITH RANGE FUNCTION

FOR PARTICULAR RANGE TAK LOOP CHLANA NA HO

```
for i in range(1, 6): # 1 se 5 tak chalega  
    print(i)
```

While LOOP

```
while condition:  
    # Loop body (yeh tab tak chalega jab tak condition True hai)
```

```
x = 1
```

```
while x <= 5:  
    print(x)  
    x += 1 # x = x + 1 (Loop counter update karna zaroori hai!)
```

PERFECT EXAMPLE

```
password = "hacker123"  
user_input = ""  
  
while user_input != password:  
    user_input = input("Enter password: ")  
  
print("    Access Granted!")
```

functions and args and parameter

1. Function Basics - Definition & Calling

Function ek reusable code block hota hai jo ek specific kaam karta hai.

Function define karna:

```
def greet(): print("Hello, Hacker!") # Function ka kaam
```

Parameters vs Arguments

Parameters: Function define karte waqt jo variables use hote hain.

Arguments: Function call karte waqt jo values pass karte hain.

```
def greet(name): # 'name' yahan parameter hai
    print(f"Hello, {name}!")

    print(f"Hello, {name}!")

greet("Trishant") # 'Trishant' argument hai
```

Variable-Length Arguments (`*args` & `**kwargs`)**

Jab tu nahi pata kitne arguments milenge, tab use karte hain.

`*args` (Multiple Positional Arguments)

```
def ports_scan(*ports):
    for port in ports:
        print(f"Scanning port {port}...")

ports_scan(22, 80, 443)
```

`**kwargs` (Multiple Keyword Arguments)

```
def user_info(**info):  
    for key, value in info.items():  
        print(f"{key}: {value}")  
  
user_info(name="Hacker", skill="Cybersecurity", level="Advanced")
```

Return Statement (Function Ka Output Wapas Bhejna)

Agar function ka output kisi aur function me use karna ho toh `return` ka use hota hai.

```
def multiply(x, y):  
    return x * y  
  
result = multiply(4, 3)  
print(result) # Output: 12
```

Error Handling

```
try:
    # Risky Code (Jisme error aa sakta hai)
except Exception as e:
    # Error handle karna
```

Agar tu **chahta hai ki koi bhi error ho ya na ho, ek specific part hamesha chale, toh finally use hota hai.**

```
try:
    print("Trying...")
    x = 10 / 2
except Exception as e:
    print(f"Error: {e}")
finally:
    print("This will always run!")
```

open and reading

with open() Statement in Python

Python me `with open()` statement ek **best practice** hai **file handling** ke liye.

```
with open("filename.txt", "mode") as file:  
    # File operations
```

Auto-close karta hai file ko, chahe program crash ho ya nahi.

`close()` **manually** likhne ki zaroorat nahi hoti.

Safe & clean code hota hai.

write file

```
with open("hacker.txt", "w") as f:  
    f.write("Hacking is fun!\n")
```

Read file

```
with open("hacker.txt", "r") as f:  
    content = f.read()  
    print(content)
```

Append file

```
with open("logs.txt", "a") as f:  
    f.write("New login attempt detected!\n")
```

Basic port scan in threading

```
import socket
import threading

def scan_port(ip, port):
    s = socket.socket(socket.AF_INET, socket.SOCK_STREAM)
    s.settimeout(1)
    result = s.connect_ex((ip, port))
    s.close()

    if result == 0:
        print(f"port {port} is open")
    else:
        print(f"port {port} is closed")

Domain = input("enter target ip or domain:")
target_ip = socket.gethostbyname(Domain)
print(f"target ip is {target_ip}")

def threaded_scan():
    threads = []
    for port in range(21, 1024):
        t = threading.Thread(target=scan_port, args=(target_ip, port))
        t.start()
        threads.append(t)

    for t in threads:
        t.join()

    print("port scan complete")

threaded_scan()
```

we could do this too----->


```

port_list = [21,22,80,443,8080]

def threaded_scan():

    threads = []

    for port in port_list:

        t = threading.Thread(target=scan_port, args=(target_ip, port))

        t.start()

        threads.append(t)

    for t in threads:

        t.join()

    print("port scan complete")

threaded_scan()

port_list = [21,22,80,443,8080]

```

Threading Ka Kya Faayda Hua?

Ports ek saath scan ho rahe hain (Parallel Execution)

Speed fast ho gayi, pehle ek ek port scan hota tha ab saare parallel scan ho rahe hain

Bruteforce, Enumeration, Scanning ke liye useful hai

Word by Word Breakdown

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

`target=scan_port` → `scan_port()` function ko execute karega

`args=(target_ip, port)` → `scan_port()` function ko IP aur port pass kar raha hai

`t.start()`

Thread start hota hai jo `scan_port(ip, port)` execute karta hai

```
threads.append(t)
```

Thread list me add ho raha hai taaki baad me `join()` kar sakein

```
t.join()
```

Saare threads complete hone tak wait karta hai

Breakdown of `s.connect_ex((ip, port))` and `(target_ip, port)`

Tumhara sawal:

Kya `s.connect_ex((ip, port))` jaruri hai?

`target_ip` aur `port` kahaan se aa rahe hain?

Chalo ek ek karke clear karte hain.

`s.connect_ex((ip, port))` Ka Matlab Kya Hai?

`s.connect_ex((ip, port))` ek function hai jo check karta hai ki port open hai ya close.

Ye ek alternative hai `s.connect()` ka jo error throw nahi karta (Isliye prefer karte hain).

Ye `0` return karega agar connection successful ho (port open ho).

Ye `1` ya koi aur error code return karega agar connection fail ho (port closed ho).

complete portscan using threading

```
import socket
import threading

def port_scan(ip, port, file_name):
    try:
        s = socket.socket(socket.AF_INET, socket.SOCK_STREAM)
        s.settimeout(1)
        result = s.connect_ex((ip, port))
        s.close()

        if result == 0:
            output = f"port {port} is open\n"
            with open(file_name, "a") as f:
                f.write(output)
            print(output)

    except Exception as e:
        print(f"[ ] Error scanning port {port}: {e}")

target = input("    Enter Target (IP/Domain): ").strip()
port_choice = input("    Enter Ports (defaults=1-1024) or comma-separated list: ").strip()
file_name = input("    Enter file name to save results: ").strip()

try:
    target_ip = socket.gethostbyname(target)
    print(f"    Target is {target_ip} ({target})")
except Exception as e:
    print(f"    Error: Unable to resolve target: {target}.")
    exit()

if port_choice:
    try:
        port_list = [int(p) for p in port_choice.split(",") if p.isdigit()]
    except Exception as e:
        print(f"    Invalid port input. Using default (1-1024).")
```

```

        port_list = range(1, 1024)
else:
    port_list = range(1, 1024)

with open(file_name, "w") as f:
    f.write(f"    Scan Results for {target} ({target_ip})\n")
    f.write("=" * 50 + "\n")

def threaded_scan():
    threads = []
    for port in port_list:
        t = threading.Thread(target=port_scan, args=(target_ip, port,
file_name))
        t.daemon = True
        t.start()
        threads.append(t)

    for t in threads:
        t.join()

    print(f"    Port scan complete. Results saved to {file_name}")

threaded_scan()

```

strip() ka use?

```
target = input("    Enter Target (IP/Domain): ").strip()
```

Problem: Jab user input deta hai, toh kabhi kabhi extra spaces aa jati hain:

scanme.nmap.org

Isme shuru ya end me spaces rahengi, jo error de sakti hain.

.strip() un extra spaces ko remove karta hai.

" hello ".strip() → "hello"

" scanme.nmap.org ".strip() → "scanme.nmap.org"

Jaruri nahi hai, but best practice hai.

Ye line ka full explanation (Port Parsing)

```
if port_choice:
```

```
try:
```

```
port_list = [int(p) for p in port_choice.split(",") if p.isdigit()]
```

python

CopyEdit

```
except ValueError: print("    Invalid port input. Using default (1-65535).")
port_list = range(1, 65536)
```

Ye kya kar raha hai?

User agar comma-separated ports de (ex: "22,80,443"), toh usko list me convert karna hai.

Step-by-Step Breakdown:

`port_choice.split(",")` → String ko comma , pe todta hai

`"22,80,443".split(",")` → ["22", "80", "443"]

List comprehension se saare items int me convert kar raha hai:

```
[int(p) for p in port_choice.split(",") if p.isdigit()]
```

"22" → 22, "80" → 80, "443" → 443

Agar koi non-numeric value ho toh ignore karega (like "abc" hata dega)

Agar error aaye (jaise "22,abc,443" me "abc" issue kare), toh full range use karega:

```
except ValueError:
print("    Invalid port input. Using default (1-65535).")
port_list = range(1, 65536)
```

Invalid input ka check + default value

Jaruri hai?

Haan, kyunki ye user input ko properly handle karta hai, nahi toh program crash ho sakta hai.

Ye `f.write("=" * 40 + "\n")` kya kar raha hai?

```
f.write("=" * 40 + "\n")
```

Ek line me = ka 40 baar repetition bana raha hai.

`"=" 40` → "=====

Purpose? File me output clean aur readable dikh sake.

File ka output aisa dikhega:

Scan Results for scanme.nmap.org (45.33.32.156)

- [] Port 22 is OPEN
- [] Port 80 is OPEN
- [] Port 443 is OPEN

Jaruri hai?

Nahi, sirf formatting ke liye hai.

Ye `t.daemon = True` kya hai?

`t.daemon = True`

Daemon threads background me kaam karte hain, bina program ko block kiye.

Example:

Tum main thread me ho, aur tumhara port scanner multiple threads bana raha hai.

Agar main program exit ho jaye, toh normal threads chalte rahenge aur program hang ho sakta hai.

Daemon threads automatically terminate ho jaate hain agar main program exit ho jaye.

output variable banana zaroori tha kya?

Code:

```
output = f"port {port} is open\n"
with open("file", "a") as f:
    f.write(output)
print(output)
```

Doubt: Kya directly `f.write()` nahi kar sakte? output variable zaroori hai kya?

Answer:

Direct likh sakte ho, output banana compulsory nahi hai. Yeh sirf readability improve karne ke liye kiya gaya hai.

"a" mode kyun use kiya, "w" kyun nahi?

"a" (append mode) isliye use kiya gaya hai taaki naye results purani file ke content ko delete na karein.

Agar "w" use karte toh purana data delete ho jata.

Example:

```
with open("file", "w") as f: # Yeh har baar purana content delete karega
    f.write(f"port {port} is open\n")
```

"w" sirf tab use karo jab purani file overwrite karni ho.

"a" use karo jab naya data add karna ho bina purana delete kiye.

mtlb aesa h jese phle hum ip lete phir port bgra phir hum file name lete h ek file create krte h w se or header likhte h phir portscan run hota h and file mai save hota hai.

Toh use kese pta konsa file mai save krna hai? Iska connection jo phle file bnayi usse kese?

Kaise Pata Chalta Hai Ki Konsi File Me Save Karna Hai?

Simple logic hai bhai!

Jab file create hoti hai (w mode) tab user se file ka naam input le chuke hote hain.

Phir har port scan hone par wahi file_name function me pass kar dete hain.

Final Flow (Ekdam Simple Tarika)

User se file_name le lo.

w mode se ek baar file create karke header likh do.

port_scan() function ko wahi file_name pass karo.

a mode se baar-baar open ports append karte raho.

Pehle file create hui, phir scan hone ke baad usi me results save hote gaye.

Demo Process

Linux Version

```
import subprocess

# Command ko run karna (ls -l)

subprocess.run(["ls", "-l"])

# Background me ping command chalana
p = subprocess.Popen("ping 8.8.8.8", shell=True)
print("Ping running in background...")

# Process list ko print karna
out = subprocess.getoutput("ps -aux")
print(out)
```

windows version ----->>>>>>>>>

```
import subprocess

# Command ko run karna (dir)
subprocess.run(["cmd", "/c", "dir"])

# Background me ping command chalana
p = subprocess.Popen("ping 8.8.8.8 -t", shell=True)
print("Ping running in background...")

# Process list ko print karna (tasklist)
out = subprocess.getoutput("tasklist")
print(out)
```


find and kill proc

linux version ----->>>

```
import subprocess
import os

# "ping" process ka PID nikalna
out = subprocess.getoutput("pgrep ping")
pids = out.split()

# Har process ko forcefully kill karna
for pid in pids:
    os.system(f"kill -9 {pid}")

print("Ping process killed successfully!")
```

Windows version ----->>

```
import subprocess
import os

# Process ka naam ke basis pe PID find karna
output = subprocess.check_output('wmic process where "name=\'notepad.exe\'" get ProcessId', shell=True)
pids = [pid for pid in output.decode().split() if pid.isdigit()]

# Har PID ko forcefully kill karna
for pid in pids:
    os.system(f"taskkill /PID {pid} /F")

print("Notepad process killed successfully!")
```

get,auth and post

get ----->>>>

```
import requests

# User se URL input lena
url = input("Enter a URL: ").strip()

# Request bhejna
response = requests.get(url)

# Response details print karna
print(f"Status Code: {response.status_code}")
print(f"Headers:\n{response.headers}")
print(f"Content:\n{response.text}")
```

post ----->>>>>>

```
import requests

# User se URL input lena
url = input("Enter a URL (e.g., https://site.com): ").strip()

# Credentials ya form data define karna
cred = {
    "username": "test",
    "password": "test"
}

# POST request bhejna
response = requests.post(url, data=cred)
```

```
# Response details print karna
print(f"Status Code: {response.status_code}")
print(f"Headers:\n{response.headers}")
print(f"Content:\n{response.text}")
```

[illegible]

after login, if we want to access the page,like dashboard profile etc
then we need to send the cookies to the server

ust because we are sending the cookies, we dont need to send the credentials
post request is not required

```
import requests

# User se URL input lena
url = input("Enter the URL: ").strip()

# Headers define karna
headers = {
    "Content-Type": "application/json",
    "User-Agent": "Mozilla/5.0"
}

# Cookies define karna
cookies = {
    "sessionid": "1234",
    "csrftoken": "abcd"
}

# GET request bhejna (headers & cookies ke sath)
response = requests.get(url, headers=headers, cookies=cookies)

# Response details print karna
print(f"Status Code: {response.status_code}")
```

```
print(f"Content:\n{response.content}")  
print(f"Text:\n{response.text}")
```

port scanner

[complete portscan using threading](#) <<<<<<<<<<----- links

```
import socket
import threading
import sys
import readline #for using arrow key for reading lines

def port_scan(ip, port, file_name):

    try:
        s = socket.socket(socket.AF_INET, socket.SOCK_STREAM)
        s.settimeout(1)
        result = s.connect_ex((ip, port))
        s.close()

        if result == 0:
            output= f"port {port} is open\n"
            with open(file_name, "a") as f:
                f.write(output)
            print(output)

    except Exception as e:
        print(f"[ ] Error scanning port {port}: {e}")

try:
    target = input("    Enter Target (IP/Domain): ").strip()
    port_choice = input("    Enter Ports (defaults=1-1024) or coma-separated list:").strip()
    file_name = input("    Enter file Name to save results :").strip()

    try:
        target_ip = socket.gethostbyname(target)
        print(f"    target is {target_ip} ({target})")
```

```

except Exception as e:
    print(f"    Error: Unable to resolve target: {target}.")
    exit()

if port_choice:
    try:
        port_list = [int(p) for p in port_choice.split(",") if
p.isdigit()]
        except Exception as e:
            print(f"    Invalid port input. Using default (1-1024).")
            port_list = range(1, 1024)

    else:
        port_list = range(1, 1024)

with open(file_name, "w") as f:
    f.write(f"    Scan Results for {target} ({target_ip})\n")
    f.write("=" * 50 + "\n")

def threaded_scan():
    threads = []
    for port in port_list:
        t = threading.Thread(target=port_scan, args=(target_ip,
port,file_name))
        t.daemon = True
        t.start()
        threads.append(t)

    for t in threads:
        t.join()

    print(f"    Port scan complete. Results saved to {file_name}")


threaded_scan()

```

```
except KeyboardInterrupt:
    print("\n[ ] Scan Interrupted! Exiting... ")
    print(" Happy Hacking! ")
    sys.exit(0)
```

```
except Exception as e:
    print(f" unexpected error : {e}")
```

direnum

for the code this is link ---->  [directory scanner](#)

```
import requests
import queue
import sys
import threading
import readline

domain = input("enter the domain name:").strip()
wordlist_file =input("enter the the wordlist name: ").strip()
output_file = input("enter where you want to save").strip()
with open(output_file, "w") as file:
    file.write(f"    Result for the domain :{domain}\n")
    file.write("==" * 50 + "\n")

q =queue.Queue()

def check_directories():
    while not q.empty():
        dir = q.get()
        url = f"https://{domain}/{dir}"

        if not dir or dir.startswith("#"):
            q.task_done()
            continue

        try:
            response = requests.get(url,timeout=5)
            if response.status_code < 400:
                outs= f"    == {dir} (status: {response.status_code}"
                print(outs)
                with open(output_file, "a") as file:
                    file.write(outs + "\n")
```



```

except requests.RequestException:
    pass

q.task_done()

try:
    def main():
        try:
            with open(wordlist_file, "r") as file:
                directories = [line.strip() for line in file.readlines()if
line.strip() and not line.startswith("#")]
                #[line.strip() for line in f.readlines()if
line.strip() and not line.startswith("#")]
                #[line.strip() for line in f.readlines()if
line.strip() and not line.startswith("#")]
            except Exception as e:
                print(f"please provide correct wordlist: {e}")
                return

        for dir in directories:
            q.put(dir)

    threads = []

    try:
        num_threads = int(input("enter speed:").strip())
        if num_threads<=0:
            raise ValueError("please provide between 1-10")

    except ValueError:
        print("enter between 1-10")
        return

    for s in range(num_threads):
        s = threading.Thread(target=check_directories)
        threads.append(s)

```

```
s.start()

for s in threads:
    s.join()


print("  directory scan complete  ")

if __name__ == "__main__":
    main()

except KeyboardInterrupt:
    print("\n[  ] Scan Interrupted! Exiting...  ")
    print("  Happy Hacking!  ")
    sys.exit(0)

except Exception as e:
    print(f" unexpected error : {e}")
```

subdomain enum

 [subdomain](#) <----- link

```
import threading
import requests
import queue
import sys
import readline

q = queue.Queue()

output_file =input("Enter file name to save results").strip()

domain = input("Enter domain name:").strip()
wordlist = input("Enter wordlist file:").strip()

def check_subdomain(domain):
    while not q.empty():
        sub =q.get()
        url = f"https://{sub}.{domain}"

        try:
            response =requests.get(url, timeout=3)
            if response.status_code < 400:
                done = (f"discovered domain : {url}")
                print(done)
                with open(output_file, "a") as f:
                    f.write(done +"\n")

        except requests.RequestException:
            pass

    q.task_done()
```

```
###print(f"we have these subdomains for you thanks for using sub_enum")

try:
    def main():
        try:
            with open(wordlist, "r") as f:
                subdomains =[line.strip() for line in f.readlines()]

        except Exception as e:
            print(f"error is {e}")
            print(f"please provide good wordlist line by lines thanks")
            return

    for sub in subdomains:
        q.put(sub)

    threads = []
    num_threads = int(input("enter speed:"))

    for s in range(num_threads):
        s = threading.Thread(target=check_subdomain,args=(domain,))
        threads.append(s)
        s.start()

    for s in threads:
        s.join()

    print("scan complete")
```

```
if __name__ == "__main__":  
    main()
```

```
except KeyboardInterrupt:  
    print("\n[ ] Scan Interrupted! Exiting... ")  
    print(" Happy Hacking! ")  
    sys.exit(0)
```

main.py

```
import subprocess
import os

def main():
    print("""
===== GHOST RECON FRAMEWORK =====
1. Subdomain Enumeration
2. Directory Enumeration
3. Port Scanning
=====
""")

    try:
        choice = input("Enter your choice (1-3): ").strip()

        # Set the working directory where 'modules' folder exists
        os.chdir('/home/ghost/Desktop/python4hackers/ghostrecon')

        if choice == "1":
            subprocess.run(["python", os.path.join("modules",
"subdomain_enum.py")])

        elif choice == "2":
            subprocess.run(["python", os.path.join("modules", "dir_enum.py")])

        elif choice == "3":
            subprocess.run(["python", os.path.join("modules",
"port_scan.py")])

        else:
            print("    Invalid Choice! Please enter 1, 2, or 3.")

    except KeyboardInterrupt:
        print("\n\n    Oops! Seems like you're in a hurry... Happy Hacking,
Ghost!        ")
```

```
if __name__ == "__main__":  
    main()
```