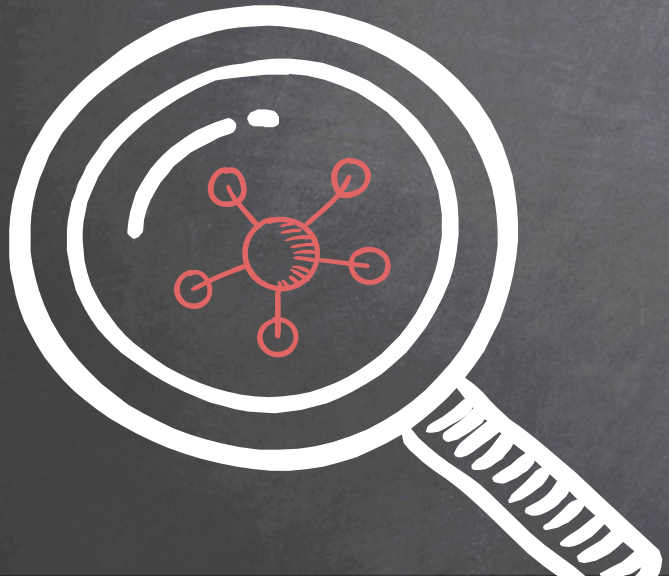
Network_Scanner

- Discover all devices on the network.
- Display their IP address.
- Display their MAC address.



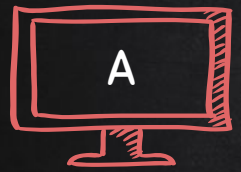
Network_Scanner

Discovering Clients On The Same Network

- The subprocess module contains a number of functions.
- These functions allow us to execute system commands.
- Commands depend on the OS which executes the script.

Syntax:

```
import subprocess
subprocess.call("COMMAND", Shell=True)
```



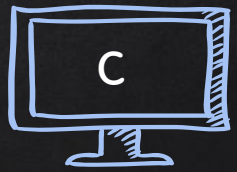
Router

IP: 10.0.2.1

MAC: 00:11:22:33:44:20

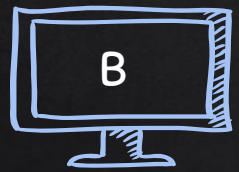
IP: 10.0.2.7

MAC: 00:11:22:33:44:55



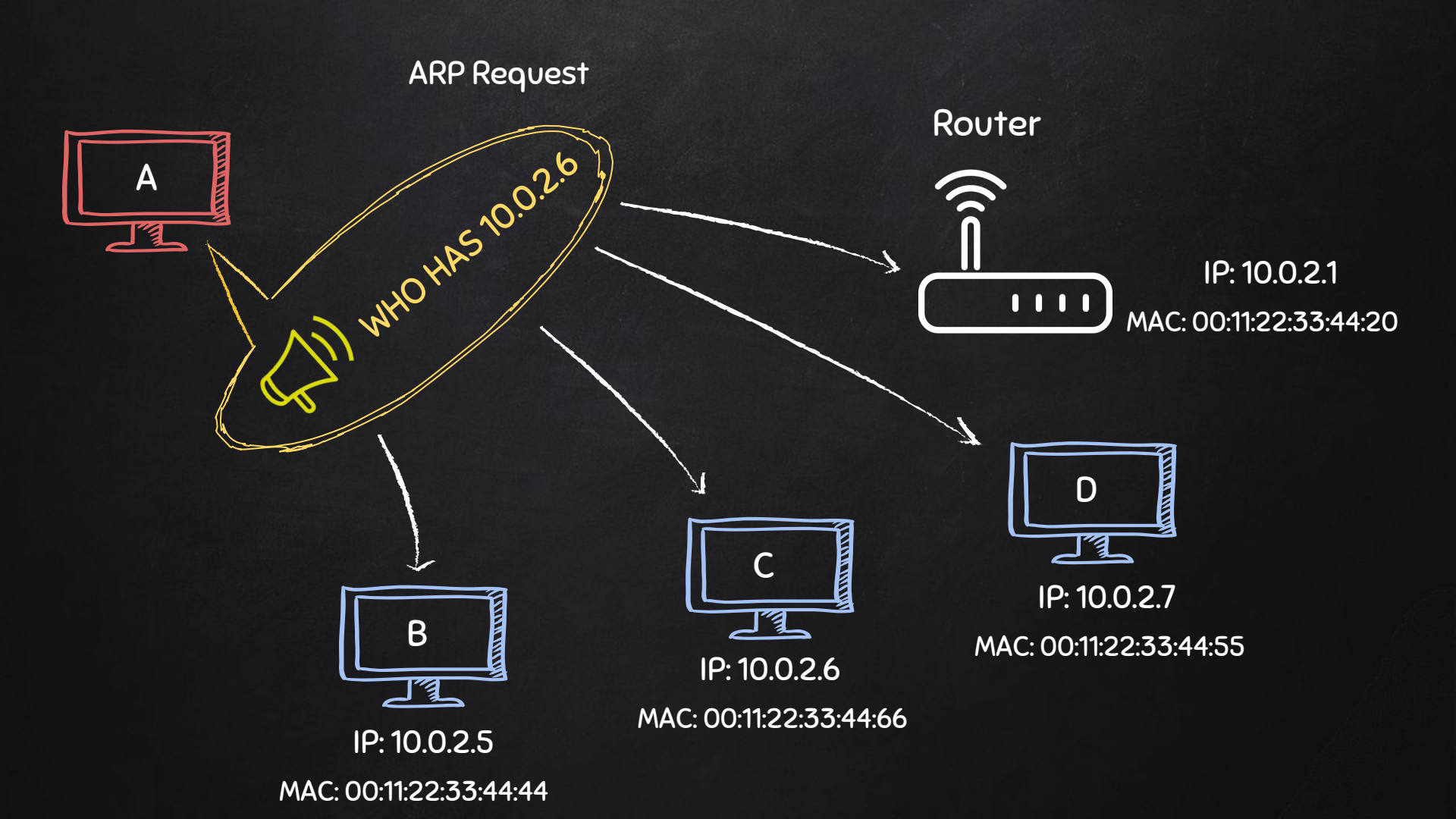
IP: 10.0.2.6

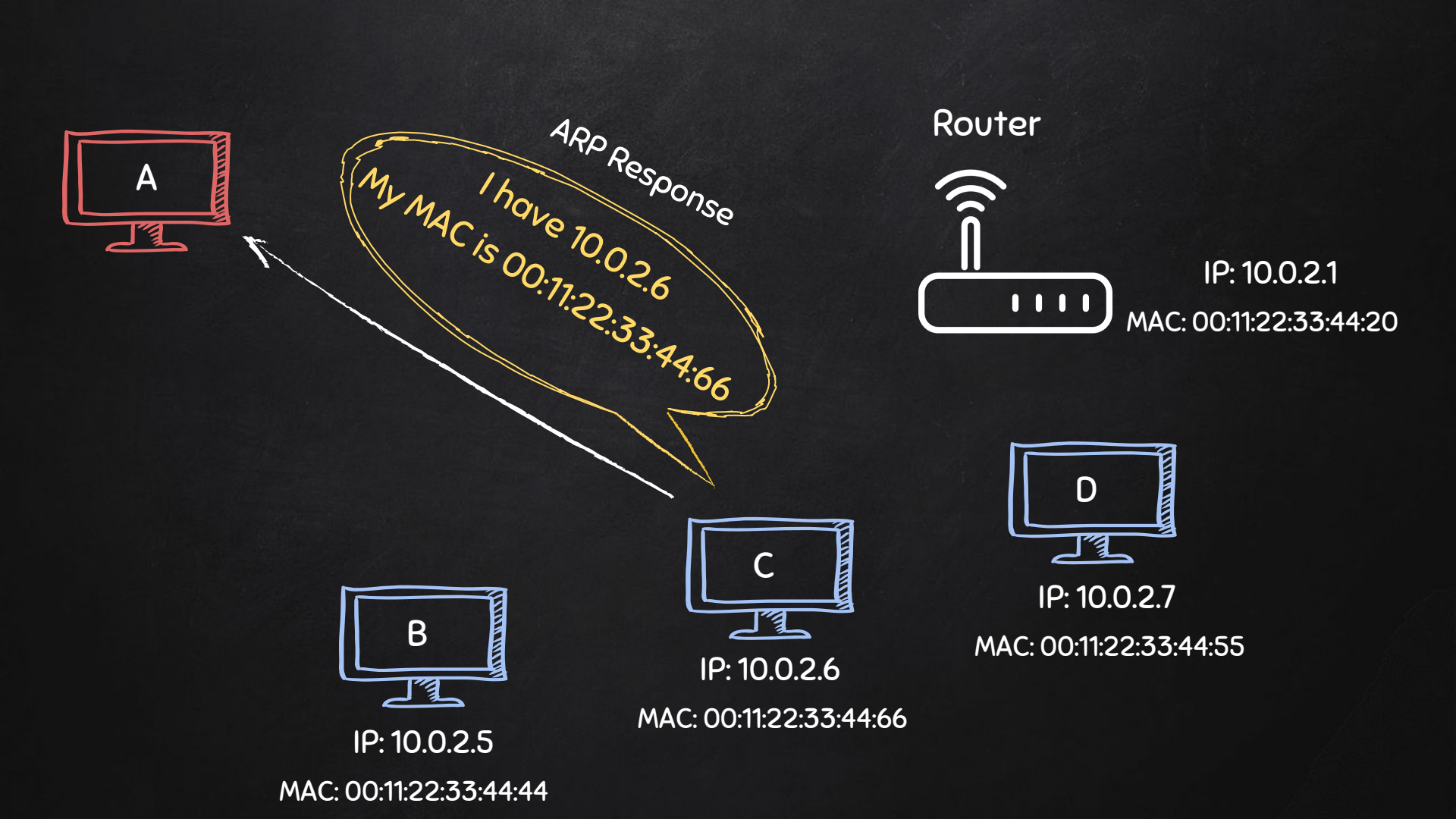
MAC: 00:11:22:33:44:66



IP: 10.0.2.5

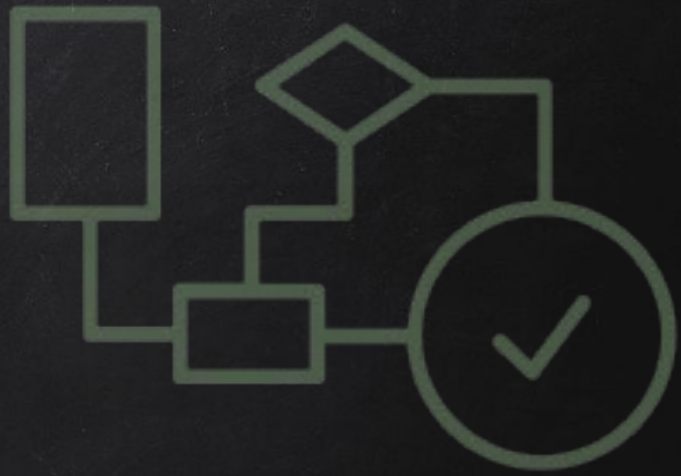
MAC: 00:11:22:33:44:44


ARP Request
A
WHO HAS 10.0.2.6
Router
IP: 10.0.2.1
MAC: 00:11:22:33:44:20
B
IP: 10.0.2.5
MAC: 00:11:22:33:44:44
C
IP: 10.0.2.6
MAC: 00:11:22:33:44:66
D
IP: 10.0.2.7
MAC: 00:11:22:33:44:55


A
ARP Response
I have 10.0.2.6
My MAC is 00:11:22:33:44:66
Router
IP: 10.0.2.1
MAC: 00:11:22:33:44:20
D
IP: 10.0.2.7
MAC: 00:11:22:33:44:55
C
IP: 10.0.2.6
MAC: 00:11:22:33:44:66
B
IP: 10.0.2.5
MAC: 00:11:22:33:44:44

Network Scanner

Algorithm



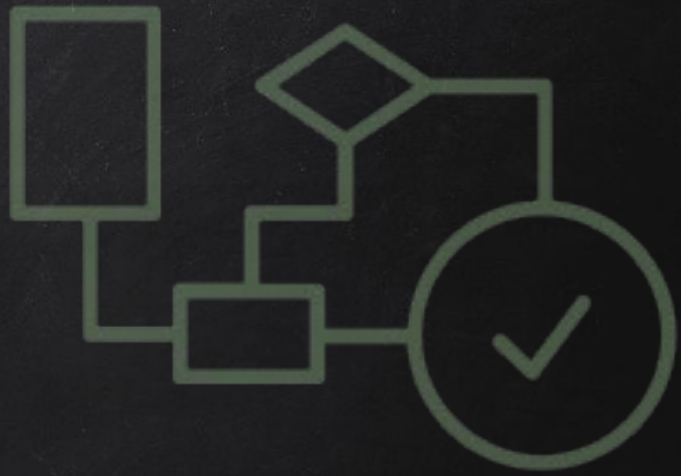
Goal → Discover clients on network.

Setps:

1. Create arp request directed to broadcast MAC asking for IP.
2. Send packet and receive response.
3. Parse the response.
4. Print result.

Network Scanner

Algorithm



Goal → Discover clients on network.

Setps:

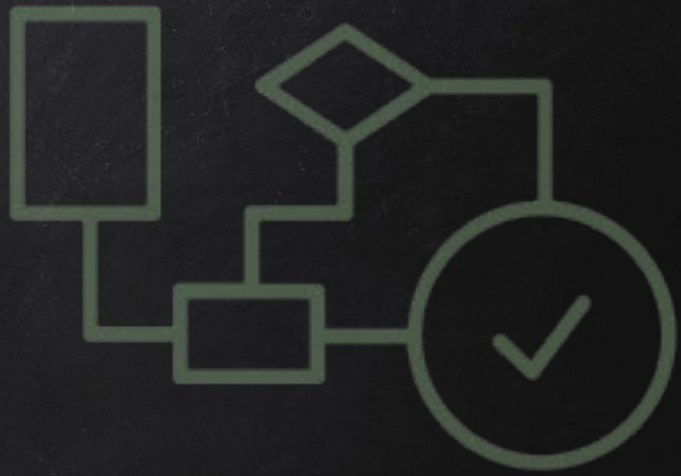
1. Create arp request directed to broadcast MAC asking for IP.

   Two main parts:

   - ➔ Use ARP to ask who has target IP.
   - ➔ Set destination MAC to broadcast MAC.

Network Scanner

Algorithm



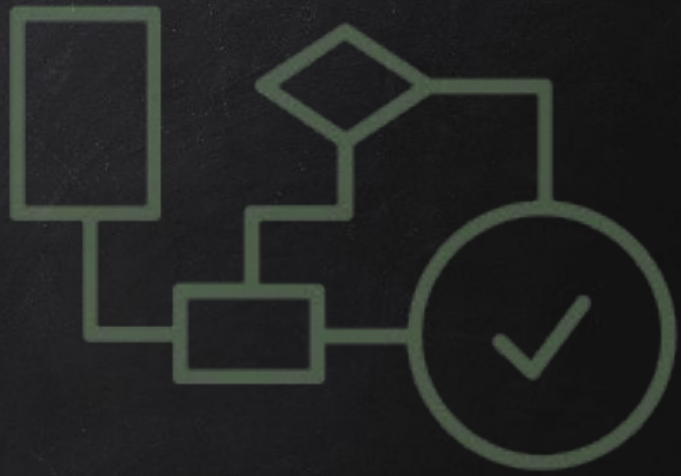
Goal → Discover clients on network.

Setps:

1. Create arp request directed to broadcast MAC asking for IP.
2. Send packet and receive response.
3. Parse the response.
4. Print result.

Network Scanner

Algorithm



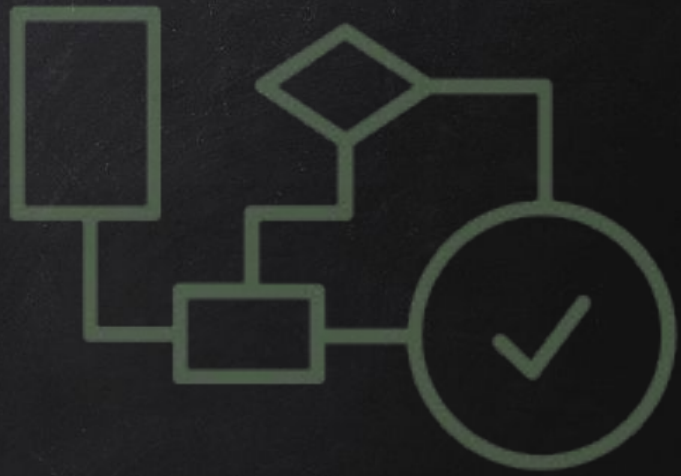
Goal → Discover clients on network.

Setps:

1. Create arp request directed to broadcast MAC asking for IP.
2. Send packet and receive response.
3. Parse the response.
4. Print result.

Network Scanner

Algorithm



Goal → Discover clients on network.

Setps:

1. Create arp request directed to broadcast MAC asking for IP.
2. Send packet and receive response.
3. Parse the response.
4. Print result.

Lists

- List of values/elements, all can be stored in one variable.

Ex:

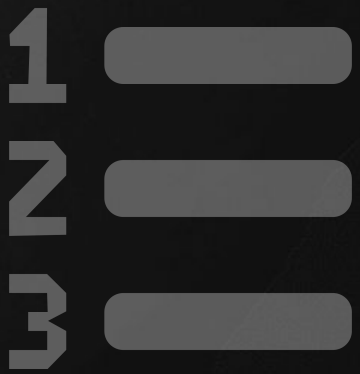
```
lucky_numbers_list = [3, 7, 8, 17, 24]
```

Python will interpret this as

index	0	1	2	3	4
value	3	7	8	17	24

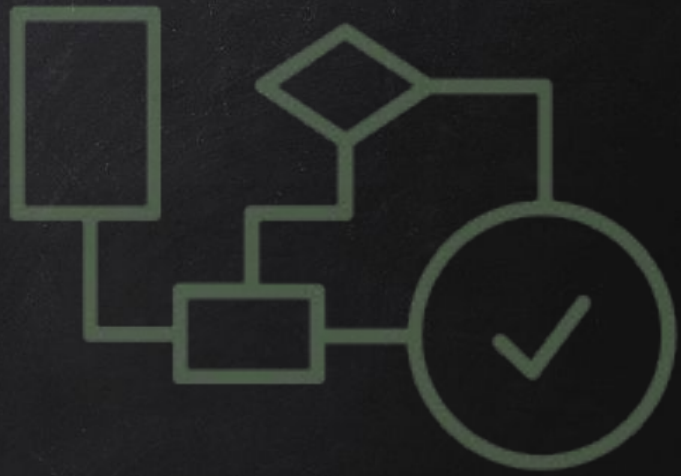
Elements can be accessed using their index

```
print(lucky_numbers_list[0])  #prints 3
print(lucky_numbers_list[1])  #prints 7
print(lucky_numbers_list[2])  #prints 8
```



Network Scanner

Algorithm



Goal → Discover clients on network.

Setps:

1. Create arp request directed to broadcast MAC asking for IP.
2. Send packet and receive response.
3. Parse the response.
4. Print result.

Dictionaries

- Similar to lists but use key instead of index.

Ex:

```
target_client = {"mac": "00:11:22:33:44:55", "ip": "10.0.2.1", "os": "windows"}
```

Python will interpret this as

Key	mac	ip	os
value	00:11:22:33:44:55	10.0.2.1	windows

Elements can be accessed using their key

```
print(target_client["mac"])  #prints 00:11:22:33:44:55
print(target_client["ip"])   #10.0.2.1
print(target_client["os"])   #windows
```

List of Dictionaries

index	0			1			2		
value	Key	mac	ip	Key	mac	ip	Key	mac	ip
	value	00:11:11:11:11:11	10.0.2.1	value	00:22:22:22:22	10.0.2.2	value	00:33:33:33:33	10.0.2.7

index	0	1	2
value	{“ip”: “10.0.2.1”, “mac”: “00:11:22:33:44:55”}	{“ip”: “10.0.2.2”, “mac”: “00:24:A2:31:11:22”}	{“ip”: “10.0.2.4”, “mac”: “00:11:A2:44:44:23”}

Elements can be accessed using list index and dict key

```
print(list[0][“mac”])  #prints 00:11:11:11:11:11
print(list[1][“mac”])  #prints 00:22:22:22:22
print(list[2][“ip”])   #prints 10.0.2.7
```