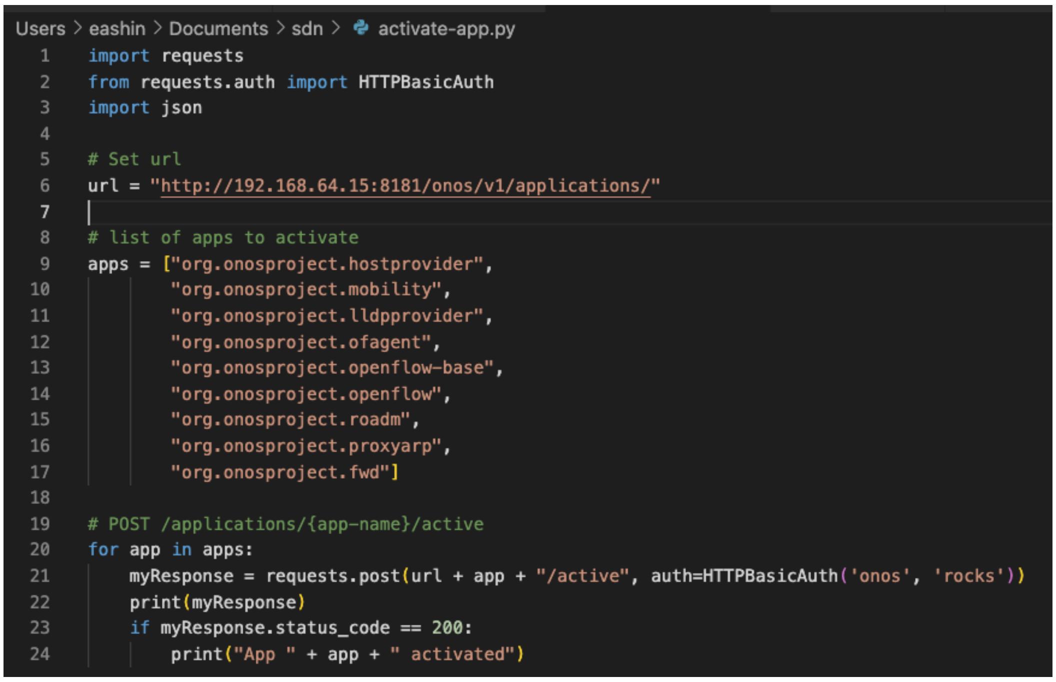
Demo 5: Using ONOS RESTful API interface to manage hosts, devices, applications and settings

1.3. Tasks

1.3.1. Task 1

- *Start the required ONOS applications using a python-based approach.*

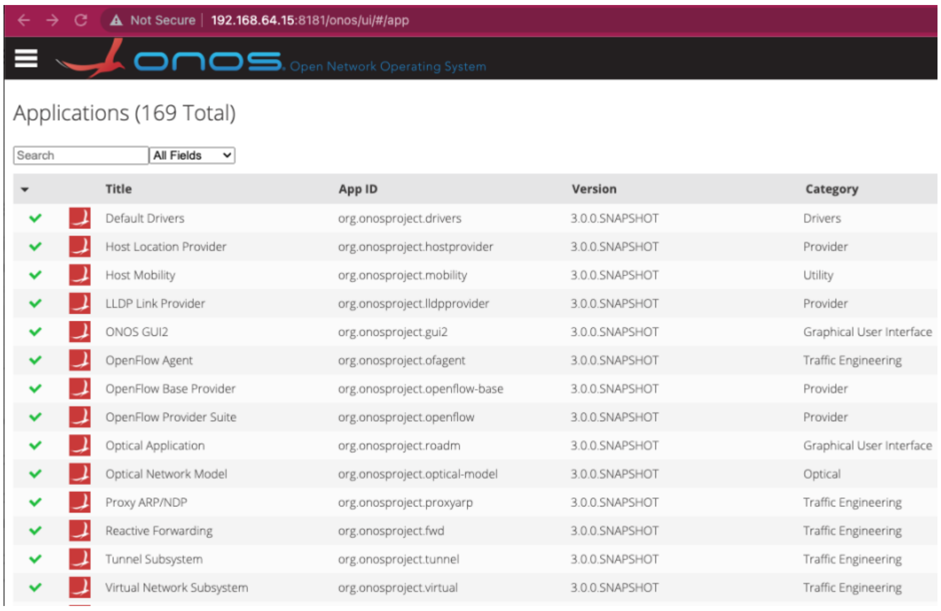
Using the below python script, I have activated required applications of ONOS:



Users > eashin > Documents > sdn > activate-app.py

```python
import requests
from requests.auth import HTTPBasicAuth
import json

# Set url
url = "http://192.168.64.15:8181/onos/v1/applications/"

# list of apps to activate
apps = ["org.onosproject.hostprovider",
        "org.onosproject.mobility",
        "org.onosproject.lldpprovider",
        "org.onosproject.ofagent",
        "org.onosproject.openflow-base",
        "org.onosproject.openflow",
        "org.onosproject.roadm",
        "org.onosproject.proxyarp",
        "org.onosproject.fwd"]

# POST /applications/{app-name}/active
for app in apps:
    myResponse = requests.post(url + app + "/active", auth=HTTPBasicAuth('onos', 'rocks'))
    print(myResponse)
    if myResponse.status_code == 200:
        print("App " + app + " activated")
```

Below it is checked from ONOS GUI:

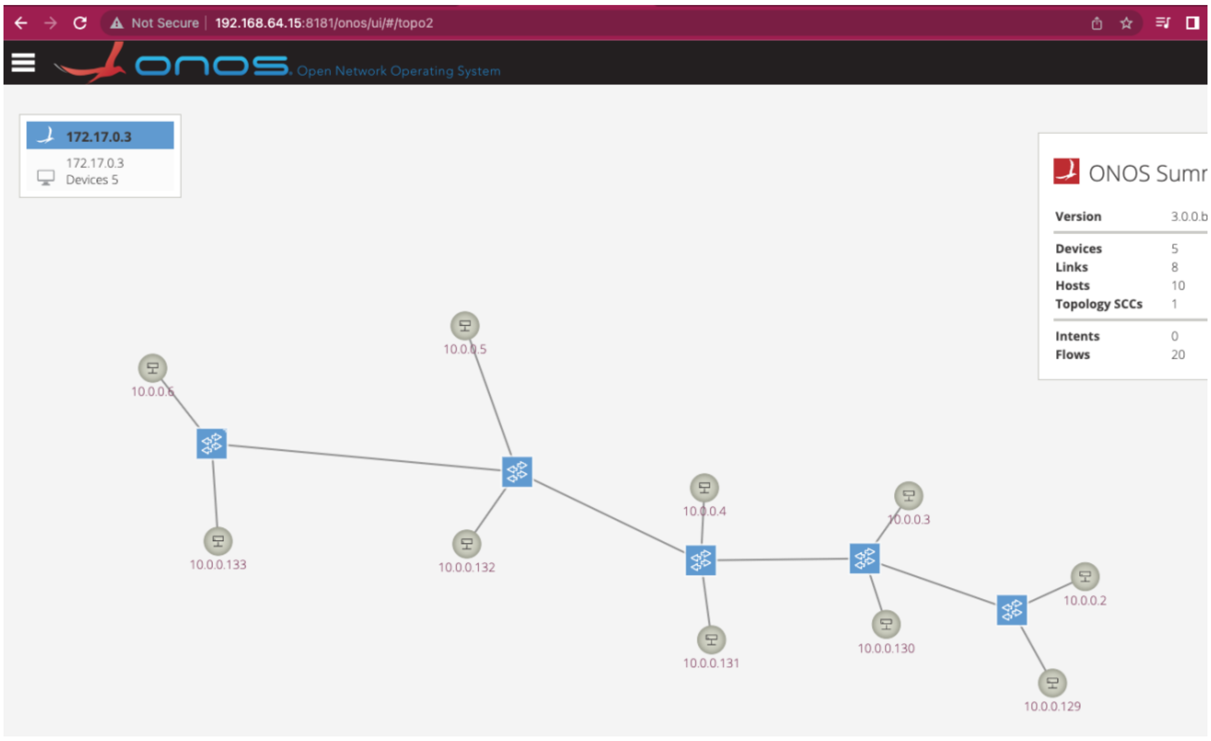

Not Secure
192.168.64.15:8181/onos/ui/#/app
onos. Open Network Operating System
Search
All Fields

Applications (169 Total)

		Title	App ID	Version	Category
✓		Default Drivers	org.onosproject.drivers	3.0.0.SNAPSHOT	Drivers
✓		Host Location Provider	org.onosproject.hostprovider	3.0.0.SNAPSHOT	Provider
✓		Host Mobility	org.onosproject.mobility	3.0.0.SNAPSHOT	Utility
✓		LLDP Link Provider	org.onosproject.lldpprovider	3.0.0.SNAPSHOT	Provider
✓		ONOS GUI2	org.onosproject.gui2	3.0.0.SNAPSHOT	Graphical User Interface
✓		OpenFlow Agent	org.onosproject.ofagent	3.0.0.SNAPSHOT	Traffic Engineering
✓		OpenFlow Base Provider	org.onosproject.openflow-base	3.0.0.SNAPSHOT	Provider
✓		OpenFlow Provider Suite	org.onosproject.openflow	3.0.0.SNAPSHOT	Provider
✓		Optical Application	org.onosproject.roadm	3.0.0.SNAPSHOT	Graphical User Interface
✓		Optical Network Model	org.onosproject.optical-model	3.0.0.SNAPSHOT	Optical
✓		Proxy ARP/NDP	org.onosproject.proxyarp	3.0.0.SNAPSHOT	Traffic Engineering
✓		Reactive Forwarding	org.onosproject.fwd	3.0.0.SNAPSHOT	Traffic Engineering
✓		Tunnel Subsystem	org.onosproject.tunnel	3.0.0.SNAPSHOT	Traffic Engineering
✓		Virtual Network Subsystem	org.onosproject.virtual	3.0.0.SNAPSHOT	Traffic Engineering

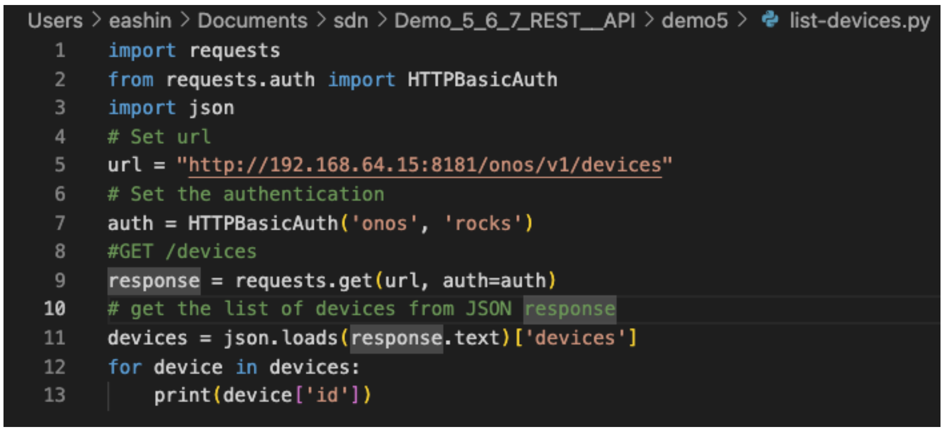
1.3.1. Task 2

A pre-defined topology created using provided script file “demo5.sh”:



- [*Create python program to list all available devices by their IDs.*]()

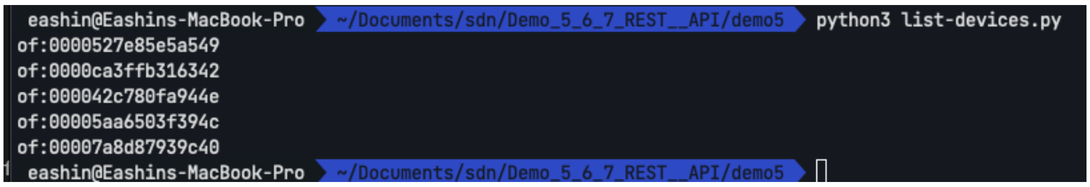
Here is the python script which can list all available devices IDs.



Users > eashin > Documents > sdn > Demo_5_6_7_REST__API > demo5 > list-devices.py

```python
import requests
from requests.auth import HTTPBasicAuth
import json
# Set url
url = "http://192.168.64.15:8181/onos/v1/devices"
# Set the authentication
auth = HTTPBasicAuth('onos', 'rocks')
#GET /devices
response = requests.get(url, auth=auth)
# get the list of devices from JSON response
devices = json.loads(response.text)['devices']
for device in devices:
    print(device['id'])
```

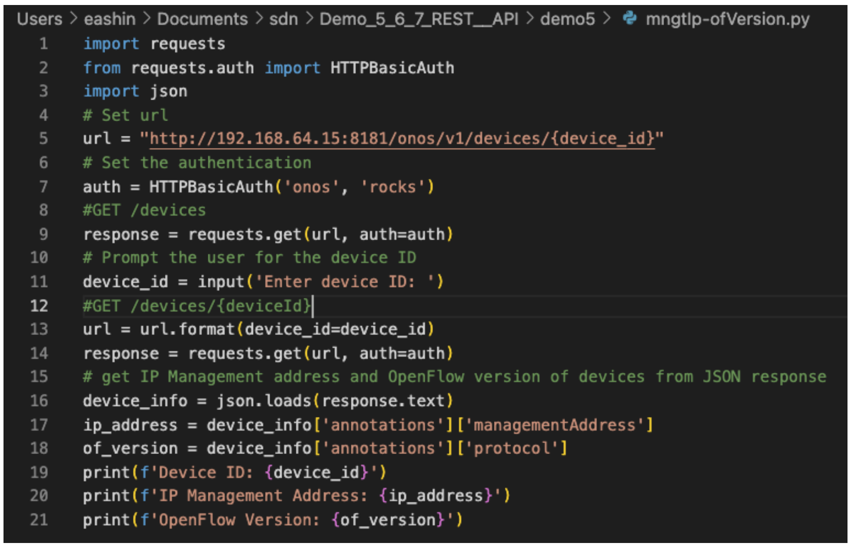
Below is the output of the program with all available device ids of the topology.



```
 eashin@Eashins-MacBook-Pro  ~/Documents/sdn/Demo_5_6_7_REST__API/demo5  python3 list-devices.py
of:0000527e85e5a549
of:0000ca3ffb316342
of:000042c780fa944e
of:00005aa6503f394c
of:00007a8d87939c40
 eashin@Eashins-MacBook-Pro  ~/Documents/sdn/Demo_5_6_7_REST__API/demo5 
```

- *Create a python program to get the IP management address and the OpenFlow version used by a given device in the pre-defined architecture.*

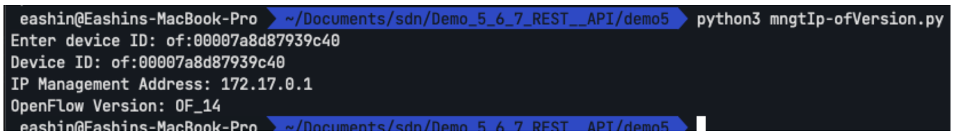
  Here is the python script which can GET the IP management address and the OpenFlow version by a given device ID.



Users > eashin > Documents > sdn > Demo_5_6_7_REST__API > demo5 > mngtIp-ofVersion.py

```python
import requests
from requests.auth import HTTPBasicAuth
import json
# Set url
url = "http://192.168.64.15:8181/onos/v1/devices/{device_id}"
# Set the authentication
auth = HTTPBasicAuth('onos', 'rocks')
#GET /devices
response = requests.get(url, auth=auth)
# Prompt the user for the device ID
device_id = input('Enter device ID: ')
#GET /devices/{deviceId}
url = url.format(device_id=device_id)
response = requests.get(url, auth=auth)
# get IP Management address and OpenFlow version of devices from JSON response
device_info = json.loads(response.text)
ip_address = device_info['annotations']['managementAddress']
of_version = device_info['annotations']['protocol']
print(f'Device ID: {device_id}')
print(f'IP Management Address: {ip_address}')
print(f'OpenFlow Version: {of_version}')
```

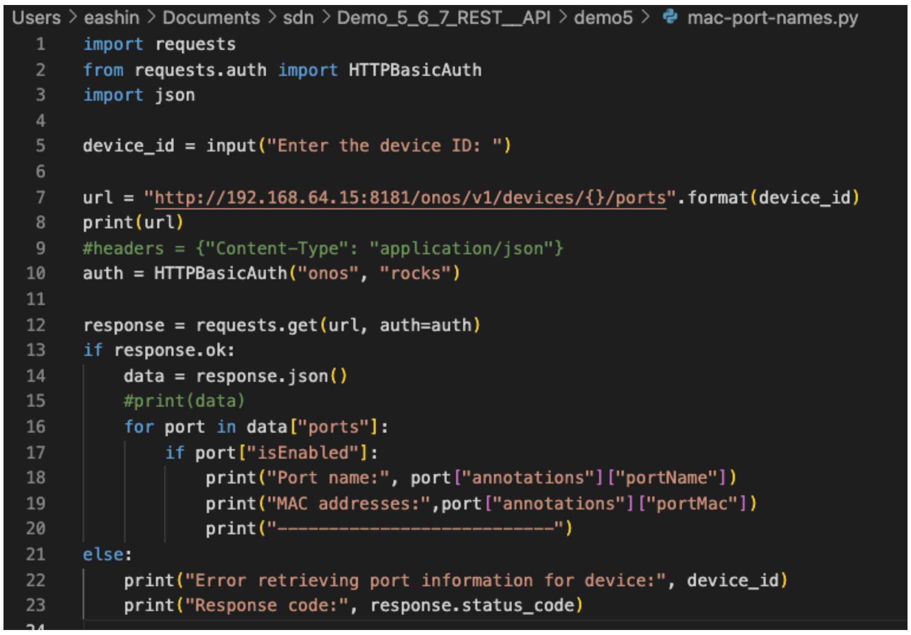
  Below is the expected output of IP Management Address and OpenFlow Version.



```
eashin@Eashins-MacBook-Pro  ~/Documents/sdn/Demo_5_6_7_REST__API/demo5  python3 mngtIp-ofVersion.py
Enter device ID: of:00007a8d87939c40
Device ID: of:00007a8d87939c40
IP Management Address: 172.17.0.1
OpenFlow Version: OF_14
eashin@Eashins-MacBook-Pro  ~/Documents/sdn/Demo_5_6_7_REST__API/demo5
```

- *Create a python program using the same device id, i.e., used in the previous question, to get the currently active MAC addresses and the Port names.*

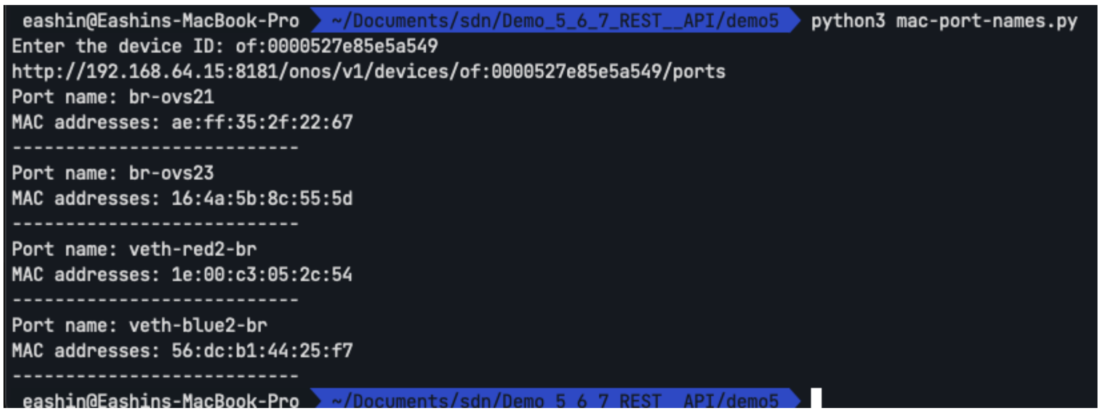
Python program that get ACTIVE MAC address and port names by a device ID.



Users > eashin > Documents > sdn > Demo_5_6_7_REST__API > demo5 > mac-port-names.py

```python
import requests
from requests.auth import HTTPBasicAuth
import json

device_id = input("Enter the device ID: ")

url = "http://192.168.64.15:8181/onos/v1/devices/{}/ports".format(device_id)
print(url)
#headers = {"Content-Type": "application/json"}
auth = HTTPBasicAuth("onos", "rocks")

response = requests.get(url, auth=auth)
if response.ok:
    data = response.json()
    #print(data)
    for port in data["ports"]:
        if port["isEnabled"]:
            print("Port name:", port["annotations"]["portName"])
            print("MAC addresses:",port["annotations"]["portMac"])
            print("--------------------------")
else:
    print("Error retrieving port information for device:", device_id)
    print("Response code:", response.status_code)
```

Below is expected output of the program wil MAC address and port names of the device.

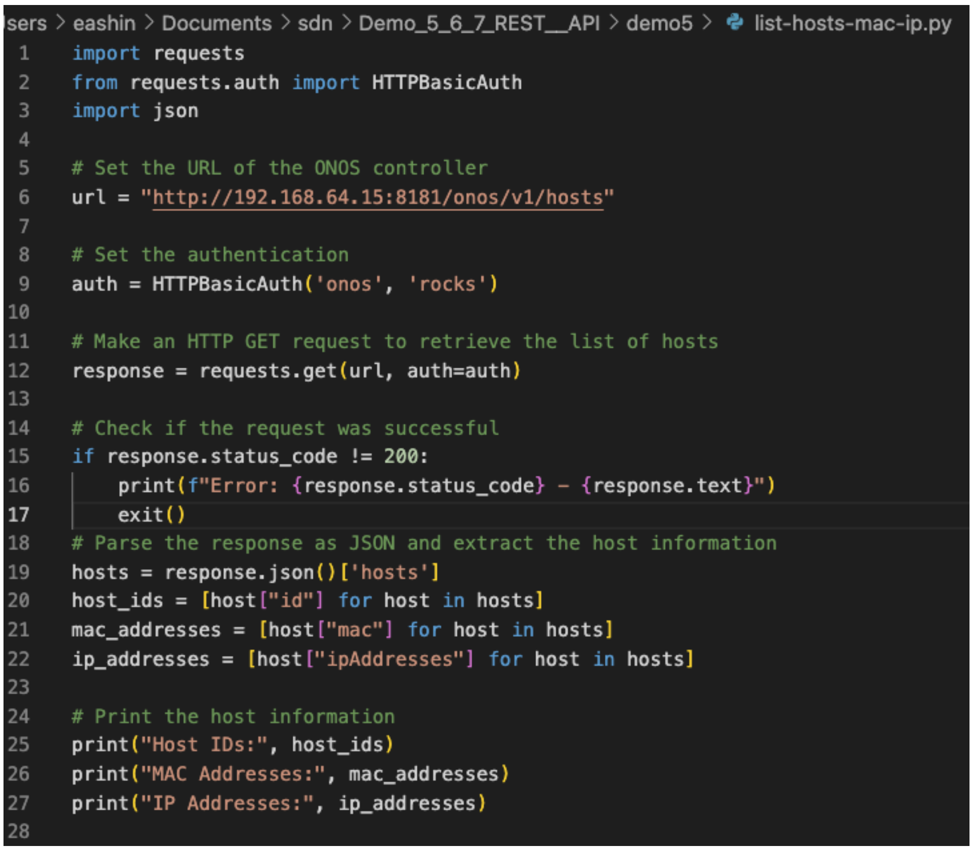


```
eashin@Eashins-MacBook-Pro  ~/Documents/sdn/Demo_5_6_7_REST__API/demo5  python3 mac-port-names.py
Enter the device ID: of:0000527e85e5a549
http://192.168.64.15:8181/onos/v1/devices/of:0000527e85e5a549/ports
Port name: br-ovs21
MAC addresses: ae:ff:35:2f:22:67
---------------------------
Port name: br-ovs23
MAC addresses: 16:4a:5b:8c:55:5d
---------------------------
Port name: veth-red2-br
MAC addresses: 1e:00:c3:05:2c:54
---------------------------
Port name: veth-blue2-br
MAC addresses: 56:dc:b1:44:25:f7
---------------------------
eashin@Eashins-MacBook-Pro  ~/Documents/sdn/Demo_5_6_7_REST__API/demo5
```

1.3.2. Task 3

- *Create a python program to list all available hosts by their id, MAC address, and IP address.*

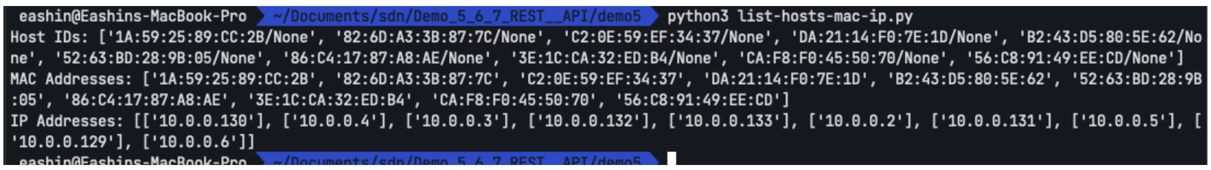
Here is the python program that list all available hosts by their id, MAC address, and IP address.



sers > eashin > Documents > sdn > Demo_5_6_7_REST__API > demo5 > list-hosts-mac-ip.py

```python
import requests
from requests.auth import HTTPBasicAuth
import json

# Set the URL of the ONOS controller
url = "http://192.168.64.15:8181/onos/v1/hosts"

# Set the authentication
auth = HTTPBasicAuth('onos', 'rocks')

# Make an HTTP GET request to retrieve the list of hosts
response = requests.get(url, auth=auth)

# Check if the request was successful
if response.status_code != 200:
    print(f"Error: {response.status_code} - {response.text}")
    exit()
# Parse the response as JSON and extract the host information
hosts = response.json()['hosts']
host_ids = [host["id"] for host in hosts]
mac_addresses = [host["mac"] for host in hosts]
ip_addresses = [host["ipAddresses"] for host in hosts]

# Print the host information
print("Host IDs:", host_ids)
print("MAC Addresses:", mac_addresses)
print("IP Addresses:", ip_addresses)
```

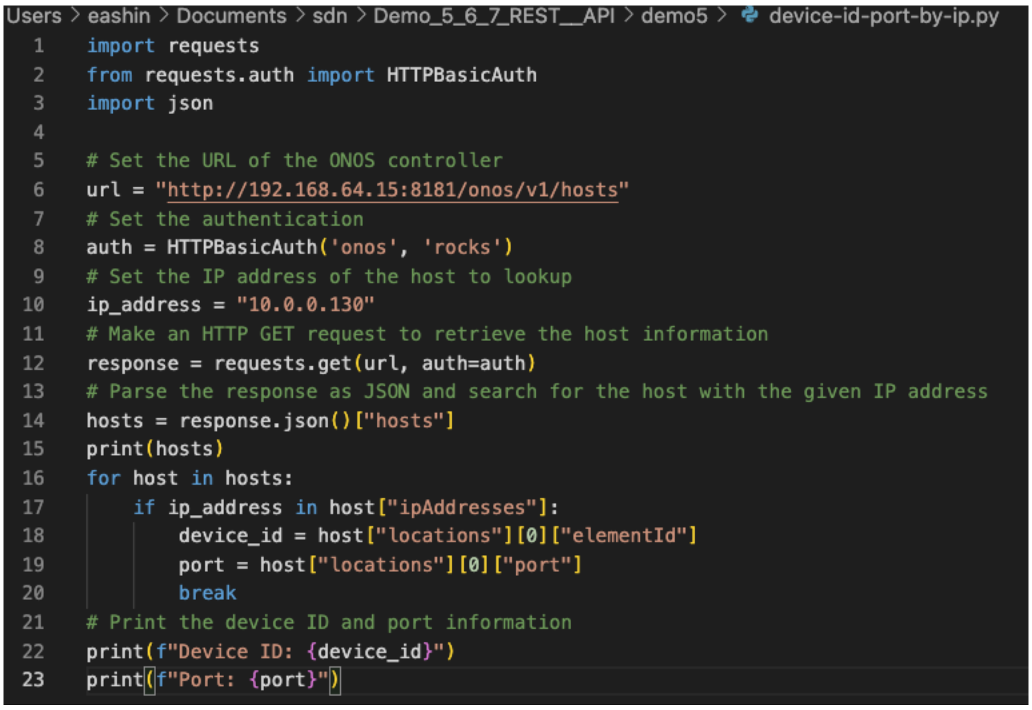
Below is the output of the program with all available Hosts IDs, MAC Addresses, and IP Addresses.



```
 eashin@Eashins-MacBook-Pro  ~/Documents/sdn/Demo_5_6_7_REST__API/demo5  python3 list-hosts-mac-ip.py
Host IDs: ['1A:59:25:89:CC:2B/None', '82:6D:A3:3B:87:7C/None', 'C2:0E:59:EF:34:37/None', 'DA:21:14:F0:7E:1D/None', 'B2:43:D5:80:5E:62/No
ne', '52:63:BD:28:9B:05/None', '86:C4:17:87:A8:AE/None', '3E:1C:CA:32:ED:B4/None', 'CA:F8:F0:45:50:70/None', '56:C8:91:49:EE:CD/None']
MAC Addresses: ['1A:59:25:89:CC:2B', '82:6D:A3:3B:87:7C', 'C2:0E:59:EF:34:37', 'DA:21:14:F0:7E:1D', 'B2:43:D5:80:5E:62', '52:63:BD:28:9B
:05', '86:C4:17:87:A8:AE', '3E:1C:CA:32:ED:B4', 'CA:F8:F0:45:50:70', '56:C8:91:49:EE:CD']
IP Addresses: [['10.0.0.130'], ['10.0.0.4'], ['10.0.0.3'], ['10.0.0.132'], ['10.0.0.133'], ['10.0.0.2'], ['10.0.0.131'], ['10.0.0.5'], [
'10.0.0.129'], ['10.0.0.6']]
```

- *Create a python program to get the device id and the port used by the host having “10.0.0.130" as an IP address in the pre-defined architecture.*

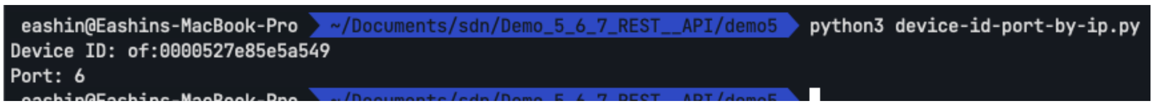
Python program below, get the device id and the port used by the host having IP “10.0.0.130"



Users > eashin > Documents > sdn > Demo_5_6_7_REST__API > demo5 > device-id-port-by-ip.py

```python
import requests
from requests.auth import HTTPBasicAuth
import json

# Set the URL of the ONOS controller
url = "http://192.168.64.15:8181/onos/v1/hosts"
# Set the authentication
auth = HTTPBasicAuth('onos', 'rocks')
# Set the IP address of the host to lookup
ip_address = "10.0.0.130"
# Make an HTTP GET request to retrieve the host information
response = requests.get(url, auth=auth)
# Parse the response as JSON and search for the host with the given IP address
hosts = response.json()["hosts"]
print(hosts)
for host in hosts:
    if ip_address in host["ipAddresses"]:
        device_id = host["locations"][0]["elementId"]
        port = host["locations"][0]["port"]
        break
# Print the device ID and port information
print(f"Device ID: {device_id}")
print(f"Port: {port}")
```

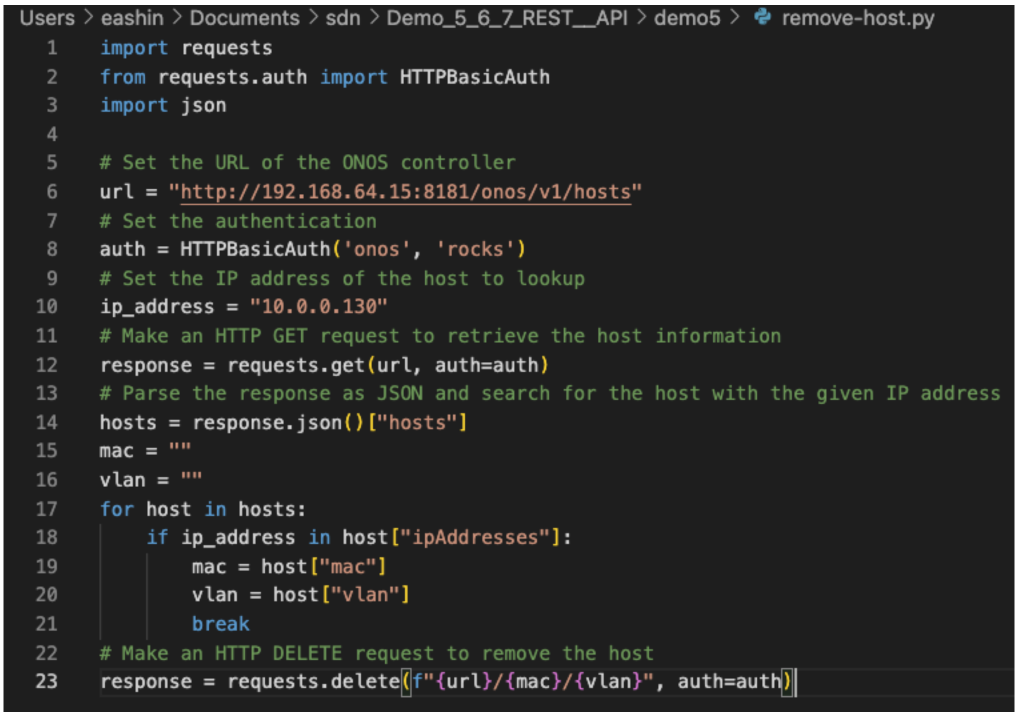
Below is the program output with expected Device ID and Port of the host with IP 10.0.0.130



```
 eashin@Eashins-MacBook-Pro  ~/Documents/sdn/Demo_5_6_7_REST__API/demo5  python3 device-id-port-by-ip.py
Device ID: of:0000527e85e5a549
Port: 6
```

- *Create a python program using the same host id, i.e., used in the previous question, to remove the host from the pre-defined architecture.*

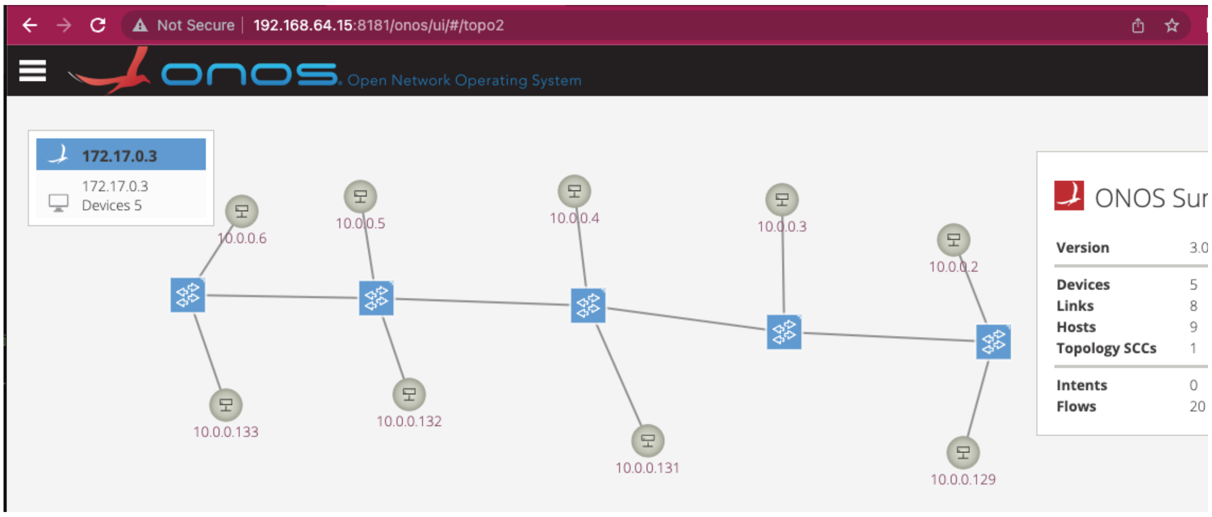
Below is a python script that remove the host from network topology.



Users > eashin > Documents > sdn > Demo_5_6_7_REST__API > demo5 > remove-host.py

```python
import requests
from requests.auth import HTTPBasicAuth
import json

# Set the URL of the ONOS controller
url = "http://192.168.64.15:8181/onos/v1/hosts"
# Set the authentication
auth = HTTPBasicAuth('onos', 'rocks')
# Set the IP address of the host to lookup
ip_address = "10.0.0.130"
# Make an HTTP GET request to retrieve the host information
response = requests.get(url, auth=auth)
# Parse the response as JSON and search for the host with the given IP address
hosts = response.json()["hosts"]
mac = ""
vlan = ""
for host in hosts:
    if ip_address in host["ipAddresses"]:
        mac = host["mac"]
        vlan = host["vlan"]
        break
# Make an HTTP DELETE request to remove the host
response = requests.delete(f"{url}/{mac}/{vlan}", auth=auth)
```

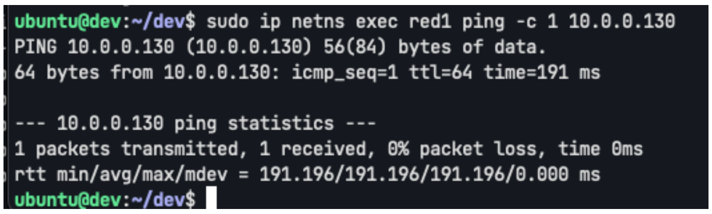
After removing the host disappears from the topology as shown in below ONOS GUI.



Version	3.0
Devices	5
Links	8
Hosts	9
Topology SCCs	1
Intents	0
Flows	20

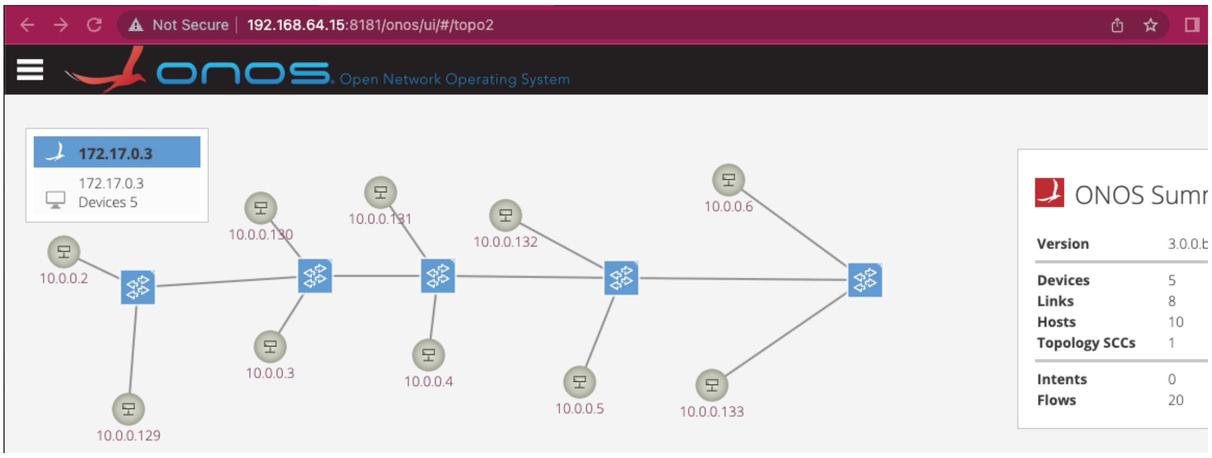
- *Ping the removed host, what do you observe?*

Pinged the host and the ping was successful



```
ubuntu@dev:~/dev$ sudo ip netns exec red1 ping -c 1 10.0.0.130
PING 10.0.0.130 (10.0.0.130) 56(84) bytes of data.
64 bytes from 10.0.0.130: icmp_seq=1 ttl=64 time=191 ms

--- 10.0.0.130 ping statistics ---
1 packets transmitted, 1 received, 0% packet loss, time 0ms
rtt min/avg/max/mdev = 191.196/191.196/191.196/0.000 ms
ubuntu@dev:~/dev$
```

Below is the ONOS GUI after the successful ping to the removed host, it again connected to the network topology.

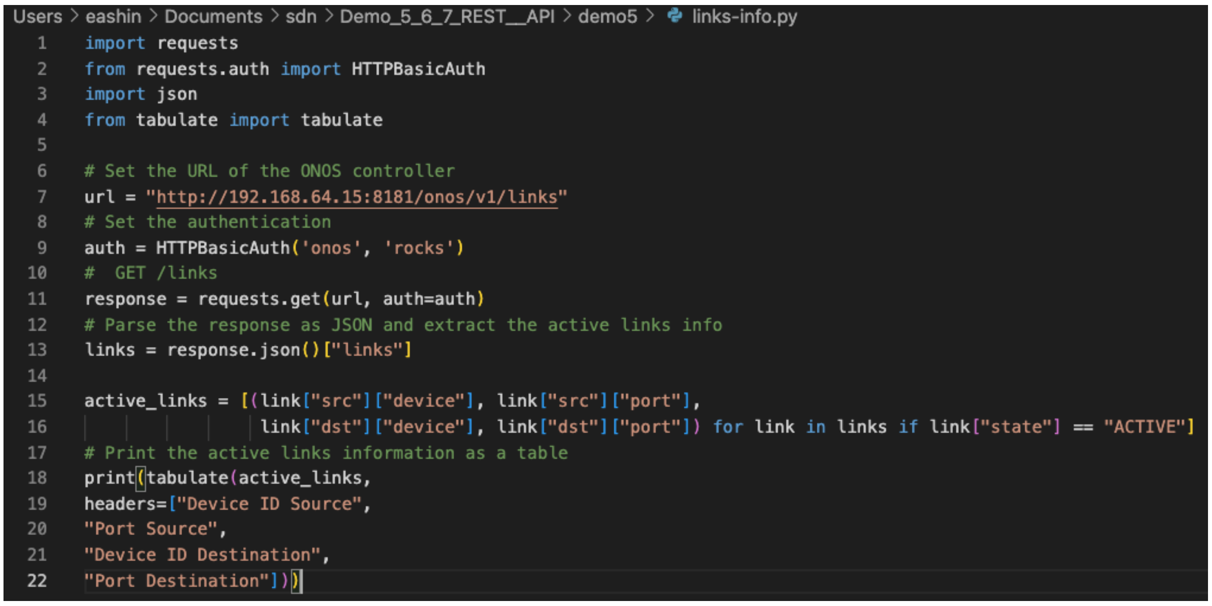


Version	3.0.0.b
Devices	5
Links	8
Hosts	10
Topology SCCs	1
Intents	0
Flows	20

1.3.3. Task 4

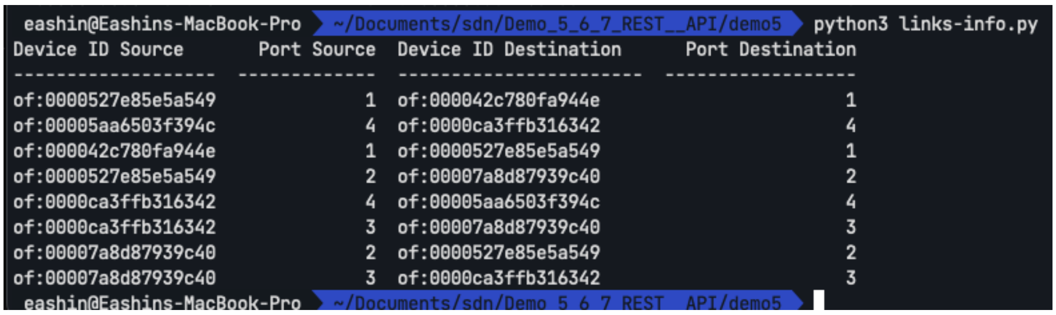
- *Create a python program to list all ACTIVE links in the pre-defined topology, the output should be a table containing device id source, port source, device id destination, port destination.*

Here is the python program which list all ACTIVE links and provide a table containing device id source, port source, device id destination, port destination.



```
Users > eashin > Documents > sdn > Demo_5_6_7_REST__API > demo5 > links-info.py
import requests
from requests.auth import HTTPBasicAuth
import json
from tabulate import tabulate

# Set the URL of the ONOS controller
url = "http://192.168.64.15:8181/onos/v1/links"
# Set the authentication
auth = HTTPBasicAuth('onos', 'rocks')
#  GET /links
response = requests.get(url, auth=auth)
# Parse the response as JSON and extract the active links info
links = response.json()["links"]

active_links = [(link["src"]["device"], link["src"]["port"],
                 link["dst"]["device"], link["dst"]["port"]) for link in links if link["state"] == "ACTIVE"]
# Print the active links information as a table
print(tabulate(active_links,
headers=["Device ID Source",
"Port Source",
"Device ID Destination",
"Port Destination"]))
```

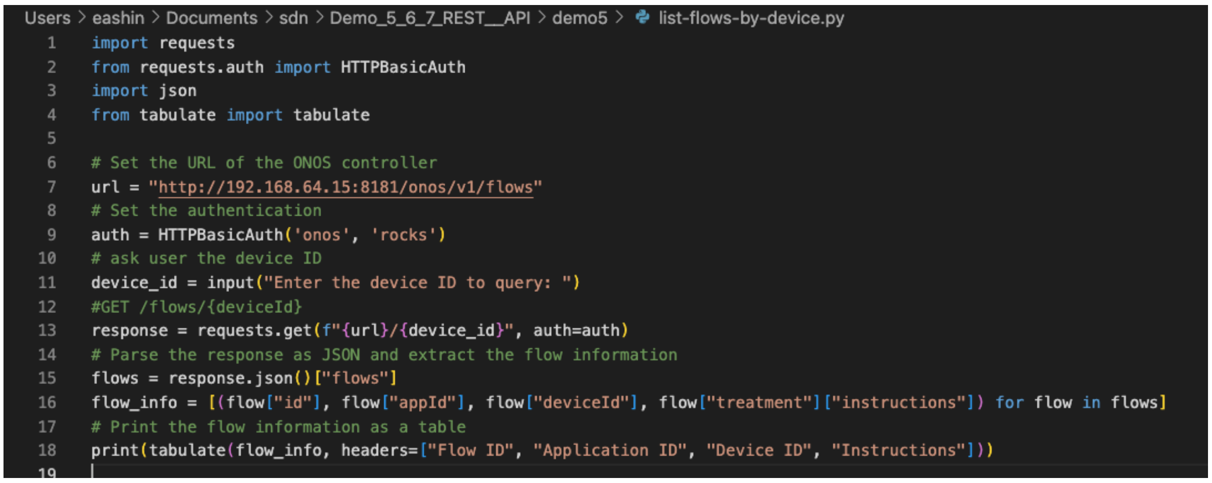
Below is the expected table with all informations.



```
eashin@Eashins-MacBook-Pro  ~/Documents/sdn/Demo_5_6_7_REST__API/demo5  python3 links-info.py
Device ID Source      Port Source  Device ID Destination      Port Destination
--------------------  -----------  -----------------------  ------------------
of:0000527e85e5a549             1  of:000042c780fa944e                       1
of:00005aa6503f394c             4  of:0000ca3ffb316342                       4
of:000042c780fa944e             1  of:0000527e85e5a549                       1
of:0000527e85e5a549             2  of:00007a8d87939c40                       2
of:0000ca3ffb316342             4  of:00005aa6503f394c                       4
of:0000ca3ffb316342             3  of:00007a8d87939c40                       3
of:00007a8d87939c40             2  of:0000527e85e5a549                       2
of:00007a8d87939c40             3  of:0000ca3ffb316342                       3
eashin@Eashins-MacBook-Pro  ~/Documents/sdn/Demo_5_6_7_REST__API/demo5 
```

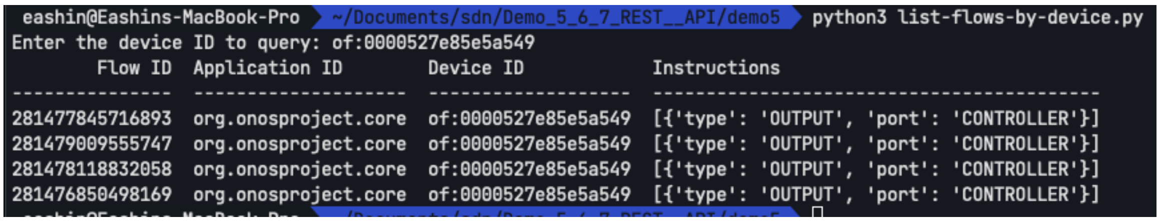
- *Create a python program to list all the flows applied to a device of your choice, the output may show the flow-id, the application id, the device id, and the instructions.*

Below is a python program to show the flow-id, the application id, the device id, and the instructions.



```
Users > eashin > Documents > sdn > Demo_5_6_7_REST__API > demo5 > list-flows-by-device.py
import requests
from requests.auth import HTTPBasicAuth
import json
from tabulate import tabulate

# Set the URL of the ONOS controller
url = "http://192.168.64.15:8181/onos/v1/flows"
# Set the authentication
auth = HTTPBasicAuth('onos', 'rocks')
# ask user the device ID
device_id = input("Enter the device ID to query: ")
#GET /flows/{deviceId}
response = requests.get(f"{url}/{device_id}", auth=auth)
# Parse the response as JSON and extract the flow information
flows = response.json()["flows"]
flow_info = [(flow["id"], flow["appId"], flow["deviceId"], flow["treatment"]["instructions"]) for flow in flows]
# Print the flow information as a table
print(tabulate(flow_info, headers=["Flow ID", "Application ID", "Device ID", "Instructions"]))
```

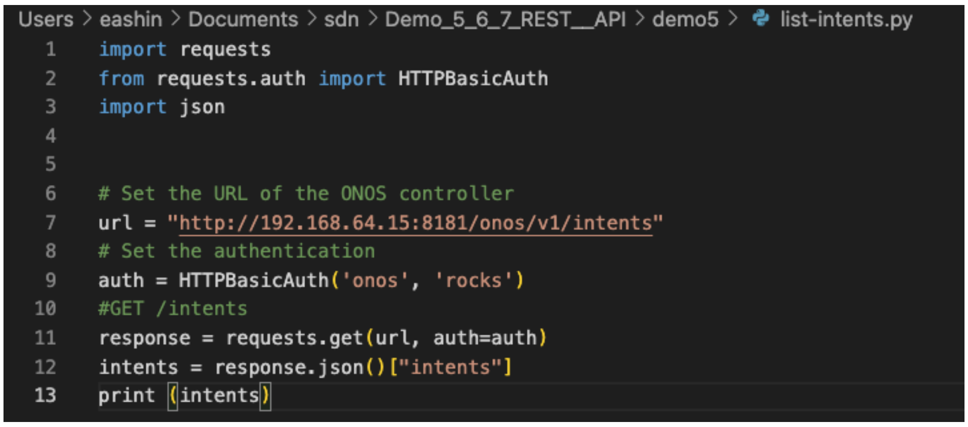
Below is the output of the program as a table contents of required info of a given device id.



```
 eashin@Eashins-MacBook-Pro  ~/Documents/sdn/Demo_5_6_7_REST__API/demo5  python3 list-flows-by-device.py
Enter the device ID to query: of:0000527e85e5a549
        Flow ID  Application ID        Device ID            Instructions
---------------  --------------------  -------------------  ------------------------------------------
281477845716893  org.onosproject.core  of:0000527e85e5a549  [{'type': 'OUTPUT', 'port': 'CONTROLLER'}]
281479009555747  org.onosproject.core  of:0000527e85e5a549  [{'type': 'OUTPUT', 'port': 'CONTROLLER'}]
281478118832058  org.onosproject.core  of:0000527e85e5a549  [{'type': 'OUTPUT', 'port': 'CONTROLLER'}]
281476850498169  org.onosproject.core  of:0000527e85e5a549  [{'type': 'OUTPUT', 'port': 'CONTROLLER'}]
```

- *Create a python program to list all intents.*

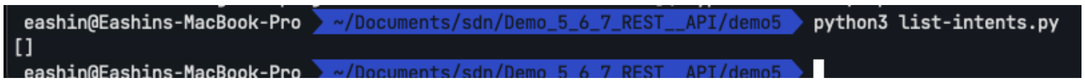
Here is the python script which list all intents



Users > eashin > Documents > sdn > Demo_5_6_7_REST__API > demo5 > list-intents.py

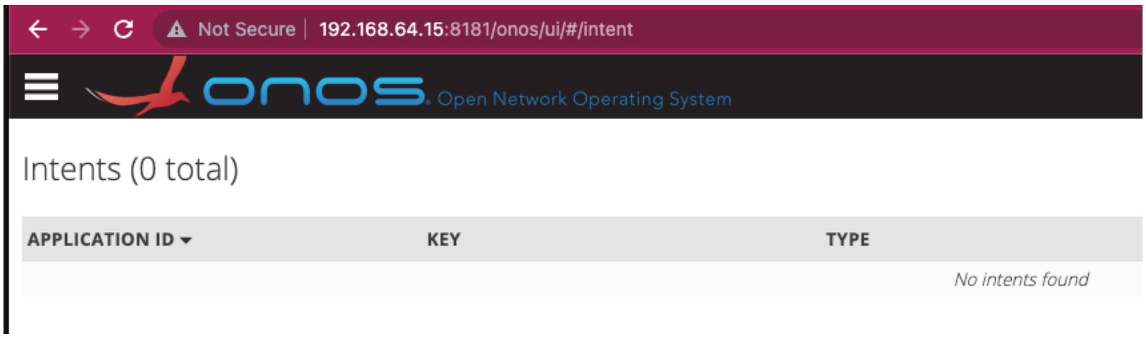
```python
import requests
from requests.auth import HTTPBasicAuth
import json


# Set the URL of the ONOS controller
url = "http://192.168.64.15:8181/onos/v1/intents"
# Set the authentication
auth = HTTPBasicAuth('onos', 'rocks')
#GET /intents
response = requests.get(url, auth=auth)
intents = response.json()["intents"]
print (intents)
```

Below is the program output as empty



```
 eashin@Eashins-MacBook-Pro  ~/Documents/sdn/Demo_5_6_7_REST__API/demo5  python3 list-intents.py
[]
 eashin@Eashins-MacBook-Pro  ~/Documents/sdn/Demo_5_6_7_REST__API/demo5 
```

I check ONOS GUI to validate that there is no intents



Not Secure | 192.168.64.15:8181/onos/ui/#/intent

Open Network Operating System

Intents (0 total)

APPLICATION ID	KEY	TYPE
		No intents found