Lab – Create a Network Device Inventory in Python (Instructor Version)

**Instructor Note**: Red font color or gray highlights indicate text that appears in the instructor copy only.

1. Objectives

Part 1: Investigate the Network Device Inventory API

Part 2: Modify the Code

1. Background / Scenario

An important skill for software developers is the ability to obtain, modify, and reuse code that was previously created. Coder communities like GitHub and Stack Overflow, among others, provide support for developers and many people freely share code there. Obtaining and modifying code for a specific need is an import skill for adding efficiency to the software development process.

In this lab, you will modify the code that was created to obtain the inventory of network hosts and repurpose it to obtain an inventory of network devices.

1. Required Resources

* Access to the APIC-EM in the DevNet sandbox at https://{YOUR-APICEM}.cisco.com
* Postman
* Python 3 with IDLE
* Python **requests** module
* Python **tabulate** module
* The functions file that you have created
* The **print\_hosts.py** file that you created or the **print\_hosts\_sol.py** file
* Access to the Internet

1. Investigate the Network Device Inventory API

In this part of the lab, you will investigate the documentation for the APIC-EM’s network device inventory to determine the necessary information you need to create your program.

* + 1. Determine the endpoint URL.
       1. Login to the APIC-EM sandbox using the URL and credentials supplied by your instructor and click **API** to access the Swagger API documentation:
       2. Under Available APIs, click **Inventory > network-device > GET /network-device**.
       3. Click **Try it out!** What is the URL that you will use in the **request.get()** method for this endpoint?

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

https://{YOUR-APICEM}.cisco.com/api/v1/network-device

* + 1. Investigate the response JSON.
       1. Copy the JSON in the Response Body and paste it into <https://codebeautify.org/jsonviewer>.
       2. Compare **GET /network-device** JSON to the **GET /host** JSON you viewed in the previous lab. How is the structure of the network device inventory JSON similar to the structure of the host inventory JSON?

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Both have the list of network devices held in the **response** key. Each device is represented by a numeric list element.

* + - 1. How many network devices are included in the inventory?

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Answers may vary, but there should be around 12 or 14 devices present. It can be very depending on the state of the APIC-EM sandbox network.

* + - 1. You want to access and display information for the network devices similar to the information that are displayed for the hosts. However, the keys do not use the names hostType and hostIP. What are the names of similar keys for the network devices?

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Similar keys are the **type** and **managementIpAddress** keys.

* + - 1. Look for other cosmetic impacts to the code. For example, there are status and error messages that are displayed to the user. They may require modification.

1. Modify the Code.
   * + 1. Open your **print\_hosts.py** file and save it as **print\_devices.py**.
       2. Locate the places that require changes and make the edits.

The student must change the value of the api\_url, update the print statements from /host to /network-device, change the name of the “host” variable to “device”, “host\_list” to “device\_list” as in the following code:

api\_url = "https://{YOUR-APICEM}.cisco.com/api/v1/network-device"

device = [ i, item["type"], item["managementIpAddress"] ]

device\_list.append( device )

* + - 1. Save and run your code. Investigate errors. If necessary, compare your file to the **print\_devices\_sol.py** file to discover the source of the errors.
      2. Create a function from your code by copying the code into your **my\_apic\_em\_functions.py** file and transforming it into a function called **print\_devices()**.
      3. Run your function file and test each function to make sure your code is error free.

1. Reflection
   1. What are the advantages of modifying and reusing code in this way?

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Efficiency is the big advantage. There are potentially fewer errors in modifying code that already works.

* 1. What are the challenges?

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

It is easy to miss things that should be changed. In addition, by making edits it is possible to introduce new errors. If you are unfamiliar with the code or do not understand what it is doing, it can be very difficult to locate these errors.

* 1. What can developers do to make the process easier?

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Documenting code with detailed comments is one thing. Improving modularity is another. By removing redundancies between code and moving those functionalities to reusable functions, code become simpler and easier to modify.