-*- coding: utf-8 -*-
"""Python + Rest Hands on and Working with Network Data in Python

Automatically generated by Colaboratory.

Original file is located at
    https://colab.research.google.com/drive/1cimwUGn8EDqLne39inWJ4F5wSPKbD4Cl

#Python + Rest and Hands on Working with Network Data in Python

What this is: an interactive notebook which will provide a framework you can keep, that reviews important Python/Rest Items and then works with some DNAC and Meraki data. You will run precanned scripts for the Meraki and DNAC Data, and you will not need to know How to do this.

But you will investigate the data from each, and create a consolidated device list of Meraki and DNAC Data through using JSON data and Python dictionaries and lists.

Before you even start... You do need a google account. and this is a read only sheet. To make the changes you need to, you need to go to File/Save a copy to drive.
This will give you a Read Write copy, that you can modify.

IF YOU DON'T WANT TO USE COLABORATORY THE CODE IN THIS DOC CAN BE PASTED INTO PYTHON WITH THE REQUESTS LIBRARY AND JUST EXECUTED, LIKE THEY DID IN THE 80s

---

##1.1 Python example applications for Networking.
"""

Before we really do much of anything, we need to understand some numbers are Strings. and some are Integers. And that is OK.

For example, run this. mouse over this cell, and hit the run triangle int he upper left corner.

```
color = 'red'
bird = 'roadrunner'
print('if i want to add two words together it looks like this',color+' '+bird)

num1 = 602
```

```
num2 = 8101312
print('when i add two numbers together its 602 and 8101312 :: ',num1+num2)
```

That is pretty fancy right? but thats wrong in my context. That is my phone number, you cant reach me at 8101914.

these numbers are strings in this context. we want to concatenate them. We want python to treat them as strings using str(the number)

```
print('my phone number is ',str(num1)+str(num2))
```

"""----------
thats great but why would i care? Because BGP AS are also numbers. but we never add them. for example, this code segment will throw up on you.

Exercise 1.1 - Flags 1 and 2
"""

```
for AS in range(65000,65005):
    print("router bgp "+AS)   ############## This will give an error. it is expected, it is the flag.
```

See? Thats disgusting. Don't ever do that again. Lets do this instead.

```
for AS in range(65000,65005):
  print("router bgp "+str(AS))
```

"""----------

1.2 Basic Python stuff, for loops.

For loops will be used many times so understanding basics is good. We'll use an example of generating a list of IP Addresses to Ping.

As you can see in the BGP Example, when using the range command, there are two values
- Start value
- stop value

```
the for loop will check BEFORE executing the last value. so if you do
for i in range(1,5):
  print(i)

  it will stop at 5 BEFORE executing the code... so basically, the last iteration is 4.
"""

for i in range(1,5):
  print(i)

"""--------------------
So if we wanted a range of IP that went 1 to 254... we'd need to... go to... you know... modify the below code to make it work.
"""

# flag below!

for octet4 in range(1,254):              # we call it octet 4 because later we may do octet 3
  print("ping ip 10.1.1."+str(octet4))    # remember the "string cheese incident" form above?
    # we have to convert the number to string (like above) or things dont work well when we add the number into the string.

"""------------------------

In addition to doing range stuff we may also do lists... like below.
"""

for interface in ["gig0/1","gig0/2","tengig1/0"]:
  print('interface '+ interface)
  print('shutdown')

  ## if you notice it does do all of them, its only the range command that has the "add one extra because its "up to, not including"

"""--------------------

Nested loop
```

```
those are cool. Sometimes you will want to nest a loop. this is an example of that that you can modify to get another flag.
"""

### To make the below print a ping script for everything in range 10.1.1.1 - 10.1.3.254, we need to change TWO lines below.
####### Flag
for octet3 in range(1,4):
  for octet4 in range(1,4):
    print('ping 10.1.'+str(octet3)+'.'+octet4)

"""##2.1 - Get some DNAC Data and make meaning of it."""

import requests
from requests.auth import HTTPBasicAuth
import json


### Get a token, specific to DNAC,
DNAC='sandboxdnac.cisco.com'
DNAC_USER='devnetuser'
DNAC_PASSWORD='Cisco123!'
INTENT_API = "https://sandboxdnac.cisco.com/dna/intent/api/v1/"

url = 'https://sandboxdnac.cisco.com/dna/system/api/v1/auth/token'
resp = requests.post(url, auth=HTTPBasicAuth(DNAC_USER, DNAC_PASSWORD),verify=False)
print(resp.text)      #<<<<<<<<<<< You should see your token)
token = resp.json()['Token']   # We set our token.


## Token = Authentication,
url = INTENT_API+"network-device"   # we set our new URL to get devices
hdr={'x-auth-token': token}        #we set our header. this is the token format for DNAC.
resp = requests.get(url, headers=hdr,verify=False)
dnac_list = resp.json()
print(json.dumps(dnac_list,indent=2))

## Modify the below to replace "roleSource" with the platformId json key. paste the output into the flag.
######## Flag

for device in dnac_list['response']:
        print(device['hostname'],device['softwareVersion'],device['roleSource'])
```

"""==========
==========

3.1 Get Some Meraki Data and make meaning of it.

If you get rate limit based API issues, wait a minute, and try again.

If you DONT WANT TO WAIT... Go to the end of this notebook and it will statically load the data from exercise 3.1, so that everything else will work. This is a workaround. The code works, so well we're rate limited.

Interesting note. All our Rest based systems use basically the same format.
URAH - "U R A HUMAN" for URL, Request library, Authentication, and Header for getting data.
See below for how easy it is to get information from Meraki. (Webex teams is the same, different URL and Authentication)
"""

Exercise 3.1

This is all you need to get information from Meraki or webex teams
TOKEN = '6bec40cf957de430a6f1f2baa056b99a4fac9ea0' ### Authentication
URL="https://api.meraki.com/api/v0/organizations/549236/networks/L_646829496481105433/devices" # URL

HDR={'X-Cisco-Meraki-API-Key': TOKEN}   #### This is the meraki header needed, - HEADER

NETWORKS=requests.get(URL,headers=HDR) ### do you see how easy this is, you have the URL, the authentication, and the header. Now we just do a request.

########### Thats it. The below lines really format it so you can read it

NET_JSON=json.loads(NETWORKS.text)
print(json.dumps(NET_JSON,indent=2))

"""### The below works just fine to create a list with Model and Firmware and Lattitude, which makes it basically worthless.
Look above at the meraki Json output, and you want to chose a Json key that will print an IP for each device... it should be the "most consistent key" for all the meraki devices.

```
### Fix the report to print "model, firmware, IP Address"
"""

## This basically is the same as Section 2.2.
## Hint, if you choose the WRONG IP key from above list (wan1Ip, etc...) your code will error out. because not all devices have a Wan Ip... you know?
meraki_list=NET_JSON
for device in meraki_list:
        print(device['model'],device['firmware'],device['lat'])

"""=========================
=========================


###4 creating a single list of IP of all meraki and dnac devices to use as a Seedfile.

You can play with this, or just execute this. This is meant to just show you how you "can". For example, you can comment out the "if device lanIp" line and de indent the print device LanIp line to see the "none" pop up.
"""

for device in dnac_list['response']:
  print(device['managementIpAddress'])
for device in meraki_list:
  if (device['lanIp']):    #### If you see the above for one of the devices it has a null lanIp. this isn't a string like "None" or "null", its boolean no. its blank. so we check to see if its present.
    print(device['lanIp'])
print("this concludes getting a single seedfile list from dnac and meraki")
print('this is pretty trivial when dealing with 5 - 10 IP. but useful when you have "lots of stuff"')
print("mission accomplished")

"""Is the Mission Accomplished?

Do you need the Meraki Data Loaded because a lot of SE are doing the same thing?

#To Manually load and see the Meraki Data, click the below box :)

if you are doing this, this is the exact data you would get if the API rate limiters let you. Use this output for the first meraki flag, and you can complete the rest of the mission from here.
```

"""

null=None
NETWORKS2=json.loads('[{"lat":37.4180951010362,"lng":-122.098531723022,"address":"","serial":"Q2QN-9J8L-SLPD","mac":"e0:55:3d:17:d4:23","wan1Ip":"10.10.10.116","wan2Ip":null,"lanIp":"10.10.10.116","url":"https://n149.meraki.com/DevNet-Sandbox-A/n/9tIK5cvc/manage/nodes/new_list/246656701813795","networkId":"L_646829496481105433","model":"MX65","firmware":"wired-14-42","floorPlanId":null},{"lat":-6.97817626419744,"lng":107.631476116217,"address":"Puri Kharisma","serial":"Q2MD-BHHS-5FDL","mac":"88:15:44:60:21:10","lanIp":null,"url":"https://n149.meraki.com/DevNet-Sandbox-A/n/raXeAcvc/manage/nodes/new_list/149624922841360","networkId":"L_646829496481105433","name":"Testing-Devnet","model":"MR53","firmware":"wireless-26-6-1","floorPlanId":null},{"lat":37.4180951010362,"lng":-122.098531723022,"address":"","serial":"Q2HP-F5K5-R88R","mac":"88:15:44:df:f3:af","lanIp":"192.168.128.2","url":"https://n149.meraki.com/DevNet-Sandbox-A/n/EFZ1Davc/manage/nodes/new_list/149624931218351","networkId":"L_646829496481105433","model":"MS220-8P","switchProfileId":null,"firmware":"switch-11-31","floorPlanId":null}]')
NET_JSON=NETWORKS2
print(json.dumps(NET_JSON,indent=2))