Exercise: Network Configuration Parser

Chandra Lingam

Python Regular Expressions https://www.udemy.com/python-regular-expressions/

Introduction

In this exercise, you will implement a regular expression pattern to parse and extract network configuration data. We would like to thank Girish U for providing this example problem.

For this exercise, we encourage you to review or use:

1. Relevant course videos
2. PythonRegularExpressionsQuickReference.pdf – for syntax help
3. Interactive Tool for writing and testing pattern
4. Implement the final solution using python notebook (sample snippet of code provided at the end of this document)

Help or Questions

Use course Q&A for any help needed to complete the assignment.

Completed pattern is provided in the course Q&A. We strongly encourage you to try this on your own and complete it.

Problem

Data consists network interface detail spanning multiple lines. A single file can contain many network interfaces information. In the below example, it has two interface entries. Assume that each interface can have at most two ip address rows.

Data Example

```
interface Gi1/0/1

vrf red

ip address secondary 2.2.2.2 255.255.255.0

ip address 1.1.1.1 255.255.255.0


interface Gi1/0/2

vrf blue

ip address  3.3.3.3 255.255.255.0
```

Output Expected

You need to write a pattern to extract details of an interface and present it as a dictionary:

```
Dictionary

   {'ip2': '1.1.1.1 255.255.255.0', 'interface': 'Gi1/0/1', 'vrf': ' red', 'ip':
'2.2.2.2 255.255.255.0'}

Dictionary

   {'ip2': None, 'interface': 'Gi1/0/2', 'vrf': ' blue', 'ip': '3.3.3.3 255.255.2
55.0'}
```

Code Snippet

```
def find_network_example():

    # PATTERN
    pattern = r""

    # DATA
    text = r'''

'''

    # successful match
    match_iter = re.finditer(pattern, text)

    for match in match_iter:
        print ('Dictionary')
        print ('   {0}'.format(match.groupdict()))
```

Bonus Problem

In the previous example, the assumption was each interface can have at most two IP address rows. How would the solution change if there can be any number of IP Address rows for an interface?

Hint: Use one pattern to capture details for the entire interface. Use a second pattern to iterate IP rows and print values as a dictionary.

```
Output: Interface 1

   {'interface': 'Gi1/0/1', 'vrf': ' red'}

   {'ip': '1.1.1.1 255.255.255.0'}
```

```
   {'ip': '2.2.2.2 255.255.255.0'}
Output: Interface 2
   {'interface': 'Gi1/0/2', 'vrf': ' blue'}
   {'ip': '3.3.3.3 255.255.255.0'}
```