From: Kirk Byers
Sent: Tuesday, December 27, 2022 9:00 PM
To: sydasif78@hotmail.com
Subject: [PyNet Learning Python] - Lesson4 / Dictionaries, Exceptions, and Regular Expressions

Note: There is a table of contents for each video at the bottom of this email including timestamps to where various content is located. This should be helpful in navigating the videos.

In this email of Learning Python we are going to cover the following:

1. **Dictionaries**
   Video https://vimeo.com/246157566
   Length is 6 minutes

2. **Dictionary Methods**
   Video https://vimeo.com/246163031
   Length is 7 minutes

3. **Sets**
   Video https://vimeo.com/246167477
   Length is 9 minutes

4. **Exceptions**
   Video https://vimeo.com/246174686
   Length is 15 minutes

5. **Regular Expressions (Part1)**
   Video https://vimeo.com/246184715
   Length is 15 minutes

6. **Regular Expressions (Part2)**
   Video https://vimeo.com/246532117
   Length is 7 minutes

7. **Regular Expressions (Part3)**
   Video https://vimeo.com/246534450
   Length is 8 minutes

8. **Regular Expressions, Other Methods**
   Video https://vimeo.com/246535038
   Length is 4 minutes

Additional Content:

Regular Expression Tutorial
This is a good resource if you are new to regular expressions.

Online Regular Expression Tester
Select 'Python' on the left-hand side.

Python Regular Expression HowTo
This is a good overview of regular expression special characters.
Start at the very top of the page and read through the 'Repeating Things' section.

Automate the Boring Stuff - Dictionaries and Structuring Data
Read through the 'The GET() Method' section.

Exercises

Reference code for these exercises is posted on GitHub at:
https://github.com/ktbyers/pynet/tree/master/learning_python/lesson4

1. Create a dictionary representing a network device. The dictionary should have key-value pairs representing the 'ip_addr', 'vendor', 'username', and 'password' fields.

Print out the 'ip_addr' key from the dictionary.

If the 'vendor' key is 'cisco', then set the 'platform' to 'ios'. If the 'vendor' key is 'juniper', then set the 'platform' to 'junos'.

Create a second dictionary named 'bgp_fields'. The 'bgp_fields' dictionary should have keys for 'bgp_as', 'peer_as', and 'peer_ip'.

Using the .update() method add all of the 'bgp_fields' dictionary key-value pairs to the network device dictionary.

Using a for-loop, iterate over the dictionary and print out all of the dictionary keys.

Using a single for-loop, iterate over the dictionary and print out all of the dictionary keys and values.

2. Create three separate lists of IP addresses. The first list should be the IP addresses of the Houston data center routers, and it should have over ten RFC1918 IP addresses in it (including some duplicate IP addresses).

The second list should be the IP addresses of the Atlanta data center routers, and it should have at least eight RFC1918 IP addresses (including some addresses that overlap with the Houston data center).

The third list should be the IP addresses of the Chicago data center routers, and it should have at least eight RFC1918 IP addresses. The Chicago IP addresses should have some overlap with both the IP addresses in Houston and Atlanta.

Convert each of these three lists to a set.

Using a set operation, find the IP addresses that are duplicated between Houston and Atlanta.

Using set operations, find the IP addresses that are duplicated in all three sites.

Using set operations, find the IP addresses that are entirely unique in Chicago.

3. Read in the 'show_version.txt' file. From this file, use regular expressions to extract the OS version, serial number, and configuration register values.

Your output should look as follows:

```
OS Version: 15.4(2)T1
Serial Number: FTX0000038X
Config Register: 0x2102
```

4. Using a named regular expression (?P<name>), extract the model from the below string:

```
show_version = '''
Cisco 881 (MPC8300) processor (revision 1.0) with 236544K/25600K
bytes of memory.
Processor board ID FTX0000038X

5 FastEthernet interfaces
1 Virtual Private Network (VPN) Module
256K bytes of non-volatile configuration memory.
126000K bytes of ATA CompactFlash (Read/Write)
'''
```

Note that, in this example, '881' is the relevant model. Your regular expression should not, however, include '881' in its search pattern since this number changes across devices.

Using a named regular expression, also extract the '236544K/25600K' memory string.

Once again, none of the actual digits of the memory on this device should be used in the regular expression search pattern.

Print both the model number and the memory string to the screen.

5. Read the 'show_ipv6_intf.txt' file.

From this file, use Python regular expressions to extract the two IPv6 addresses.

The two relevant IPv6 addresses you need to extract are:

```
2001:11:2233::a1/24
2001:cc11:22bb:0:2ec2:60ff:fe4f:feb2/64
```

Try to use the re.DOTALL flag as part of your search. Your search pattern should not include any of the literal characters in the IPv6 address.

From this, create a list of IPv6 addresses that includes only the above two addresses.

Print this list to the screen.

CLASS OUTLINE

1. Dictionaries (VIDEO1)
   A. What is a dictionary? [0:03]
   B. Why do we need dictionaries? [0:40]
   C. How to create a dictionary [1:33]
   D. Adding keys to a dictionary [1:58]
   E. Using variables as keys [2:52]
   F. Accessing keys in a dictionary [4:00]
   G. Accessing keys that don't exist [4:32]
   H. Dictionaries are mutable [5:05]

2. Dictionary Methods (VIDEO2)
   A. get() method [0:34]
   B. copy() method [2:23]
   C. pop() method for key removal [3:01]
   D. update() method [3:49]
   E. Looping over dictionaries [5:04]
   F. Using .values() [5:32]
   G. Using .items() [6:01]

3. Sets (VIDEO3)
   A. What is a set? [0:04]
      1. Each element has to be unique
      2. Sets are unordered
      3. No indices
   B. Looping over a set [2:00]
   C. Set operations [2:23]
      1. Union [2:39]
      2. Intersection [3:33]
      3. Difference [4:11]
      4. Symmetric diff [5:53]

4. Exceptions (VIDEO4)
   A. Example exception [0:19]
   B. Non-zero exit status [2:22]
   C. Using try/except [2:38]
   D. Catching an exception and re-raising [6:33]
   F. Catching an exception and printing [8:22]
   G. Catching multiple exceptions [9:34]
   H. Adding a finally clause [12:00]

5. Regular Expressions (Part1) (VIDEO5)

```
   A. Why we need regex?   [0:05]
   B. Some special characters   [1:24]
      1. Any single character: . [1:33]
      2. One or more times: +   [4:54]
      3. Zero or more times: *   [5:56]
      4. Beginning of line: ^   [6:37]
      5. End of line: $   [6:55]
      6. Digit character class: \d   [7:30]
      7. Whitespace character class: \s   [9:29]
      8. Non-whitespace: \S   [10:20]
      9. Construct your own character class: []   [11:11]
      10. Parenthesis to save things: ()   [12:23]
   C. By default will be greedy.   [5:35]

6. Regular Expressions (Part2) (VIDEO6)
   A. Why you always use raw strings   [0:04]
   B. Using re.search()   [1:21]
      1. Arguments   [2:06]
      2. If successful, match object   [2:30]
         a. match.group(0)   [2:34]
         b. match.group(1)   [3:48]
   C. Constructing named patterns   [4:55]
      1. match.groupdict()   [5:55]

7. Regular Expressions (Part3) (VIDEO7)
   A. Disabling greedy behavior   [0:31]
   B. Flags to re.search   [3:02]
      1. re.M  [4:58]
      2. re.DOTALL   [6:50]
      3. re.I   [7:41]

8. Regular Expressions, Other Methods (VIDEO8)
   A. re.split()   [0:14]
   B. re.sub()   [1:48]
   C. re.findall()   [3:38]
```

Kirk Byers
https://pynet.twb-tech.com
