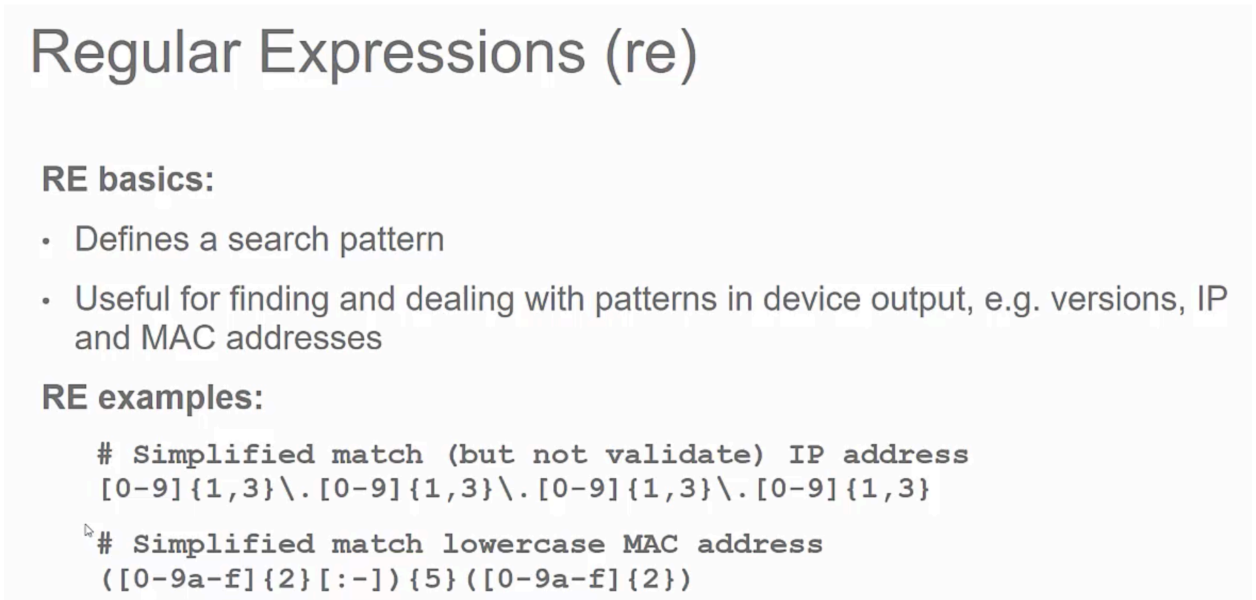


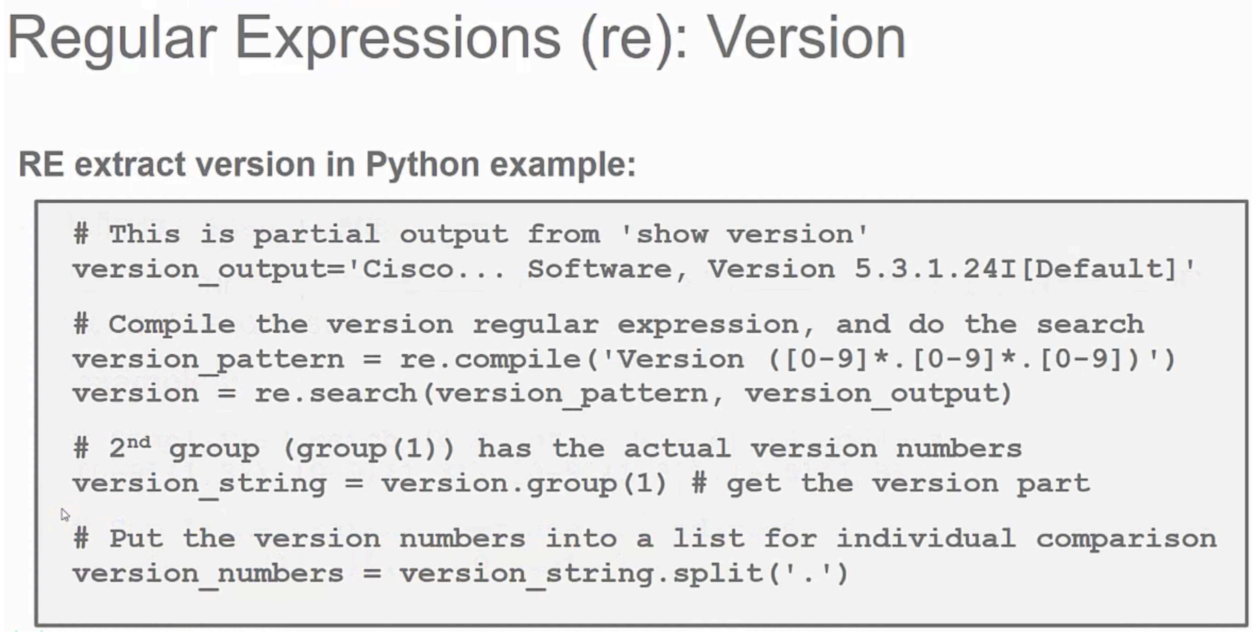
Regular Expressions (re)

RE basics:

- Defines a search pattern
- Useful for finding and dealing with patterns in device output, e.g. versions, IP and MAC addresses

RE examples:

```
# Simplified match (but not validate) IP address
[0-9]{1,3}\.[0-9]{1,3}\.[0-9]{1,3}\.[0-9]{1,3}

# Simplified match lowercase MAC address
([0-9a-f]{2}[:-]){5}([0-9a-f]{2})
```



Regular Expressions (re): Version

RE extract version in Python example:

```
# This is partial output from 'show version'
version_output='Cisco... Software, Version 5.3.1.24I[Default]'

# Compile the version regular expression, and do the search
version_pattern = re.compile('Version ([0-9]*.[0-9]*.[0-9])')
version = re.search(version_pattern, version_output)

# 2nd group (group(1)) has the actual version numbers
version_string = version.group(1) # get the version part

# Put the version numbers into a list for individual comparison
version_numbers = version_string.split('.')
```

In Python, in order to use regular expressions, you will do the following:

- Import `re`, the regular expression library
- Define and compile your regular expression match pattern, which will have wildcards, possible values for matching, number of repetitions, groupings, and so on, as in the examples above for matching IP and MAC addresses.
- Perform a search for your specific string, using the pattern created above
- The result is a regular expression match object holding the actual matched value from the string.

Regular Expression Example

The following code provides an example of using regular expressions to find and extract the software version from the output of a Cisco IOS XR device.
The actual output of the command to show the version, in brief format, is as follows:

```
RP/0/0/CPU0:kcy#show version brief

Thu Dec  3 16:11:02.098 UTC

Cisco IOS XR Software, Version 5.3.1.24I[Default]
Copyright (c) 2015 by Cisco Systems, Inc.

ROM: GRUB, Version 1.99(0), DEV RELEASE

kcy uptime is 1 day, 21 hours, 27 minutes
System image file is "bootflash:disk0/xrvr-os-mbi-5.3.1.24I/mbixrvr-rp.vm"

cisco IOS XRv Series (Pentium Celeron Stepping 3) processor with 3169911K bytes o
Pentium Celeron Stepping 3 processor at 2718MHz, Revision 2.174
IOS XRv Chassis

1 Management Ethernet
7 GigabitEthernet
97070k bytes of non-volatile configuration memory.
866M bytes of hard disk.
2321392k bytes of disk0: (Sector size 512 bytes).
```

You can see the version number on the third line of output:

```
Cisco IOS XR Software, Version 5.3.1.24I[Default]
```

The goal is to extract the relevant parts of that version string, so that you can do some comparisons against a desired version number. The steps to accomplish this task are:

- Create a regular expression pattern that will be used to match the important parts of the version string. In this case, that pattern will match the string 'Version ' followed by three dot-separated numbers.

  ```
  version_pattern = re.compile('Version ([0-9]*\.[0-9]*\.[0-9])')
  ```

  In this case, the version string is relatively straightforward; other devices and variants may require a more involved pattern match string.

- Perform a search for the pattern in the output of the `show version brief` command that was issued. Note that there are two ways to perform the search. You can use the 're' object like in the 'compile' step previously:

```
version = re.search(version_pattern, version_output)
```

You could also use the 'version_pattern' object directly to perform the search:

```
version = version_pattern.search(version_output)
```

Each method yields the same result, a 'version' object that contains specifics about the match that was made. With Python, there are often multiple ways to create your code. Your choice may depend on performance or perhaps even personal preference.

Your regular expression search produces matched text from the input string, which is organized into groups. Roughly speaking, groups are defined by parenthesis. In the example, group 0 is the entire matched string; and group 1 is the part that was matched inside the inner parenthesis:

```
version_string = version.group(1) # get the version part
```

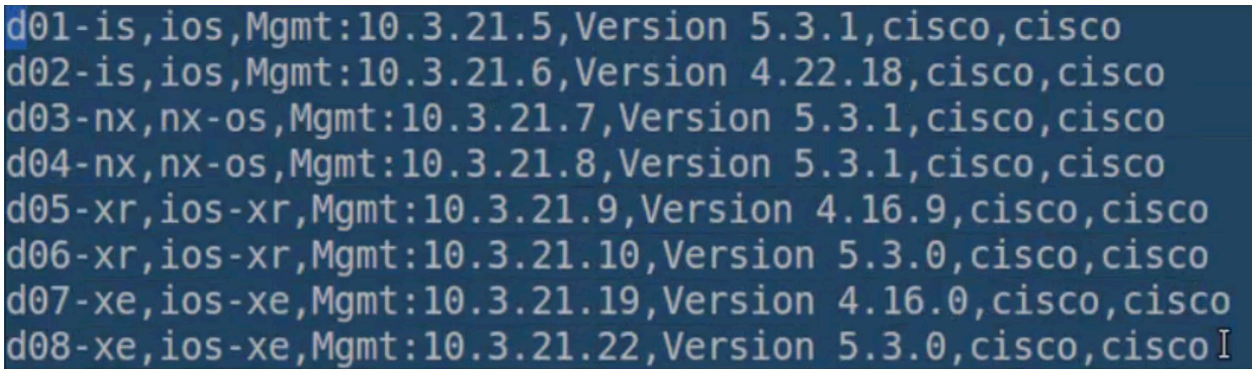
The code sets variable 'version_string' to a subset of the matched string – and that subset is just the purely numerical portion of the matched version string. In this case, that would be the value '5.3.1'.

- The resulting version string '5.3.1' is ready to be decomposed into its version parts, for the major version, minor version, and so on. With string manipulation it is easy to create a list of the different numerical version numbers:

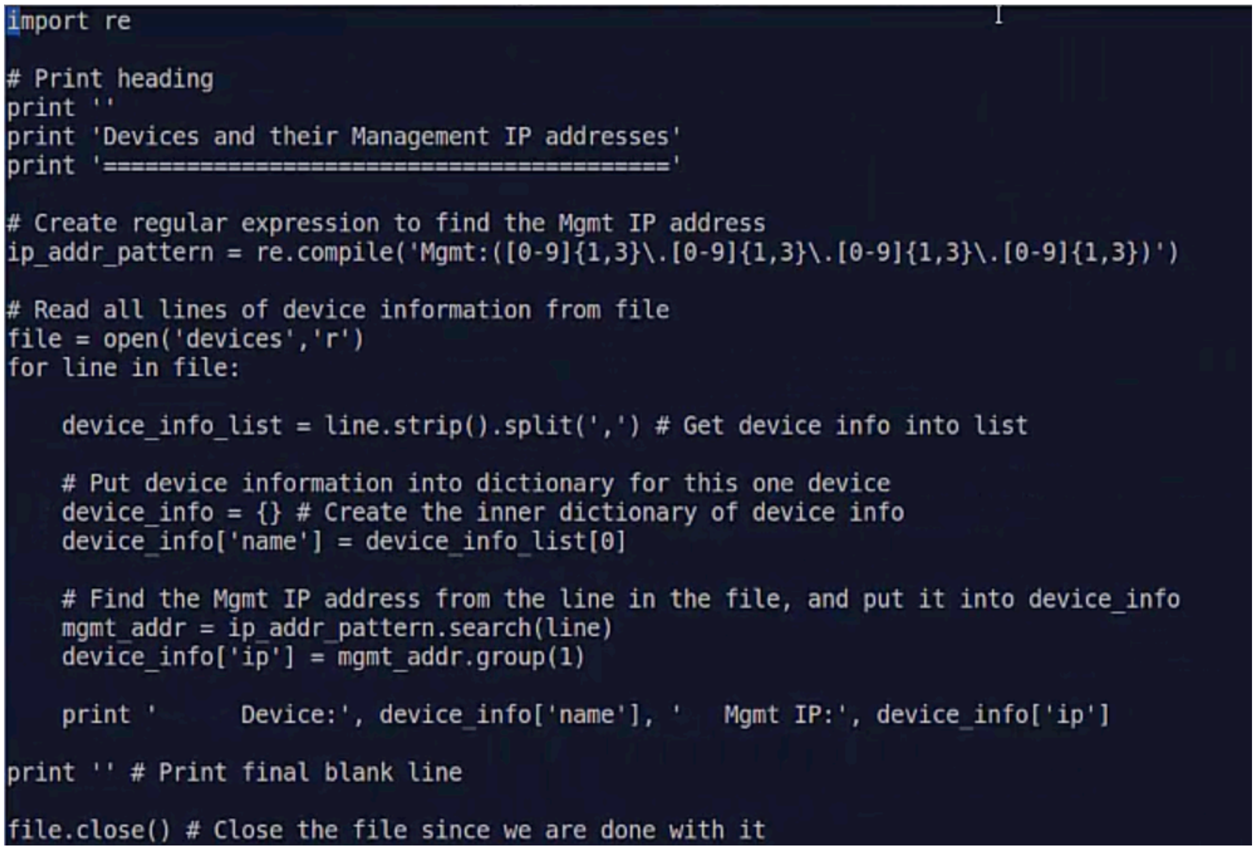
```
version_numbers = version_string.split('.')
```

In this example, you have seen how regular expressions can be useful in taking complex CLI output and extracting the important pieces of information. That information can be used in comparisons, and those comparisons can be used to create code that evaluates data and takes the appropriate action..

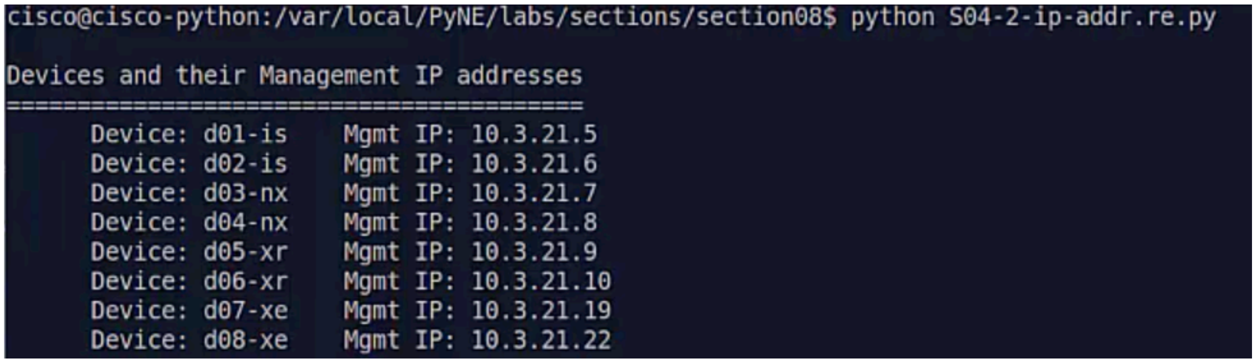
Example – Find Mgmt IP from file



```
d01-is,ios,Mgmt:10.3.21.5,Version 5.3.1,cisco,cisco
d02-is,ios,Mgmt:10.3.21.6,Version 4.22.18,cisco,cisco
d03-nx,nx-os,Mgmt:10.3.21.7,Version 5.3.1,cisco,cisco
d04-nx,nx-os,Mgmt:10.3.21.8,Version 5.3.1,cisco,cisco
d05-xr,ios-xr,Mgmt:10.3.21.9,Version 4.16.9,cisco,cisco
d06-xr,ios-xr,Mgmt:10.3.21.10,Version 5.3.0,cisco,cisco
d07-xe,ios-xe,Mgmt:10.3.21.19,Version 4.16.0,cisco,cisco
d08-xe,ios-xe,Mgmt:10.3.21.22,Version 5.3.0,cisco,cisco
```



```
import re

# Print heading
print ''
print 'Devices and their Management IP addresses'
print '========================================='

# Create regular expression to find the Mgmt IP address
ip_addr_pattern = re.compile('Mgmt:([0-9]{1,3}\.[0-9]{1,3}\.[0-9]{1,3}\.[0-9]{1,3})')

# Read all lines of device information from file
file = open('devices','r')
for line in file:

    device_info_list = line.strip().split(',') # Get device info into list

    # Put device information into dictionary for this one device
    device_info = {} # Create the inner dictionary of device info
    device_info['name'] = device_info_list[0]

    # Find the Mgmt IP address from the line in the file, and put it into device_info
    mgmt_addr = ip_addr_pattern.search(line)
    device_info['ip'] = mgmt_addr.group(1)

    print '      Device:', device_info['name'], '   Mgmt IP:', device_info['ip']

print '' # Print final blank line

file.close() # Close the file since we are done with it
```



```
cisco@cisco-python:/var/local/PyNE/labs/sections/section08$ python S04-2-ip-addr.re.py

Devices and their Management IP addresses
=========================================
      Device: d01-is    Mgmt IP: 10.3.21.5
      Device: d02-is    Mgmt IP: 10.3.21.6
      Device: d03-nx    Mgmt IP: 10.3.21.7
      Device: d04-nx    Mgmt IP: 10.3.21.8
      Device: d05-xr    Mgmt IP: 10.3.21.9
      Device: d06-xr    Mgmt IP: 10.3.21.10
      Device: d07-xe    Mgmt IP: 10.3.21.19
      Device: d08-xe    Mgmt IP: 10.3.21.22
```