1. **Regular Expression Introduction**  
   Video link [https://vimeo.com/832769533](https://t.dripemail2.com/c/eyJhbGciOiJIUzI1NiJ9.eyJhdWQiOiJkZXRvdXIiLCJpc3MiOiJtb25vbGl0aCIsInN1YiI6ImRldG91cl9saW5rIiwiaWF0IjoxNzU4MDIyNjQ3LCJuYmYiOjE3NTgwMjI2NDcsImFjY291bnRfaWQiOiI0MjU0NDk3IiwiZGVsaXZlcnlfaWQiOiJvcXlrNnhraXdmNXVtNjZ4ZXAxNCIsInRva2VuIjoib3F5azZ4a2l3ZjV1bTY2eGVwMTQiLCJzZW5kX2F0IjoxNzU4MDM0ODAwLCJlbWFpbF9pZCI6NzUwNDk2NSwiZW1haWxhYmxlX3R5cGUiOiJDYW1wYWlnbiIsImVtYWlsYWJsZV9pZCI6NDc4NjE0LCJ1cmwiOiJodHRwczovL3ZpbWVvLmNvbS84MzI3Njk1MzM_X19zPW16cG02bnEzY210engwM3RrNWRnIn0.HJlOV7N2wcl7PhGyqKmXXkNqxBFu-NRgU7-4hyZDKmk)  
   Length is 3 minutes
2. **Special Characters**  
   Video link [https://vimeo.com/830294637](https://t.dripemail2.com/c/eyJhbGciOiJIUzI1NiJ9.eyJhdWQiOiJkZXRvdXIiLCJpc3MiOiJtb25vbGl0aCIsInN1YiI6ImRldG91cl9saW5rIiwiaWF0IjoxNzU4MDIyNjQ3LCJuYmYiOjE3NTgwMjI2NDcsImFjY291bnRfaWQiOiI0MjU0NDk3IiwiZGVsaXZlcnlfaWQiOiJvcXlrNnhraXdmNXVtNjZ4ZXAxNCIsInRva2VuIjoib3F5azZ4a2l3ZjV1bTY2eGVwMTQiLCJzZW5kX2F0IjoxNzU4MDM0ODAwLCJlbWFpbF9pZCI6NzUwNDk2NSwiZW1haWxhYmxlX3R5cGUiOiJDYW1wYWlnbiIsImVtYWlsYWJsZV9pZCI6NDc4NjE0LCJ1cmwiOiJodHRwczovL3ZpbWVvLmNvbS84MzAyOTQ2Mzc_X19zPW16cG02bnEzY210engwM3RrNWRnIn0.JtBL_JtqJV4ojxYy2EWmS8gUO_BHjsYuzy6mbr20GIU)  
   Length is 5 minutes
3. **(Somewhat) Simple Examples**  
   Video link [https://vimeo.com/830644262](https://t.dripemail2.com/c/eyJhbGciOiJIUzI1NiJ9.eyJhdWQiOiJkZXRvdXIiLCJpc3MiOiJtb25vbGl0aCIsInN1YiI6ImRldG91cl9saW5rIiwiaWF0IjoxNzU4MDIyNjQ3LCJuYmYiOjE3NTgwMjI2NDcsImFjY291bnRfaWQiOiI0MjU0NDk3IiwiZGVsaXZlcnlfaWQiOiJvcXlrNnhraXdmNXVtNjZ4ZXAxNCIsInRva2VuIjoib3F5azZ4a2l3ZjV1bTY2eGVwMTQiLCJzZW5kX2F0IjoxNzU4MDM0ODAwLCJlbWFpbF9pZCI6NzUwNDk2NSwiZW1haWxhYmxlX3R5cGUiOiJDYW1wYWlnbiIsImVtYWlsYWJsZV9pZCI6NDc4NjE0LCJ1cmwiOiJodHRwczovL3ZpbWVvLmNvbS84MzA2NDQyNjI_X19zPW16cG02bnEzY210engwM3RrNWRnIn0.w1VxLNUn79FDsAJ1o-hGEZ4aP3XpEs5A6SRNhHbfBzU)  
   Length is 7 minutes
4. **Additional Examples**  
   Video link [https://vimeo.com/830428490](https://t.dripemail2.com/c/eyJhbGciOiJIUzI1NiJ9.eyJhdWQiOiJkZXRvdXIiLCJpc3MiOiJtb25vbGl0aCIsInN1YiI6ImRldG91cl9saW5rIiwiaWF0IjoxNzU4MDIyNjQ3LCJuYmYiOjE3NTgwMjI2NDcsImFjY291bnRfaWQiOiI0MjU0NDk3IiwiZGVsaXZlcnlfaWQiOiJvcXlrNnhraXdmNXVtNjZ4ZXAxNCIsInRva2VuIjoib3F5azZ4a2l3ZjV1bTY2eGVwMTQiLCJzZW5kX2F0IjoxNzU4MDM0ODAwLCJlbWFpbF9pZCI6NzUwNDk2NSwiZW1haWxhYmxlX3R5cGUiOiJDYW1wYWlnbiIsImVtYWlsYWJsZV9pZCI6NDc4NjE0LCJ1cmwiOiJodHRwczovL3ZpbWVvLmNvbS84MzA0Mjg0OTA_X19zPW16cG02bnEzY210engwM3RrNWRnIn0.kM2XwgoYBdp4ycYj5vmYCKqkg0oRSGWIYnVGdViWF6A)  
   Length is 10 minutes
5. **Capture Groups and Raw Strings**  
   Video link [https://vimeo.com/830646534](https://t.dripemail2.com/c/eyJhbGciOiJIUzI1NiJ9.eyJhdWQiOiJkZXRvdXIiLCJpc3MiOiJtb25vbGl0aCIsInN1YiI6ImRldG91cl9saW5rIiwiaWF0IjoxNzU4MDIyNjQ3LCJuYmYiOjE3NTgwMjI2NDcsImFjY291bnRfaWQiOiI0MjU0NDk3IiwiZGVsaXZlcnlfaWQiOiJvcXlrNnhraXdmNXVtNjZ4ZXAxNCIsInRva2VuIjoib3F5azZ4a2l3ZjV1bTY2eGVwMTQiLCJzZW5kX2F0IjoxNzU4MDM0ODAwLCJlbWFpbF9pZCI6NzUwNDk2NSwiZW1haWxhYmxlX3R5cGUiOiJDYW1wYWlnbiIsImVtYWlsYWJsZV9pZCI6NDc4NjE0LCJ1cmwiOiJodHRwczovL3ZpbWVvLmNvbS84MzA2NDY1MzQ_X19zPW16cG02bnEzY210engwM3RrNWRnIn0.RTe4rxuyfBr1GWh45MR1wc3UwbutN5_HCGesR3JsSEU)  
   Length is 5 minutes
6. **Anchors and re.MULTILINE**  
   Video link [https://vimeo.com/830685707](https://t.dripemail2.com/c/eyJhbGciOiJIUzI1NiJ9.eyJhdWQiOiJkZXRvdXIiLCJpc3MiOiJtb25vbGl0aCIsInN1YiI6ImRldG91cl9saW5rIiwiaWF0IjoxNzU4MDIyNjQ3LCJuYmYiOjE3NTgwMjI2NDcsImFjY291bnRfaWQiOiI0MjU0NDk3IiwiZGVsaXZlcnlfaWQiOiJvcXlrNnhraXdmNXVtNjZ4ZXAxNCIsInRva2VuIjoib3F5azZ4a2l3ZjV1bTY2eGVwMTQiLCJzZW5kX2F0IjoxNzU4MDM0ODAwLCJlbWFpbF9pZCI6NzUwNDk2NSwiZW1haWxhYmxlX3R5cGUiOiJDYW1wYWlnbiIsImVtYWlsYWJsZV9pZCI6NDc4NjE0LCJ1cmwiOiJodHRwczovL3ZpbWVvLmNvbS84MzA2ODU3MDc_X19zPW16cG02bnEzY210engwM3RrNWRnIn0.C_9XUGBOTxFVF3qAArJyVenpVDZVqPgn2dttw5z4_-M)  
   Length is 3 minutes
7. **re.findall()**  
   Video link [https://vimeo.com/830690686](https://t.dripemail2.com/c/eyJhbGciOiJIUzI1NiJ9.eyJhdWQiOiJkZXRvdXIiLCJpc3MiOiJtb25vbGl0aCIsInN1YiI6ImRldG91cl9saW5rIiwiaWF0IjoxNzU4MDIyNjQ3LCJuYmYiOjE3NTgwMjI2NDcsImFjY291bnRfaWQiOiI0MjU0NDk3IiwiZGVsaXZlcnlfaWQiOiJvcXlrNnhraXdmNXVtNjZ4ZXAxNCIsInRva2VuIjoib3F5azZ4a2l3ZjV1bTY2eGVwMTQiLCJzZW5kX2F0IjoxNzU4MDM0ODAwLCJlbWFpbF9pZCI6NzUwNDk2NSwiZW1haWxhYmxlX3R5cGUiOiJDYW1wYWlnbiIsImVtYWlsYWJsZV9pZCI6NDc4NjE0LCJ1cmwiOiJodHRwczovL3ZpbWVvLmNvbS84MzA2OTA2ODY_X19zPW16cG02bnEzY210engwM3RrNWRnIn0.kcknlIaVti6SpUs54JAYydI2f9hJuejdTBFEobqIIWM)  
   Length is 5 minutes
8. **Match logic**  
   Video link [https://vimeo.com/832457474](https://t.dripemail2.com/c/eyJhbGciOiJIUzI1NiJ9.eyJhdWQiOiJkZXRvdXIiLCJpc3MiOiJtb25vbGl0aCIsInN1YiI6ImRldG91cl9saW5rIiwiaWF0IjoxNzU4MDIyNjQ3LCJuYmYiOjE3NTgwMjI2NDcsImFjY291bnRfaWQiOiI0MjU0NDk3IiwiZGVsaXZlcnlfaWQiOiJvcXlrNnhraXdmNXVtNjZ4ZXAxNCIsInRva2VuIjoib3F5azZ4a2l3ZjV1bTY2eGVwMTQiLCJzZW5kX2F0IjoxNzU4MDM0ODAwLCJlbWFpbF9pZCI6NzUwNDk2NSwiZW1haWxhYmxlX3R5cGUiOiJDYW1wYWlnbiIsImVtYWlsYWJsZV9pZCI6NDc4NjE0LCJ1cmwiOiJodHRwczovL3ZpbWVvLmNvbS84MzI0NTc0NzQ_X19zPW16cG02bnEzY210engwM3RrNWRnIn0.JL3EEUbb0861QZ62qc3nN9KFRV5h1funssw47p8ChCE)  
   Length is 1 minute
9. **re.DOTALL**  
   Video link [https://vimeo.com/832460190](https://t.dripemail2.com/c/eyJhbGciOiJIUzI1NiJ9.eyJhdWQiOiJkZXRvdXIiLCJpc3MiOiJtb25vbGl0aCIsInN1YiI6ImRldG91cl9saW5rIiwiaWF0IjoxNzU4MDIyNjQ3LCJuYmYiOjE3NTgwMjI2NDcsImFjY291bnRfaWQiOiI0MjU0NDk3IiwiZGVsaXZlcnlfaWQiOiJvcXlrNnhraXdmNXVtNjZ4ZXAxNCIsInRva2VuIjoib3F5azZ4a2l3ZjV1bTY2eGVwMTQiLCJzZW5kX2F0IjoxNzU4MDM0ODAwLCJlbWFpbF9pZCI6NzUwNDk2NSwiZW1haWxhYmxlX3R5cGUiOiJDYW1wYWlnbiIsImVtYWlsYWJsZV9pZCI6NDc4NjE0LCJ1cmwiOiJodHRwczovL3ZpbWVvLmNvbS84MzI0NjAxOTA_X19zPW16cG02bnEzY210engwM3RrNWRnIn0.00i1MhQT0f8FAAvptkCcxbcjM-jJbBmwbcX6Q0FVaO4)  
   Length is 2 minutes
10. **re.escape()**  
    Video link  [https://vimeo.com/832488661](https://t.dripemail2.com/c/eyJhbGciOiJIUzI1NiJ9.eyJhdWQiOiJkZXRvdXIiLCJpc3MiOiJtb25vbGl0aCIsInN1YiI6ImRldG91cl9saW5rIiwiaWF0IjoxNzU4MDIyNjQ3LCJuYmYiOjE3NTgwMjI2NDcsImFjY291bnRfaWQiOiI0MjU0NDk3IiwiZGVsaXZlcnlfaWQiOiJvcXlrNnhraXdmNXVtNjZ4ZXAxNCIsInRva2VuIjoib3F5azZ4a2l3ZjV1bTY2eGVwMTQiLCJzZW5kX2F0IjoxNzU4MDM0ODAwLCJlbWFpbF9pZCI6NzUwNDk2NSwiZW1haWxhYmxlX3R5cGUiOiJDYW1wYWlnbiIsImVtYWlsYWJsZV9pZCI6NDc4NjE0LCJ1cmwiOiJodHRwczovL3ZpbWVvLmNvbS84MzI0ODg2NjE_X19zPW16cG02bnEzY210engwM3RrNWRnIn0.p1poQRRuW1pbV81n85ausPFHypA3Xe6Kh9OspEtecfU)  
    Length is 3 minutes
11. **Named Capture Groups**(\*optional/intermediate content)  
    Video link [https://vimeo.com/832490540](https://t.dripemail2.com/c/eyJhbGciOiJIUzI1NiJ9.eyJhdWQiOiJkZXRvdXIiLCJpc3MiOiJtb25vbGl0aCIsInN1YiI6ImRldG91cl9saW5rIiwiaWF0IjoxNzU4MDIyNjQ3LCJuYmYiOjE3NTgwMjI2NDcsImFjY291bnRfaWQiOiI0MjU0NDk3IiwiZGVsaXZlcnlfaWQiOiJvcXlrNnhraXdmNXVtNjZ4ZXAxNCIsInRva2VuIjoib3F5azZ4a2l3ZjV1bTY2eGVwMTQiLCJzZW5kX2F0IjoxNzU4MDM0ODAwLCJlbWFpbF9pZCI6NzUwNDk2NSwiZW1haWxhYmxlX3R5cGUiOiJDYW1wYWlnbiIsImVtYWlsYWJsZV9pZCI6NDc4NjE0LCJ1cmwiOiJodHRwczovL3ZpbWVvLmNvbS84MzI0OTA1NDA_X19zPW16cG02bnEzY210engwM3RrNWRnIn0.apQrozy3AntTeAMKkq9uczfyWyIkZk0dB3BLa8SV0OI)  
    Length is 2 minutes
12. [**regex101.com**](http://regex101.com/)  
    Video link  [https://vimeo.com/832496547](https://t.dripemail2.com/c/eyJhbGciOiJIUzI1NiJ9.eyJhdWQiOiJkZXRvdXIiLCJpc3MiOiJtb25vbGl0aCIsInN1YiI6ImRldG91cl9saW5rIiwiaWF0IjoxNzU4MDIyNjQ3LCJuYmYiOjE3NTgwMjI2NDcsImFjY291bnRfaWQiOiI0MjU0NDk3IiwiZGVsaXZlcnlfaWQiOiJvcXlrNnhraXdmNXVtNjZ4ZXAxNCIsInRva2VuIjoib3F5azZ4a2l3ZjV1bTY2eGVwMTQiLCJzZW5kX2F0IjoxNzU4MDM0ODAwLCJlbWFpbF9pZCI6NzUwNDk2NSwiZW1haWxhYmxlX3R5cGUiOiJDYW1wYWlnbiIsImVtYWlsYWJsZV9pZCI6NDc4NjE0LCJ1cmwiOiJodHRwczovL3ZpbWVvLmNvbS84MzI0OTY1NDc_X19zPW16cG02bnEzY210engwM3RrNWRnIn0.TFZsAipy03_MtqzbTPpPXL3p-yQVpK8vyqHDo3AkRIA)  
    Length is 5 minutes

**\*Optional/intermediate content - this content is more intermediate and can be safely skipped.** In other words, feel free to skip this content if you are time limited or are struggling with the more fundamental content.  
  
  
  
  
**Collateral Material (programs used in the videos):**  
  
*In collateral, we try to provide the Python scripts and related files shown in the videos. In other words, we try to make it easy for you to reproduce the examples from the videos.*  
  
[Lesson5 Collateral Program](https://t.dripemail2.com/c/eyJhbGciOiJIUzI1NiJ9..w3FEmz8oWL9esnEty0vjM03ACi0MNc5RPz4eIrL6fIk)  
  
  
  
  
**Additional Content:**  
  
[Regular Expression HOWTO](https://t.dripemail2.com/c/eyJhbGciOiJIUzI1NiJ9.eyJhdWQiOiJkZXRvdXIiLCJpc3MiOiJtb25vbGl0aCIsInN1YiI6ImRldG91cl9saW5rIiwiaWF0IjoxNzU4MDIyNjQ3LCJuYmYiOjE3NTgwMjI2NDcsImFjY291bnRfaWQiOiI0MjU0NDk3IiwiZGVsaXZlcnlfaWQiOiJvcXlrNnhraXdmNXVtNjZ4ZXAxNCIsInRva2VuIjoib3F5azZ4a2l3ZjV1bTY2eGVwMTQiLCJzZW5kX2F0IjoxNzU4MDM0ODAwLCJlbWFpbF9pZCI6NzUwNDk2NSwiZW1haWxhYmxlX3R5cGUiOiJDYW1wYWlnbiIsImVtYWlsYWJsZV9pZCI6NDc4NjE0LCJ1cmwiOiJodHRwczovL2RvY3MucHl0aG9uLm9yZy8zL2hvd3RvL3JlZ2V4Lmh0bWw_X19zPW16cG02bnEzY210engwM3RrNWRnIn0.oQfy3dRuVeOZ6uTLKSy9OeGawOrmOIPaYxyiLJv3DAY)  
Python.org documentation on regular expressions. This document is very good in covering the fundamental details of Python regular expressions. I recommend reading through the section named, "Non-capturing and Named Groups".  
  
[Google for Education: Python Regular Expressions](https://t.dripemail2.com/c/eyJhbGciOiJIUzI1NiJ9.eyJhdWQiOiJkZXRvdXIiLCJpc3MiOiJtb25vbGl0aCIsInN1YiI6ImRldG91cl9saW5rIiwiaWF0IjoxNzU4MDIyNjQ3LCJuYmYiOjE3NTgwMjI2NDcsImFjY291bnRfaWQiOiI0MjU0NDk3IiwiZGVsaXZlcnlfaWQiOiJvcXlrNnhraXdmNXVtNjZ4ZXAxNCIsInRva2VuIjoib3F5azZ4a2l3ZjV1bTY2eGVwMTQiLCJzZW5kX2F0IjoxNzU4MDM0ODAwLCJlbWFpbF9pZCI6NzUwNDk2NSwiZW1haWxhYmxlX3R5cGUiOiJDYW1wYWlnbiIsImVtYWlsYWJsZV9pZCI6NDc4NjE0LCJ1cmwiOiJodHRwczovL2RldmVsb3BlcnMuZ29vZ2xlLmNvbS9lZHUvcHl0aG9uL3JlZ3VsYXItZXhwcmVzc2lvbnM_X19zPW16cG02bnEzY210engwM3RrNWRnIn0.SZ8Ztr69STQ1yDMIPO6br-jiqhFqepNZwN7NRhxMz_c)  
Good, practical content on how to use regular expressions in Python. Makes the important distinction that the leftmost match will win.  
  
  
  
  
**Exercises**  
  
Reference code for these exercises is posted on GitHub at:  
    [https://github.com/twin-bridges/learning\_python/tree/main/lesson5/exercises](https://t.dripemail2.com/c/eyJhbGciOiJIUzI1NiJ9.eyJhdWQiOiJkZXRvdXIiLCJpc3MiOiJtb25vbGl0aCIsInN1YiI6ImRldG91cl9saW5rIiwiaWF0IjoxNzU4MDIyNjQ3LCJuYmYiOjE3NTgwMjI2NDcsImFjY291bnRfaWQiOiI0MjU0NDk3IiwiZGVsaXZlcnlfaWQiOiJvcXlrNnhraXdmNXVtNjZ4ZXAxNCIsInRva2VuIjoib3F5azZ4a2l3ZjV1bTY2eGVwMTQiLCJzZW5kX2F0IjoxNzU4MDM0ODAwLCJlbWFpbF9pZCI6NzUwNDk2NSwiZW1haWxhYmxlX3R5cGUiOiJDYW1wYWlnbiIsImVtYWlsYWJsZV9pZCI6NDc4NjE0LCJ1cmwiOiJodHRwczovL2dpdGh1Yi5jb20vdHdpbi1icmlkZ2VzL2xlYXJuaW5nX3B5dGhvbi90cmVlL21haW4vbGVzc29uNS9leGVyY2lzZXM_X19zPW16cG02bnEzY210engwM3RrNWRnIn0.AHHuEftqGd-sc9Ojv6M1h3d_-tBodlYQLbLE6sFCI8c)  
  
  
1. Using this [show\_version.txt](https://t.dripemail2.com/c/eyJhbGciOiJIUzI1NiJ9..KhldpPlleQ1iSxKQlqWaqpjVQL-Y0VRtLydp_3xJW3g) file from a Cisco IOS-XE router, extract both the serial number and the model number. You should use regular expressions to accomplish this. These items are contained on the following lines (both the model number and serial number are underlined).

cisco C1111-4P (1RU) processor with 1401823K/6147K bytes of memory.

Processor board ID FGL222290LB

2. Using this [show\_ip\_bgp\_neighbors.txt](https://t.dripemail2.com/c/eyJhbGciOiJIUzI1NiJ9..HInNZD6LRPQXHbWCiYw_AeqNKX5KuVvaq5GaFy49vow) file (from a Cisco IOS router), parse the BGP information and extract the following items: bgp\_neighbor\_ip, remote\_as, bgp\_version, remote\_router\_id, and bgp\_state. This information is in the first three lines of the BGP neighbor output:

​BGP neighbor is 10.220.88.38, remote AS 44, external link

BGP version 4, remote router ID 10.220.88.38

BGP state = Established, up for 11w0d

Use regular expressions and optionally named capture groups to accomplish this.  
  
Print these five fields out to standard output.  
  
  
3. Using this [arista\_show\_ip\_arp.txt](https://t.dripemail2.com/c/eyJhbGciOiJIUzI1NiJ9..J_Yry_fkPsPxnk_zMzi-gbEMSosFKQAK9OkEqONznWQ) file from an arista vEOS switch, parse the ARP table and extract both the IP address and the corresponding MAC address.  
  
You should use re.findall() to accomplish this. You should create a dictionary as your final data structure with the IP address as the key and the MAC address as the corresponding value.  
  
Note that you can directly cast a list of two-value tuples into a dictionary: the first element in the tuple will become the key, the second element will become the corresponding value:

In [1]: my\_list = [

...: ("key1", "value1"),

...: ("key2", "value2")

...: ]

In [2]: dict(my\_list)

Out[2]: {'key1': 'value1', 'key2': 'value2'}

4. Inside this [show\_lldp.txt](https://t.dripemail2.com/c/eyJhbGciOiJIUzI1NiJ9..ERPaApj06EXe9vyWA6KkxTn9ybdF_BxcD4XbYmaNp_U) file, there are four LLDP entries (corresponding to "show lldp neighbors detail" output from an NX-OS switch). Each of these four entries starts with a "Chassis id:" line and ends with a "Vlan ID:" line.  
  
Use regular expressions to extract the four sections and to save these sections into a list.  
  
Note, this problem is challenging and you might need to use re.DOTALL to accomplish it.  
  
Once you have saved the four separate LLDP entries into a list, you should then use re.search() to parse the remote\_system\_name, remote\_port and the local\_port. These entries are contained in the "System Name", "Port id" and "Local Port id:" lines, respectively.  
  
Print out  the local\_port and then the remote\_system\_name and remote\_port that is reachable via that local\_port.  
  
  
5. Using this [aruba\_cx\_show\_ipv6\_intf.txt](https://t.dripemail2.com/c/eyJhbGciOiJIUzI1NiJ9..mHVQ6V3itnDD4AJTNzH7r2OsSAcGuhB-lgRXHfWnTIg) file, extract the intf\_name, the intf\_state, the admin\_state, and the ipv6\_addr using regular expressions.  
  
Print these values to standard output. These values show up in the first four lines of the output and are underlined below:

Interface 1/1/1 is up

Admin state is up

IPv6 address:

2001:0db8:85a3:0000:0000:8a2e:0370:7334/24 [VALID]​

**CLASS OUTLINE**

1. **Regular Expression Introduction**
   1. The Simple Case (literal characters) [0:55]
2. **RegEx Special Characters**
   1. Regular Expressions: Simple Characters [0:10]
      1. By default, regular expressions will be greedy. So .\* will match as many characters as it can. [1:10]
   2. Anchors - by default anchors are the beginning or end of the entire string. [3:00]
3. **(Somewhat) Simple Examples**
   1. Import re library [0:30]
   2. re.search .search() - The main method we will use regular expressions to “search” for patterns in some string [re.search(pattern, string, flag=0) [0:43]
      1. Example ‘re.search(“.”, line)’ [1:18]
      2. Example ‘re.search(“..”, line)’ [1:23]
      3. Example ‘m=re.search(“.\*”, line)’ star is greedy  [2:01]
      4. Example ‘m=re.search(“.\*?”, line)’ match as few characters as possible [2:30]
      5. Example ‘m=re.search(“.+?”, line)’ zero or more characters [3:01]
      6. Example ‘m=re.search(“Cisco\s.\*Fuji”, line)’ [3:49]
      7. Example ‘m=re.search(“Cisco\s+Fuji”, alt\_line)’ [5:14]
      8. Example ‘m=re.search(“\S+\s+Fuji”, alt\_line)’ [6:09]
      9. Example ‘m=re.search(“\w+\s+Fuji”, alt\_line)’ [6:33]
4. **Additional Examples**
   1. Example ‘m=re.search(“^Cisco.\*\d+”, line)’ caret anchor and digits [0:15]
   2. Example ‘m=re.search(“^Cisco.\*\d\d”, line)’ caret anchor and two digits [2:13]
   3. Example m=‘re.search(“^Cisco.\*\d\d\.\d\.\d”, line)’ two digits, literal period, one digit, literal period, and one digit [2:48]
   4. Example ‘m=re.search(“^Cisco.\*(\d\d\.\d\.\d)”, line)’ parentheses to capture things [4:17]
   5. Example ‘m=re.search(“^Cisco.\*(\d\d\.\d\.\d).\*$”, line)’ end line achor [4:42]
   6. Example ‘m=re.search(“^Cisco.\*[\d\.]”, line)’ constructing your own character class with digits and periods [6:11]
   7. Example ‘re.search(“^Cisco.\*[\d\.]”, line)’ constructing your own character class with digits and periods [6:11]
   8. Example ‘m=re.search(“^Cisco.\*[\d\.]+”, line)’ constructing your own character class with digits and periods, with repeat [6:42]
   9. Example ‘m=re.search(“^Cisco.\*[\d\.]+, REL”, line)’ not saved/error / no parentheses [7:14]
   10. Example ‘m=re.search(“^Cisco.\*([\d\.]+), REL”, line)’ with parentheses [8:25]
   11. Example ‘m=re.search(“^Cisco.\*ion ([\d\.]+), REL”, line)’ desired result of version number [9:27]
5. **Capture Groups and Raw Strings**
   1. Special Character Examples [0:15]
      1. Example m = ‘re.search(r“Configuration register is (.\*)”, data) [1:00]
      2. Raw string [1:26]
         1. In general, we want the general expression special characters and not the Python special characters. “r” turns off Python special characters meaning [2:21]
      3. Example: m = re.search(“Versions (.\*)”, line), captures all characters [3:02]
      4. Example: m = re.search(“Versions (.\*),”, line) Add the comma to stop the capturing [3:23]
      5. Example: m = re.search(“Versions (.\*?),”, line) Question mark makes the wild card be non-greedy [4:22]
6. **Anchors and re.MULTILINE**
   1. Using Anchors “^” beginning and “$” by default will match the entire string [0:27]
   2. In Python, we can change to line by line behavior with flags [1:55]
      1. Example: m = re.search(“^Config.\*2102$”, data, flags=re.MULTILINE) - line by line basis [2:10]
      2. “re.MULTILINE” can also abbreviated as “re.M” [2:53]
7. **re.findall()**
   1. Extracting patterns that repeat [0:17]
   2. Example: pattern = “Internet\s+(\d+\.\d+\.\d+\.\d+)\s+” [0:53]
   3. Example: re.findall(pattern, data) - Find all occurrences of pattern in data and return a list [2:09]
   4. Example: pattern = “Internet\s+(\d+\.\d+\.\d+\.\d+)\s+[-\d]+\s+(\w+\.\w+\.\w+)\s+” Expanding on pattern, IP and MAC Address [2:27]
      1. Returns a list of tuples [4:18]
   5. Making things more readable with f-strings [4:45]
8. **Match logic**
   1. You can use the Match object to differentiate between successful and failed searches [0:17]
   2. The Match object will be None on a failed search [0:53]
9. **re.DOTALL**
   1. By default the dot “.” special character does not include newlines [0:25]
   2. We can change the behavior to include newlines by adding the “flags=re.DOTALL” (or abbreviated “re.D”) argument [1:15]
10. **Re.escape**
    1. What if we have characters from a networking device or from user-input and we want to search on those [0:19]
    2. Many times we just want the literal pattern with no special regular expressions meaning [2:10]
    3. re.escape will automatically escape any special characters [2:24]
11. **Named Capture Groups**
    1. We can name our capture groups
       1. Example: (?P<confred>\S+) [0:42]
       2. m.groupdict() can return a dictionary of the named capture group [1:58]
       3. m.group() can call .group() method and provide the group name [2:05]
12. **Regex101.com**
    1. Using [regex101.com](http://regex101.com/) - login and select Python on left navigation [0:23]
    2. Window: Regular Expression and Test String [0:55]
    3. Quick Reference (bottom right) [1:00]
    4. Example: copied and pasted a bit of show version into Test String window [1:59]
    5. Set “Regex Flags” using drop down in Regular Expression window (can set multiline or single line) [2:28] Explanation window (upper right) [3:45]