Gmail

Константин Городецкий <gorodetskiykp@gmail.com>

[PyNet Learning Python 2018] - Lesson2 / Numbers, Files, Lists, and Linters

1 письмо

ktbyers@twb-tech.com <ktbyers@twb-tech.com> 9 февраля 2018 г., 2:22
Кому: gorodetskiykp@gmail.com

Konstantin

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. **Numbers**
   Video **https://vimeo.com/244128549**
   Length is 9 minutes

2. **Files**
   Video **https://vimeo.com/244127459**
   Length is 10 minute

3. **Lists**
   Video **https://vimeo.com/244257596**
   Length is 6 minutes

4. **List Slices**
   Video **https://vimeo.com/244259492**
   Length is 4 minutes

5. **Lists are Mutable**
   Video **https://vimeo.com/244287000**
   Length is 5 minutes

6. **Tuples**
   Video **https://vimeo.com/244153105**
   Length is 3 minutes

7. **Using .join()**
   Video **https://vimeo.com/245464488**
   Length is 3 minutes

8. **sys.argv**
   Video **https://vimeo.com/245464766**
   Length is 2 minutes

9. **Linters**
   Video **https://vimeo.com/245102246**
   Length is 6 minutes

Additional Content:

Automate the Boring Stuff with Python (Chapter 4 on Strings)
Up through the section named "Removing Values from Lists with del Statements"

Dive Into Python, Introducing Lists

Exercises
Reference code for these exercises is posted on GitHub at:
https://github.com/ktbyers/pynet/tree/master/learning_python/lesson2

1. Open the "show_version.txt" file for reading. Use the .read() method to read in the entire file contents to a variable. Print out the file contents to the screen. Also print out the type of the variable (you should have a string at this point).

Close the file.

Open the file a second time using a Python context manager (with statement). Read in the file contents using the .readlines() method. Print out the file contents to the screen. Also print out the type of the variable (you should have a list at this point).

2. Create a list of five IP addresses.

Use the .append() method to add an IP address onto the end of the list. Use the .extend() method to add two more IP addresses to the end of the list.

Use list concatenation to add two more IP addresses to the end of the list.

Print out the entire list of ip addresses. Print out the first IP address in the list. Print out the last IP address in the list.

Using the .pop() method to remove the first IP address in the list and the last IP address in the list.

Update the new first IP address in the list to be '2.2.2.2'. Print out the new first IP address in the list.

3. Read in the "show_arp.txt" file using the readlines() method. Use a list slice to remove the header line.

Use pretty print to print out the resulting list to the screen, syntax is:

```
from pprint import pprint
pprint(some_var)
```

Use the list .sort() method to sort the list based on IP addresses.

Create a new list slice that is only the first three ARP entries.

Use the .join() method to join these first three ARP entries back together as a single string using the newline character ('\n') as the separator.

Write this string containing the three ARP entries out to a file named "arp_entries.txt".

4. Read in the "show_ip_int_brief.txt" file into your program using the .readlines() method.

Obtain the list entry associated with the FastEthernet4 interface. You can just hard-code the index at this point since we haven't covered for-loops or regular expressions. Use the string .split() method to obtain both the IP address and the corresponding MAC address associated with the IP.

Create a two element tuple from the result (intf_name, ip_address).

Print that tuple to the screen.

Use pycodestyle on this script. Get the warnings/errors to zero. You might need to 'pip install pycodestyle' on your computer (you should be able to type this from the shell prompt). Alternatively, you can type 'python -m pip install pycodestyle'.

5. Read the 'show_ip_bgp_summ.txt' file into your program. From this BGP output obtain the first and last lines of the output.

From the first line use the string .split() method to obtain the local AS number.

From the last line use the string .split() method to obtain the BGP peer IP address.

Print both local AS number and the BGP peer IP address to the screen.

CLASS OUTLINE

1. Numbers (VIDEO1)
   A. Creating an integer type [0:10]
   B. Arithmetic Operators [0:23]
      I. Modulo Operator [0:59]
      II. Raise to the power [1:19]
      III. Division behavior and Python2 [1:31]
         a. from __future__ import division [2:44]
      IV. Whitespace in Python [3:32]
   C. Type float [4:00]
   D. Type casting an integer to a float [4:50]
   E. Strange math behavior of .1 + .2 [5:53]
   F. Increment by one [8:00]

2. Files (VIDEO2)
   A. Open a file for reading [0:26]
      I. f.read() [0:58]
      II. f.seek(0) [1:43]
      III. f.close() [2:22]
   B. f.readline() [3:23]
   C. f.readlines() [3:56]
   D. Context manager [5:04]
      I. Automatic close [5:56]
   E. Writing a file [6:30]
      I. f.flush() [7:30]
      II. Destructive operation [7:43]
   F. Append a file [8:12]

3. Lists (VIDEO3)
   A. Data that is inherently sequential [0:06]
   B. Examples [0:19]
   C. Creating a list [0:56]
      I. Creating a list with existing values [1:16]
      II. Lists can store different types of data [1:39]
      III. Accessing element 0, 1, 2, 3 [1:56]
      IV. Changing the value of an element [2:23]
   D. List methods [2:53]
      I. append() [2:57]
      II. list concatenation [3:13]
      III. list.extend() [3:48]

```
      IV. removing elements [4:14]
         a. my_list.pop() [4:22]
         b. my_list.pop(0) [4:37]
         c. del my_list[0] [4:58]

4. List slices (VIDEO4)
   A. Read from show_version.txt [0:12]
   B. Creating new lists from an existing list [0:47]
   C. First index is included [1:07]
   D. Last index is excluded [1:14]
   E. First number is excluded (go to beginning) [1:39]
   F. Last number is excluded (go to end) [2:02]
   G. Can use negative indices [3:21]

5. Lists are Mutable (VIDEO5)
   A. What does mutable mean [0:14]
   B. Mutable data types (lists, dictionaries) [2:13]
   C. Immutable data types (strings, numbers) [3:03]
   D. Be careful with copying lists [4:48]

6. Tuples (VIDEO6)
   A. Use parenthesis and not brackets [0:19]
   B. Why use tuples? [0:34]
   C. Behaves like a list, but can't modify it [0:48]
   D. Tuples use less memory [2:07]
   E. Type cast as a list [2:21]

7. Using .join() (VIDEO7)
   A. Review of using .split()   [0:11]
      I. Split returns a list   [0:34]
   B. .join() reunites the parts   [0:59]
      I. String separator, pass in a list   [1:06]
      II. Separator between each element   [1:30]
      III. .join() is a string method   [1:35]
   C. Example using an IP address   [2:26]

8. sys.argv (VIDEO8)
   A. Behavior of sys.argv   [0:25]
      I. Returns a list, element1 is the program name   [0:31]
      II. Additional args are remaining elements   [0:54]

9. Linters (VIDEO9)
   A. Advantages of using linters [0:04]
      I. Improve readability [0:23]
      II. Flag obvious errors and save debugging time [0:40]
      III. Flag potential problems [0:50]
   B. Two most common linters: pylint and pycodestyle [1:01]
      I. Installing pylint and pycodestyle linters [1:20]
      II. Example using pylint and pycodestyle [1:35]
```

```
   III. Example using pylint and pycodestyle [2:08]
C. Using pylama and a setup.cfg file [3:35]
D. You decide what characteristics matters [4:07]
E. Zero errors [5:03]
F. Integrate with CI [5:16]
```

Kirk Byers
