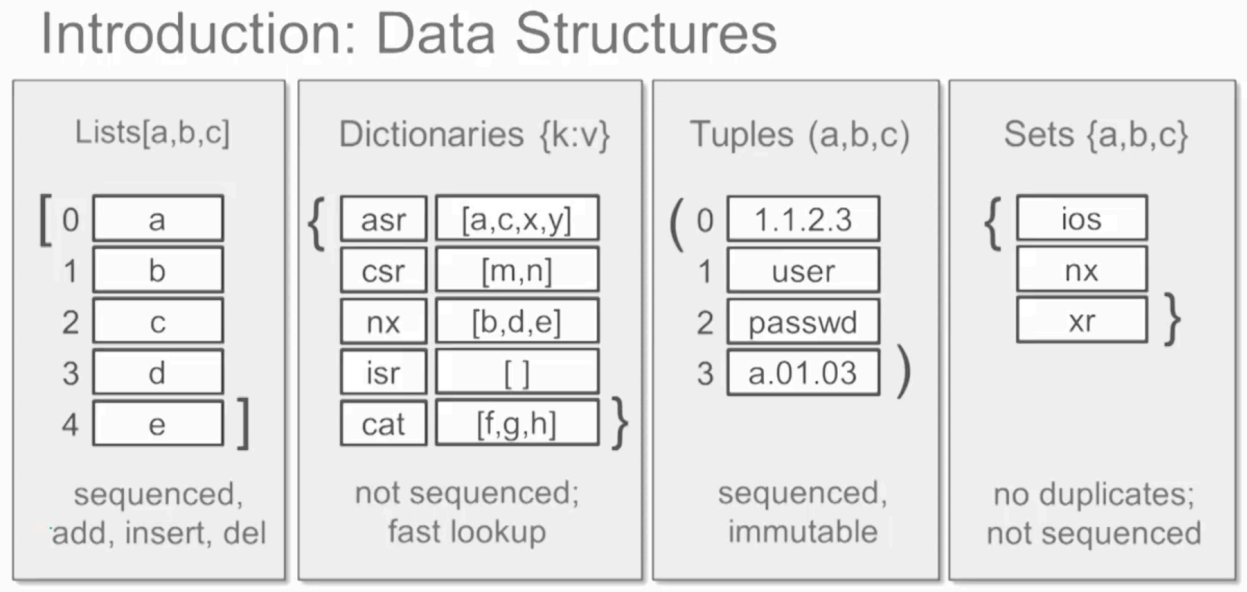
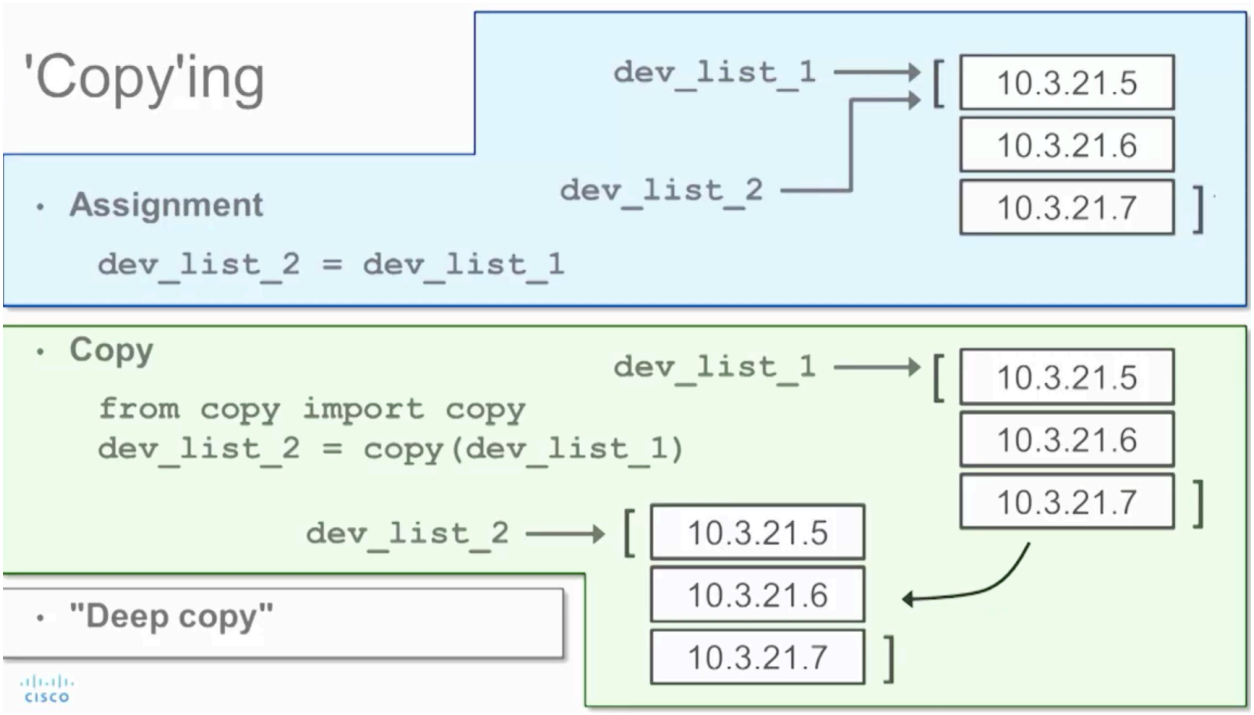


** In other programming languages dictionaries are referred to as maps or hash tables.
Lists in python begin with []
Dictionary in python begin with {}
Tuples in python begin with ()
Sets in python begin with {}

Data Structures Overview

- **Lists** and **Dictionaries** are used most often
- **Tuples** for never-changing lists of items, as keys to dictionary items
- **Sets** for testing *inclusion*, doing *unions* and *intersections*
- **Complex** data structures can be created by having lists of tuples, dictionaries of lists, lists of dictionaries of lists, etc.
- **Comprehensions** offer a quicker way to create lists and dictionaries
- **Copy** operations require understanding of "everything is an object"



Assigning an object to another is a form of copy
Using the copy method will create an actual copy of the object, however is shallow which can be problematic with sophisticated data structures.
Using a deep copy will allow a robust copy of advanced data structures.

There are 3 common steps with most scripts you create:

- Read in a set of information, such as a list of MAC addresses in a switch forwarding table.
- Analyze the information, in some way such as calculating the number of MAC addresses that are reachable via each interface.
- Output the information, such as printing a list of interfaces and the count of MAC address to the screen for the user to see.

In between these steps, you will need to store information temporarily until you are ready to output the results.

For example, to build a script that counts the MAC addresses associated with each interface, you need to read in the MAC address table from a switch:

```
Vlan    Mac Address       Type        Ports
----    -----------       --------    -----
  11    fa16.3e14.a0f5    DYNAMIC     Gi0/2
  11    fe00.6abe.ff01    DYNAMIC     Gi0/2
  14    0050.56be.2548    DYNAMIC     Gi0/1
  14    0410.50f2.3f44    DYNAMIC     Gi0/1
  14    0072.7f33.debe    DYNAMIC     Gi0/1
```

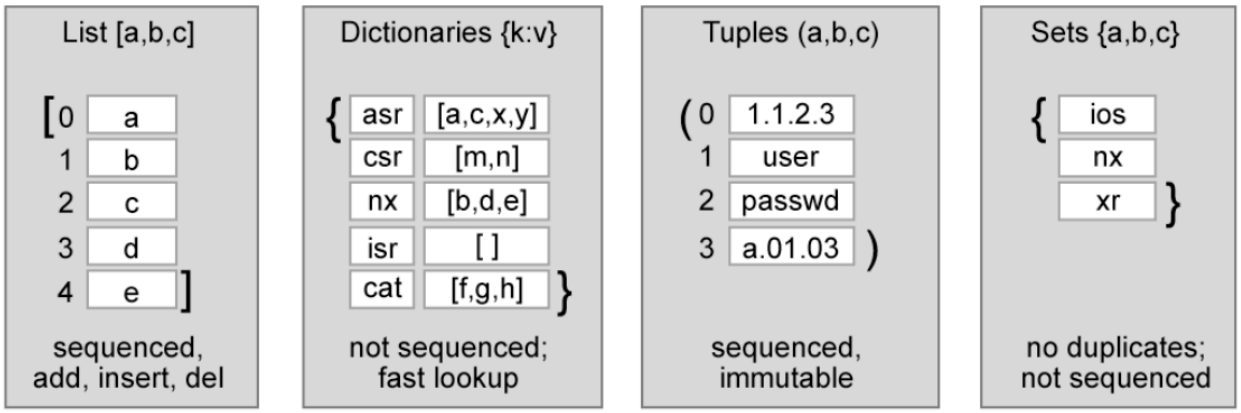
As your script reads in the lines of data, it will store the interface names that are seen in the MAC address table and a count of the number of MAC addresses associated with each interface. The information that it stores would look like similar to:

```
Interface       Count of MAC Addresses
Gi0/1           3
Gi0/2           2
```

Each time your script reads a line from the MAC address table, it will increment the counter associated with the interface name. This information needs to be stored until the script has finished processing all entries in the MAC address table and you are ready to output the final totals to the user.

In programming, the mechanism for storing data temporarily inside your program is called a data structure. There are different kinds of data structures that are useful for storing different kinds of data and accessing the data in different ways. Python supports a rich set of data structures that are useful for storing various types of information. The data structures are:

- **Lists:** sequences of items, which are indexed by number. Items can be heterogeneous (of different types). Items can be added to a list, removed from a list, and inserted into a list.
- **Dictionaries:** unordered tables of key-value pairs, which are indexed by the value of the key. Items can be added and removed from a dictionary.
- **Tuples:** sequences of items like lists, but a tuple is immutable, meaning its items cannot be changed.
- **Sets:** unordered collections of items. Items can be added or removed from a set, but there can be no duplicate values.



A list is a simple data structure that stores a set of items in a list that is ordered by sequence number (0, 1, 2, 3, 4). If you wanted to simply store the list of all MAC addresses in the table, you could store this information in a list:

```
[
  0 'fa16.3e14.a0f5',
  1 'fe00.6abe.ff01',
  2 '0050.56be.2548',
  3 '0410.50f2.3f44',
  4 '0072.7f33.debe'
]
```

A dictionary is a bit like a list, but instead of identifying the items in the list using a sequence number (such as item #2 is 0050.56be.2548), the items in a dictionary are identified by their name. The item name is also called a key.

You can store the list of interface names seen in the MAC address table in a dictionary. The key in this dictionary would be the interface name and the value would be the number of MAC addresses associated with the interface:

```
{
   'Gi0/1':'3',
   'Gi0/2':'2'
}
```

Each time your script encounters an entry in the MAC address table for interface Gi0/1, it can look up the entry in the dictionary by name (such as Gi0/1) and increment the value by 1. A dictionary is useful in this scenario because you can reference the entry in the dictionary by name.

Data structures can also be combined to provide more flexibility. If you are writing a script that lists the MAC addresses associated with each interface in addition to keeping a count, you could use a dictionary where the key is the interface name and the value is the list of MAC addresses associated with the interface.

```
{
  'Gi0/1':[
            0 '0050.56be.2548',
            1 '0410.50f2.3f44',
            2 '0072.7f33.debe'
          ],
  'Gi0/2':[
            0 'fa16.3e14.a0f5',
            1 'fe00.6abe.ff01'
          ]
}
```

Here are a few points to understand about these data structures:

- The most common and frequently used data structures are lists and dictionaries.
- Tuples never change, which means they can be used for fixed-value types of uses, such as keys for dictionary items.
- Sets are mainly used for testing whether an item is present (inclusion). You can also do operations on sets such as taking the intersection or the union of two sets.
- These data structures can be used together, so that you can have complex data representations, such as lists of dictionaries, dictionaries of lists or lists of tuples.
- 'Comprehensions' are a shorthand way of creating certain data structures. The syntax can be confusing at first but once understood, they can be employed to make your code more concise.
- Copying variable items in Python relates to the previously discussed idea that 'everything is an object'. This lesson will discuss the differences between assignment, copy (a shallow copy), and deep copy.

```
python $cat S03-1-list.py
from pprint import pprint

device_info = [] # Create my device_info list

# Open the file and read in the single line of device info
file = open('devices','r')
file_line = file.readline().strip()

print 'read line: ', file_line # ¡Print out the line I just read

# Here is the main part: use the string 'split' function to convert
# the comma-separated string into a list of items
device_info = file_line.split(',')

pprint(device_info) # Print out the list with nice formatting

file.close() # Be a good steward of resourcs and close the file
```