Copying Dictionaries

Remember the following operations do different things. Copying is generally the same as with lists:

- **Assignment:** Variable points at same object

```
dev2 = dev1  # dev2 now points at same object as dev1
```

- **Copy:** Variable points at a shallow copy of object

```
dev2 = dev1.copy()
```

- **Deep copy:** Variable points at a deep copy of object

```
dev2 = copy.deepcopy(dev1)
```

Dictionaries with other Data Types

List of dictionaries:

```
dev30 = {'name':'xrv-0','ip':'10.1.1.30','user':'cisco'}
dev31 = {'name':'xrv-1','ip':'10.1.1.31','user':'cisco'}
dev32 = {'name':'xrv-2','ip':'10.1.1.32','user':'cisco'}

dev_list = list()
dev_list.append(dev30) # appends dict dev30 to list
dev_list.append(dev31) # appends dict dev31 to list
dev_list.append(dev32) # appends dict dev32 to list
```

Dictionaries with Dictionaries

Dictionary of dictionaries:

```
dev30 = {'ip':'10.1.1.30','user':'cisco'}
dev31 = {'ip':'10.1.1.31','user':'cisco'}
dev32 = {'ip':'10.1.1.32','user':'cisco'}

devices = {}
devices['xrv-0'] = dev30 # sets item 'xrv-0' to dev30
devices['xrv-1'] = dev31 # sets item 'xrv-1' to dev31
devices['xrv-2'] = dev32 # sets item 'xrv-2' to dev32
```

Dictionaries of Lists of Dictionaries

Dictionary of lists of dictionaries

```
dev30 = {'name':'xrv-0','ip':'10.1.1.30','user':'cisco'}
dev31 = {'name':'xrv-1','ip':'10.1.1.31','user':'cisco'}
dev32 = {'name':'xrv-2','ip':'10.1.1.32','user':'cisco'}

xrv_devices = []
xrv_devices.append(dev30)
xrv_devices.append(dev31)
xrv_devices.append(dev32)

all_devices = {'xrv':xrv_devices,
               'nx-os':nx_devices,
               'ios':ios_devices}
```

Copying dictionaries in Python is the same as copying lists, meaning you should be aware of:

- **Assignment:** Assigning one variable name to reference (point to) another object that always exists. In the example, dev1 and dev2 will reference the same object:

```
dev2 = dev1
```

- **Copy:** You can make a copy – a shallow copy – of a dictionary by using the 'copy' function. Unlike lists, you do not need to import the copy function to do a shallow copy of a dictionary.

```
dev2 = dev1.copy()
```

- Deep copy: Shallow copies only make copies of the first level of items in a data structure. Doing a deep copy – that copies all levels of items – requires importing and using the `deepcopy` function.

```
from copy import deepcopy
dev2 = deepcopy(dev1)
```

List of Dictionaries

Dictionaries become even more useful when you begin to build structures that hold more information by combining dictionaries with other data structures.
Here is a list of dictionaries:

```
# Create three dictionaries holding device information for specific devices
dev30 = {'name':'xrv-0','ip':'10.1.1.30','user':'cisco'}
dev31 = {'name':'xrv-1','ip':'10.1.1.31','user':'cisco'}
dev32 = {'name':'xrv-2','ip':'10.1.1.32','user':'cisco'}

# Create a list to hold information about all of my devices created above
dev_list = [i for i in range(3)]
dev_list[0] = dev30 # sets list item 0 to dictionary dev30
dev_list[1] = dev31 # sets list item 1 to dictionary dev31
dev_list[2] = dev32 # sets list item 2 to dictionary dev32
```

The example above is creating three dictionaries to hold device information, called dev30, dev31, and dev32. The following Python commands are used to create dev_list:

- **List comprehension:** list comprehension is clearly identified by the square brackets surrounding the for statement. In this situation, a list comprehension is being used to create a dev_list with some initial empty values.
- **Range:** a functionality in `for` loops, which allows you to easily iterate a specified number of times. In this case, the range command is used to create a list with three empty items at index 0, 1, and 2.

> **Note**
>
> Python 'idioms' like the ones above will occur as you read sample code, and the purpose here is to help you become familiar with a few of the common ones.

At this point in the example of a list of dictionaries, there are three dictionaries representing the three devices. The next step is to add these dictionaries to the list, in order to create the list of dictionaries, by assigning the values for dev_list[0], dev_list[1], and dev_list[2] to the dictionaries that were just created:

```
dev_list[0] = dev30 # sets list item 0 to dictionary dev30
dev_list[1] = dev31 # sets list item 1 to dictionary dev31
dev_list[2] = dev32 # sets list item 2 to dictionary dev32
```

The result is a list that holds the following data:

```
[   {   'ip': '10.1.1.30', 'name': 'xrv-0', 'user': 'cisco'},
    {   'ip': '10.1.1.31', 'name': 'xrv-1', 'user': 'cisco'},
    {   'ip': '10.1.1.32', 'name': 'xrv-2', 'user': 'cisco'}]
```

The above result was printed by Python, using the 'pretty printing' function available, which is discussed later in this section.

Instead of creating a list with 3 items and then assigning the dictionaries to those items, you could have created an empty set and used the append function to add the dictionaries to the list. The code would look similar to:

```
dev_list = []
dev_list.append(dev30) # sets list item 0 to dictionary dev30
dev_list.append(dev31) # sets list item 1 to dictionary dev31
dev_list.append(dev32) # sets list item 2 to dictionary dev32
```

You will often be able to accomplish the same task multiple ways. Sometimes the choice is a matter of taste or personal preference, but as you become more proficient you will also learn how different Python methods impact performance or reusability of your code.

The choice is often up to the developer, but it is important to strive to be consistent with pre-existing coding styles, if updating or modifying existing code.

Dictionary of Dictionaries

You will sometimes wish to create dictionaries of dictionaries. This example is creating the same functionality as before, only this time using a different data structure. Here is the code:

```
dev30 = {'ip':'10.1.1.30','user':'cisco'}
dev31 = {'ip':'10.1.1.31','user':'cisco'}
dev32 = {'ip':'10.1.1.32','user':'cisco'}
devices = {}
devices['xrv-0'] = dev30 # sets item 'xrv-0' to dev30
devices['xrv-1'] = dev31 # sets item 'xrv-1' to dev31
devices['xrv-2'] = dev32 # sets item 'xrv-2' to dev32
```

In the example above, the same dev30, dev31, and dev32 dictionaries are created to hold device information. Then a dictionary called 'devices' is created. The final step is adding the individual device dictionaries to the dictionary called devices.
The outer dictionary however provides some extra value. Notice that the keys being used are actually the names for the specific devices. So the dictionary holds extra information, that wasn't present in the list of dictionaries.

The actual data looks as follows:

```
{   'xrv-0': {   'ip': '10.1.1.30', 'name': 'xrv-0', 'user': 'cisco'},
    'xrv-1': {   'ip': '10.1.1.31', 'name': 'xrv-1', 'user': 'cisco'},
    'xrv-2': {   'ip': '10.1.1.32', 'name': 'xrv-2', 'user': 'cisco'}}
```

Dictionary of Lists of Dictionaries

The final example features three levels of hierarchy, this time being a dictionary of lists of dictionaries.

The example demonstrates a use case where there are several devices in the network using various OS types. The goal is to keep track of these devices. The solution features the following:

- **Device Types dictionary:** An outer dictionary, with items representing the different types of devices (XR, NX, IOS).
- **Devices list:** For each device type in the dictionary, there is a list of devices from the network that are that specific type (XR or NX or IOS).
- **Device Info dictionary:** For each device in the list, there is a dictionary containing the device information for that specific device.

The sample code should look familiar – roughly the same as the list of dictionaries in the first example. Notice that the list is specific to a particular device type. And an extra dictionary has been added which contains the devices sorted by device type.
Here is the code:

```
# Create dictionaries holding device information for these three XRv devices
dev30 = {'name':'xrv-0','ip':'10.1.1.30','user':'cisco'}
dev31 = {'name':'xrv-1','ip':'10.1.1.31','user':'cisco'}
dev32 = {'name':'xrv-2','ip':'10.1.1.32','user':'cisco'}
# Create a list for our XRv devices that we have just defined
xrv_devices = []
xrv_devices.append(dev30)
xrv_devices.append(dev31)
xrv_devices.append(dev32)
all_devices = {'xrv':xrv_devices,  # all XRv devices go here
'nx-os':nx_devices, # all NX devices go here
'ios':ios_devices}  # all IOS devices go here
```

Only XRv devices are shown in the example, and so you should assume that 'nx_devices' and 'ios_devices' have been defined elsewhere.