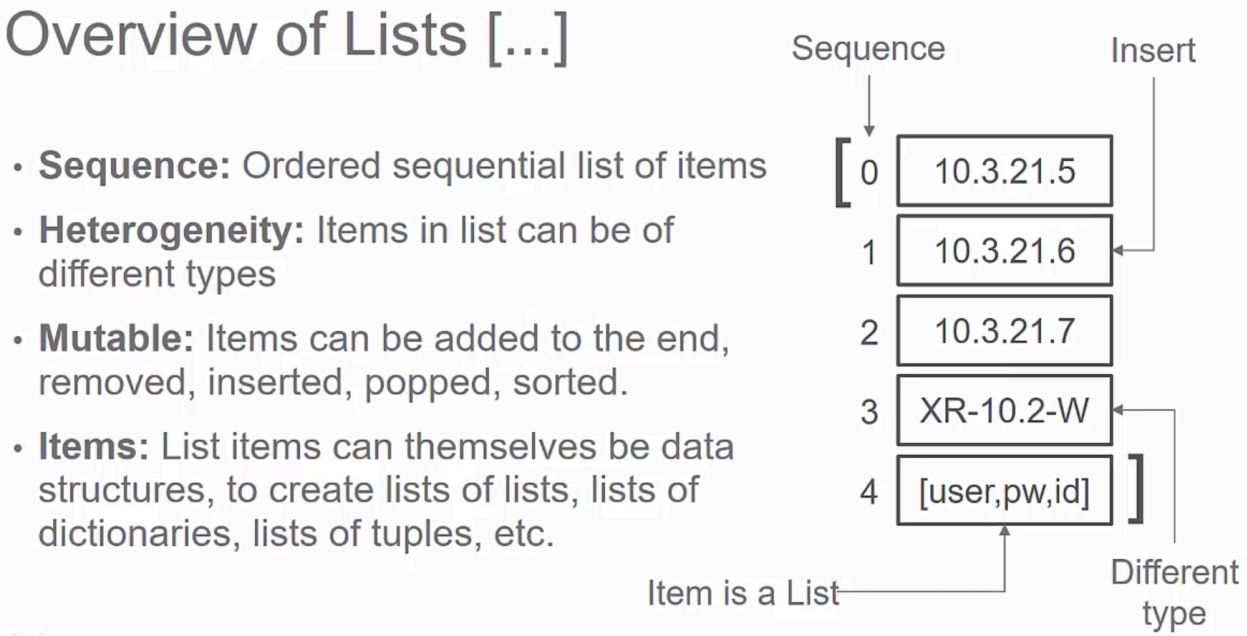
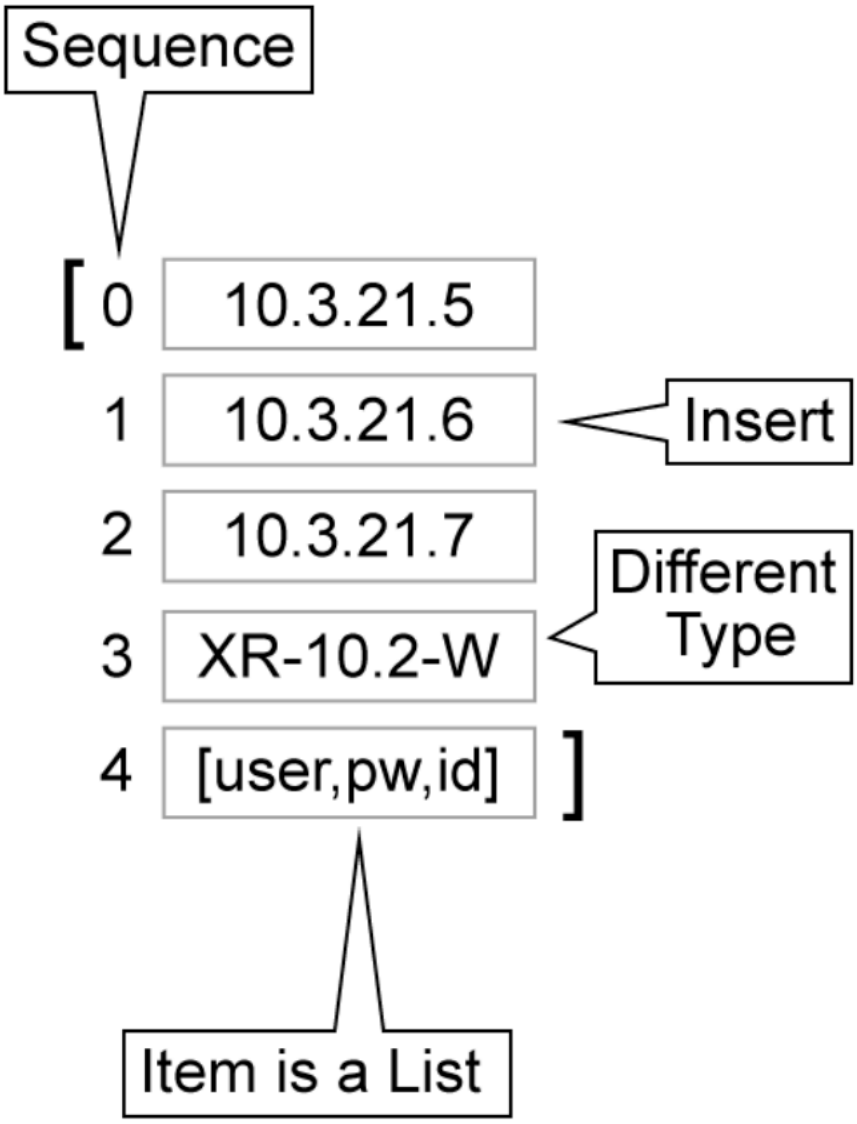



Unlike other programming languages where all of the elements of the data structure must be the same; arrays or vectors, in python they can be heterogenous.

Items in data structures, can themselves be data structures.

- **Sequence:** Ordered sequential list of items
- **Heterogeneous:** Items in a list can be of different types
- **Mutable:** Items can be added, removed, inserted, popped, and sorted
- **Items:** List items can be data structures, to create lists of lists, lists of dictionaries or lists of tuples



# Creating a List

**Create empty list:**

- Using the 'list()' function:

```
my_list = list() # creates the empty list 'my_list'
```

- Using empty square brackets:

```
my_list = []     # creates the empty list 'my_list'
```

**Create populated list:**

```
# create list 'dev_info' holding IP, username, and pw
dev_info=['1.1.1.1','username','password']
```

## Lists Overview

Lists are one of the most useful tools in the Python language. The following overview describes the basic attributes of a Python list:

- **Sequenced:** Python lists are sequenced, meaning they are ordered. You will therefore be doing operations such as append, insert, and remove.
- **Heterogeneity:** Python lists can hold items of differing types, as shown in the figure; some items are IP addresses, and one item is actually another list. However, heterogeneity is not required, and can be confusing. So often lists will be of the same type of item.
- **Mutable:** Lists can be changed – that is one of their fundamental purposes. There are operations to append, insert, remove, pop, even to sort a list.
- **Items:** The items in a list can be as simple as numbers or strings, or the items can be data structures such as other lists, tuples, sets, or dictionaries.

Notice from the diagram that square brackets ('[' and ']') indicate lists; when you see data that is listed in Python, if it is in square brackets it is probably a list. The main exception is when you are referring to a specific index for a data structure.

## Creating a List

- There are two methods you can use to create an empty list, each of which has the same effect. The following two examples show creation of an empty list:

```
my_list = list() # create an empty list
my_list = [] # create an empty list
```

- To create a populated list with fixed values, you use square brackets. The following example shows the creation of a populated list:

```
# create list holding IP address, username, and password
dev_info = ['10.3.21.5', 'username', 'password']
```

# Converting to a List

**Converting to a list from other data types:**

- From a tuple:

```
device_tuple = ('1.1.1.1','username','password')
device_list = list(device_tuple)
```

```
['1.1.1.1', 'username', 'password']
```

- From a String:

```
device_info_string = '1.1.1.1,username,password'
device_info = device_info_string.split(',')
```

```
['1.1.1.1', 'username', 'password']
```

Converting data structures is a common task

## Converting to a List

You can also take an existing item, such as a tuple or a string, and create lists from those types of data:

Converting a tuple to a list:

```
device_tuple = ('10.3.21.5', 'username', 'password')
device_list = list(device_tuple)
```

The resulting data from device_list would be:

```
['10.3.21.5', 'username', 'password']
```

Converting a string to a list:

```
device_string = '10.3.21.5 ,username, password'
device_list = device_string.split(',')
```

The example uses the string 'split' function, to separate the string using the character passed into the function (in this case, a comma).

The resulting data from device_list would be:

```
['10.3.21.5', 'username', 'password']
```

## Adding Items to a List

You can add items to the end of a list using 'append', or you can insert items at a specific location inside the list using 'insert'. Remember that lists are sequenced and so the order of the items may be important.

To append an item to the end of a list:

```
device_types = ['IOS','XR'] # create list
device_types.append('XE') # add 'XE' to list
```

The result of the append would be the device_types list as follows:

```
['IOS','XR','XE']
```

To insert an item at some location in the list:

```
device_types.insert(2,'NX') # insert 'NX' at index 2
```

The result of the insert would be the device_types list as follows:

```
['IOS','XR','NX','XE']
```

## Removing Items from a List

There are two ways to remove an item from a list:

- Remove by value: you remove an item by value using the 'remove' function and specifying the value that you want to remove.
- Delete using the specific index by using the `del` operation, and specifying the index of the item you want to remove.

> **Note**
>
> Indices start at 0 in Python.

Remove by value:

```
device_list.remove('NX')  # remove item matching 'NX'
```

The result of this removal, using the previous 'device_list' examples:

```
['IOS','XR','XE']
```

Remove by index using `del`:

```
del device_list[1] # remove item at index 1
```

**Note**

Remember, Python indexes start at 0. Before the del function is executed, 'IOS' is at index 0, 'XR' is at index 1, and 'XE' is at index 2.

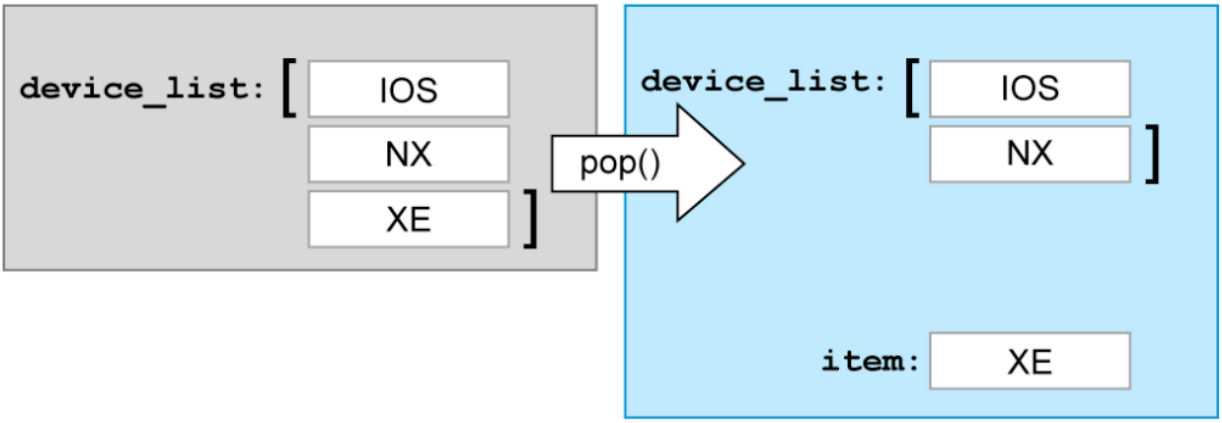
The result of this deletion:

```
['IOS','XE']
```

## Popping an Item off a List

It is also possible to pop an item off a list, where the item is removed from the list but you have a new variable that references the popped item. Popping can occur from the end of the list, from the beginning, or from any point in between.

- Get an item and remove it from a list by index using pop()




Assuming a device_list with three values 'IOS', 'NX', and 'XE':

```
item = device_list.pop() # pop the last item off list
# and point 'item' it it
```

The resulting value of 'device_list' will be:

```
['IOS','NX']
```

The resulting value of 'item' will be:

```
'XE'
```

```
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 dictionary with nice formatting

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