Comparison Overview

- **Operators**:
  - == Equal to
  - != Not equal to
  - < Less than
  - > Greater than
  - <= Less than or equal to
  - >= Greater than or equal to
  - 'in' Present in
- **Numbers**: arithmetic compare
- **Strings**: lexicographic compare
- **Strings**: substring functions
- **Lists, dictionaries, tuples, sets**
  - Lists: compared item by item
  - Tuples: compared item by item
  - Sets: membership ('in')
  - Dicts: compared (k,v) by (k,v)

Simple Comparisons

- Equality: '=='

```
if version == dev_info.version:
```

- Inequality: '!='

```
if version != dev_info.version:
```

- Arithmetic: '<', '>', '<=', '>='

```
if utilization > max_utilization:
```

- Membership: 'in'

```
if dev_info.os_type in cisco_os_types:
```

Data Structure Comparison

Lists, Tuples:

- Compared in order item by item, including items which are themselves lists, dictionaries, or tuples.

Dictionaries:

- Sorted (key,value) items compared in order item by item, including items which are themselves lists, dictionaries, or tuples.

Other objects (classes, functions, etc.)

- Compare equal only if they are actually the same object.

Simple comparisons refers to comparisons of basic items such as strings and numbers:

- Equality or inequality of strings or numbers. The examples compare the device version with some expected version number to see if it needs to be updated:

```
if version == dev_info.version:

if version != dev_info.version:
```

- Arithmetic comparison of items to provide equality or greater/less than evaluation. The example checks to see if the calculated utilization for a link is greater than the maximum:

```
if utilization > max_utilization:
```

- Set comparison involves testing if a specific value is a member of the set, or whether it is an item in a list or tuple, or whether it is a key in a dictionary. The example tests if the device type is a Cisco device type:

```
if dev_info.os_type in cisco_os_types:
```