7 Containers II

Key terms: `set dict sort sorted` **key value**

Reading: Severance 9

Exercise: Create a dictionary object containing the names and years of birth of your family members, then print out their names ordered from youngest to eldest.

Unlike the `list` or `tuple`, two of Python's container classes, the `set` and the `dict`, are unordered and include only unique entries. This makes them powerful tools for collating matches when inputs contain duplicates.

```
# Who called me today? Show the unique callers from my call list.

calls = ['Vimberto', 'Hinoki', 'Masha', 'Hinoki', 'Hinoki']
print(set(calls))
```

The `dict`, or dictionary, is a key-value store that accommodates all kinds of look-ups. The dictionary consists of any number of pairs, each containing two corresponding but distinct parts: a key and a value. Both halves can be numbers or strings, among other types; it is usual to make dictionaries with all keys having the same type, and all values doing so as well.

The keys are unique, so all the pairs are unique. Values may match each other; consider the dictionary `delicious = { 'Grape': True, 'Avocado': True }`. Like the integer indices of a list, dictionary values can be looked up using square brackets (`cities['Chile']`) or with the `get()` function (`cities.get('Chile')`). Attempting to access an undefined key is signalled by a `KeyError` in the former style.

Unlike lists, sets and dictionaries have no order. A new sorted list containing elements from any type of container is returned from the `sorted()` built-in. Lists can also be sorted in place with their `sort()` method; tuples cannot. Reversing a sort is as simple as adding `reverse=True`. To use sort orders other than the alphabetical or numeric defaults, pass the argument `key` to call another comparison function.

```
# Add the JCPA to a dictionary of past treaties:

treaties = {
 'Nuclear Test Ban Treaty': 1963,
 'Hague Conventions': 1899,
 'Peace of Olomouc': 1479,
 'Treaty of Windsor': 1175,
 'Ili River Treaty': 638
 }
treaties [ 'Joint Comprehensive Plan of Action' ] = 2015

# Order treaties by name (key):
print (sorted(treaties))

# Order treaties by year (value):
print (sorted(treaties, key=treaties.get))
```

A hash function transforms immutable objects, like numbers and strings, into unique codes. The elements of a `set` and the keys of a `dict` can only contain hashable objects, and to be hashable they must be immutable. If an object changed after hashing, its hash would change, necessitating a lot of work to keep hashes up to date. The `frozenset` is an immutable kind of `set` providing the same benefits as a `tuple` does over a `list`. As such, a `frozenset` can be a key in a `set`, but not vice-versa.

Next, we will cover block and inline conditional expressions.