Exercise 39: Dictionaries, Oh Lovely Dictionaries

Now I have to hurt you with another container you can use, because once you learn this container a massive world of ultra-cool will be yours. It is the most useful container ever: the dictionary.

Python calls them "dicts." Other languages call them "hashes." I tend to use both names, but it doesn't matter. What does matter is what they do when compared to lists. You see, a list lets you do this:

```
>>> things = ['a', 'b', 'c', 'd']
>>> print things[1]
b
>>> things[1] = 'z'
>>> print things[1]
z
>>> print things
['a', 'z', 'c', 'd']
>>>
```

You can use numbers to "index" into a list, meaning you can use numbers to find out what's in lists. You should know this about lists by now, but make sure you understand that you can *only* use numbers to get items out of a list.

What a `dict` does is let you use *anything*, not just numbers. Yes, a dict associates one thing to another, no matter what it is. Take a look:

```
>>> stuff = {'name': 'Zed', 'age': 36, 'height':
6*12+2}
>>> print stuff['name']
Zed
>>> print stuff['age']
36
>>> print stuff['height']
```

```
74
>>> stuff['city'] = "San Francisco"
>>> print stuff['city']
San Francisco
>>>
```

You will see that instead of just numbers we're using strings to say what we want from the `stuff` dictionary. We can also put new things into the dictionary with strings. It doesn't have to be strings though. We can also do this:

```
>>> stuff[1] = "Wow"
>>> stuff[2] = "Neato"
>>> print stuff[1]
Wow
>>> print stuff[2]
Neato
>>> print stuff
{'city': 'San Francisco', 2: 'Neato',
    'name': 'Zed', 1: 'Wow', 'age': 36,
    'height': 74}
>>>
```

In this code I used numbers, and then you can see there are numbers and strings as keys in the dict when I print it. I could use anything. Well, almost but just pretend you can use anything for now.

Of course, a dictionary that you can only put things in is pretty stupid, so here's how you delete things, with the `del` keyword:

```
>>> del stuff['city']
>>> del stuff[1]
>>> del stuff[2]
>>> stuff
{'name': 'Zed', 'age': 36, 'height': 74}
>>>
```

We'll now do an exercise that you *must* study very carefully. I want you to type this exercise in and try to understand what's going on. Take note of when I put things in a dict, get from them, and all the operations I use here.

```
   # create a mapping of state to abbreviation
1  states = {
2      'Oregon': 'OR',
```

```
    'Florida': 'FL',
    'California': 'CA',
    'New York': 'NY',
    'Michigan': 'MI'
}

# create a basic set of states and some cities in them
cities = {
    'CA': 'San Francisco',
    'MI': 'Detroit',
    'FL': 'Jacksonville'
}

# add some more cities
cities['NY'] = 'New York'
cities['OR'] = 'Portland'

# print out some cities
print '-' * 10
print "NY State has: ", cities['NY']
print "OR State has: ", cities['OR']

# print some states
print '-' * 10
print "Michigan's abbreviation is: ", states['Michigan']
print "Florida's abbreviation is: ", states['Florida']

# do it by using the state then cities dict
print '-' * 10
print "Michigan has: ", cities[states['Michigan']]
print "Florida has: ", cities[states['Florida']]

# print every state abbreviation
print '-' * 10
for state, abbrev in states.items():
    print "%s is abbreviated %s" % (state, abbrev)

# print every city in state
print '-' * 10
for abbrev, city in cities.items():
    print "%s has the city %s" % (abbrev, city)

# now do both at the same time
print '-' * 10
for state, abbrev in states.items():
    print "%s state is abbreviated %s and has city %s" %
(
        state, abbrev, cities[abbrev])

print '-' * 10
# safely get a abbreviation by state that might not be
there
state = states.get('Texas', None)

if not state:
```

```
58      print "Sorry, no Texas."
59
60  # get a city with a default value
61  city = cities.get('TX', 'Does Not Exist')
    print "The city for the state 'TX' is: %s" % city
```

What You Should See

```
$ python ex39.py
----------
NY State has:  New York
OR State has:  Portland
----------
Michigan's abbreviation is:  MI
Florida's abbreviation is:  FL
----------
Michigan has:  Detroit
Florida has:  Jacksonville
----------
California is abbreviated CA
Michigan is abbreviated MI
New York is abbreviated NY
Florida is abbreviated FL
Oregon is abbreviated OR
----------
FL has the city Jacksonville
CA has the city San Francisco
MI has the city Detroit
OR has the city Portland
NY has the city New York
----------
California state is abbreviated CA and has city San
Francisco
Michigan state is abbreviated MI and has city
```

```
Detroit
New York state is abbreviated NY and has city New
York
Florida state is abbreviated FL and has city
Jacksonville
Oregon state is abbreviated OR and has city
Portland
----------
Sorry, no Texas.
The city for the state 'TX' is: Does Not Exist
```

Study Drills

1. Do this same kind of mapping with cities and states/regions in your country or some other country.
2. Go find the Python documentation for dictionaries (aka dicts, dict) and try to do even more things to them.
3. Find out what you *can't* do with dictionaries. A big one is that they do not have order, so try playing with that.

Common Student Questions

What is the difference between a list and a dictionary?

A list is for an ordered list of items. A dictionary (or `dict`) is for matching some items (called "keys") to other items (called "values").

What would I use a dictionary for?

Any time you have to take one value and "look up" another value. In fact you could call dictionaries "look up tables."

What would I use a list for?

A list is for any sequence of things that need to go in order, and you only need to look them up by a numeric index.

What if I need a dictionary, but I need it to be in order?

Take a look at the `collections.OrderedDict` data structure in Python. Search for it online to find the documentation.