Exercise 39: Dictionaries, Oh Lovely Dictionaries

You are now going to learn about the Dictionary data structure in Python. A Dictionary (or "dict") is a way to store data just like a `list`, but instead of using only numbers to get the data, you can use almost anything. This lets you treat a `dict` like it's a database for storing and organizing data.

Let's compare what can dicts can do to what lists can do. You see, a list lets you do this:

```
>>> things = ['a', 'b', 'c', 'd']
>>> print things[1]
b
>>> things[1] = 'z'
>>> print things[1]
z
>>> 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': 39, 'height': 6 * 12 + 2}
>>> print stuff['name']
Zed
>>> print stuff['age']
39
>>> 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
>>> stuff
{'city': 'San Francisco', 2: 'Neato', 'name': 'Zed', 1: 'Wow', 'age': 39, '
```

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': 39, 'height': 74}
```

A Dictionary Example

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

to their abbreviations, and then the abbreviations to cities in the states. Remember, "mapping" or "associating" is the key concept in a dictionary.

```
    # create a mapping of state to abbreviation
    states = {
        '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')

if not state:
    print "Sorry, no Texas."

# get a city with a default value
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
```

What Dictionaries Can Do

Dictionaries are another example of a data structure, and like lists they are one of the most commonly used data structures in programming. A dictionary is used to *map* or *associate* things you want to store to keys you need to get them. Again, programmers don't use a term like "dictionary" for something that doesn't work like an actual dictionary full of words, so let's use that as our real world example.

Let's say you want to find out what the word "Honorificabilitudinitatibus" means. Today you would simply copy-paste that word into a search engine and then find out the answer, and we could say a search engine is like a really huge super complex version of the *Oxford English Dictionary* (OED). Before search engines what you would do is this:

1. Go to your library and get "the dictionary". Let's say it's the OED.
2. You know "honorificabilitudinitatibus" starts with the letter 'H' so you look on the side of the book for the little tab that has 'H' on it.
3. Then you'd skim the pages until you are close to where "hon" started.
4. Then you'd skim a few more pages until you found "honorificabilitudinitatibus" or hit the beginning of the "hp" words and realize this word isn't in the OED.
5. Once you found the entry, you'd read the definition to figure out what it means.

This process is nearly exactly the way a dict works, and you are basically "mapping" the word "honorificabilitudinitatibus" to its definition. A dict in Python is just like a dictionary in the real world like the OED.

Study Drills

1. Do this same kind of mapping with cities and states/regions in your country or some other country.
2. Find the Python documentation for dictionaries 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?

When 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?

Use a list for any sequence of things that need to be 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.

Buy DRM-Free

When you buy directly from the author, Zed A. Shaw, you'll get a professional quality PDF and hours of HD Video, all DRM-free and yours to download and use as you see fit.

$29.99

BUY DIRECTLY FROM THE AUTHOR
(HTTPS://SHOP.LEARNCODETHEHARDWAY.ORG/ACCESS/BUY/2/)

OR, YOU CAN READ LEARN PYTHON THE HARD WAY FOR FREE (HTTPS://LEARNPYTHONTHEHARDWAY.ORG/BOOK/) RIGHT HERE, VIDEO LECTURES NOT INCLUDED.

Other Buying Options

BUY ON AMAZON (HTTP://BIT.LY/AMZNLPTHW)

BUY A HARD COPY FROM THE PUBLISHER (HTTP://BIT.LY/INFORMITLPTHW)

BUY A HARD COPY FROM BARNES & NOBLE (HTTP://BIT.LY/BNLPTHW)

Previous (ex38.html) Next (ex40.html)

HOME (/) ABOUT (HTTPS://LEARNCODETHEHARDWAY.ORG/ABOUT/) CONTACT (HTTPS://LEARNCODETHEHARDWAY.ORG/CONTACT/)

(https://twitter.com/lzsthw)