CMSC 201 Fall 2018
Lab 13 – Dictionaries

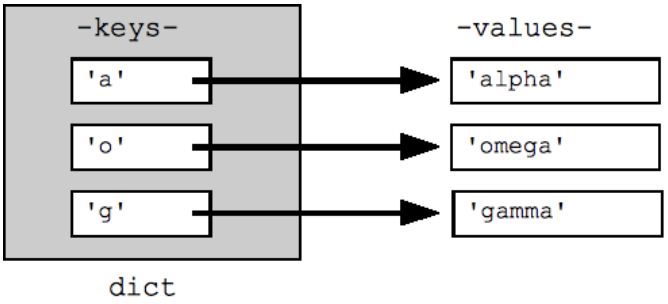
Assignment: Lab 13 – Dictionaries
Due Date: **During discussion**, December 3rd through December 6th
Value: 10 points (during lab)

This week's lab will give you practice with using dictionaries.

(Having concepts explained in a new and different way can often lead to a better understanding, so make sure to pay attention as your TA explains.)

Part 1A: Review – Dictionaries

A very useful data type available in Python is the ***dictionary***. Dictionaries are sometimes found in other languages as "associative memories" or "associative arrays." Dictionaries are data structures that map a key to a value. So, in the example below, we have a dictionary that maps the key '**a**' to the value '**alpha**'; the key '**o**' to the value '**omega**'; and the key '**g**' to the value '**gamma**'.



(Image from https://developers.google.com/edu/python/dict-files)

We can create this dictionary with this line of code:

```
greek = {"a": "alpha", "o": "omega", "g": "gamma"}
```

Dictionaries may look a lot like lists, but there are a few key differences:

1. A dictionary uses curly braces instead of square brackets
2. A dictionary is made up of (key, value) pairs
3. The key and value are separated by a colon (**:**)
4. The keys must be unique (just like the indexes of a list are unique)
    a. The keys can only be immutable data types

Lists are indexed by ***order***, which we see as a range of numbers. Dictionaries are indexed by ***association***, or their key values. Keys can be any immutable type, and every key in a dictionary must be unique. Strings, floats, and integers are common choices for a key.

Part 1B: Review – Dictionary Functions

We can start by looking at how we could create a simple dictionary. Let's create a new dictionary called **animals**.

```
animals = {"Clifford" : "dog",    "Hedwig" : "owl",
           "George"   : "monkey", "Kha"    : "snake",
           "Laika"    : "dog"}
```

In this dictionary, we have mapped famous animals, using their name as the key, and their species as the value. Since there may be multiple animals of the same species (*e.g.*, Clifford and Laika are both dogs), it makes sense to use the unique value (the name) as the key.

Using a dictionary, we can perform a number of operations. The examples below use the **animals** dictionary defined above.

A. **Iterate** through the dictionary:

```
keys = list( animals.keys() )
for i in range(len(keys)):
    print(keys[i], "is a famous", \
                animals[ keys[i] ])
```

OUTPUT:

```
Laika is a famous dog
George is a famous monkey
```

(and so on)

B. **Access** a specific entry:

```
print("Kha is the", animals["Kha"], \
      "from 'The Jungle Book'")
```

OUTPUT:

```
Kha is the snake from 'The Jungle Book'
```

C. **Safely access** a specific entry:

```
print("Kha is the", animals.get("Kha"), \
      "from 'The Jungle Book'")
# if there was no "Kha" key, this would simply
# print out None, rather than crashing
```

D. **Add** something to the dictionary:

```
animals["Punxsutawney Phil"] = "groundhog"
```

E. **Updating** the value of something in the dictionary:

```
animals["Hedwig"] = "snowy owl"
```

F. **Deleting** something from the dictionary:

```
# Laika was a Soviet space dog, the first
# animal to orbit the Earth.  She did not
# survive more than a few hours in space. :(
del animals["Laika"]
```

G. **Checking if a key is present** in the dictionary:

```
"Laika" in animals
# this will return False, as Laika's no longer in
the dictionary

"Clifford" in animals
# this will return True
```

H. Dictionaries also have methods that enable some additional functionality. In addition to the commands and examples above, here are some of the more helpful methods we can use.

These both return a "view" by default, so we must cast them to a list to use them.

a. **`list( animals.values() )`**
   Returns a list of the values in dictionary **`animals`**

```
['groundhog', 'snowy owl', 'monkey', 'dog',
 'snake']
```

b. **`list( animals.keys() )`**
   Returns a list of the keys in dictionary **`animals`**

```
['Punxsutawney Phil', 'Hedwig', 'George',
 'Clifford', 'Kha']
```

Part 1C: New Material – Lists as Values

Although we didn't discuss it in detail during class, it is possible to use lists as the value in a (key, value) pair, and to update the list as new items are entered that belong with that key. In order to do so, though, we must pay attention to and handle two different scenarios:

1. The key does not yet exist in the dictionary
    a. We must create a new key and start the list from scratch
2. The key already exists in the dictionary
    a. We must append to the existing list, without overwriting it

To accomplish these, the first thing to do is use the `.get()` function to access a key's value. The `.get()` function is safer than square brackets, because if the key does not exist it will return `None` (where square brackets would cause an error).

If we see that the key already exists, we can create a key:value pair with a single element list for the value. If it does already exist, we simply append to the existing list. For example, code to accomplish that might look like the following.

```
# if the key doesn't already exist in the dictionary
if myDict.get(theKey) == None:
    myDict[theKey] = [newValue]
else:
    myDict[theKey].append(newValue)
```

And when we want to access something in the list of values, we'll need to first index into the dictionary (using the key) and then into the list (using regular indexes).

```
# print all the elements of a key's value list
valueList = myDict[knownKey]
for i in range(len(valueList)):
    print(valueList[i])
```