Dictionaries are unordered mappings for storing objects.

Previously, we saw how lists store objects in an ordered sequence, dictionaries use a key-value pairing instead

This key-value pair allows users to quickly grab objects without needing to know an index location

Dictionaries use curly braces and colons to signify the keys and their associated values like so:

{'key1':'value1','key2':'value2'}

So when to choose a list and when to choose a dictionary?

Dictionaries:

- Objects retrived by key name
- Unordered and can not be stored
- Quickly retrieve a value without knowing its exact index location

Lists:

- Objects retrieved by location
- Ordered Sequence can be indexed or sliced and sorted

How to consutrct a Dictionary:

In [1]:
```
my_dict = {'key1' : 'value1', 'key2' : 'value2' }
```

In [2]:
```
my_dict
```

Out[2]:
```
{'key1': 'value1', 'key2': 'value2'}
```

In [3]:
```
my_dict['key1']
```

Out[3]:
```
'value1'
```

In [4]:
```
prices_lookup = {'apple': 2.99, 'oranges':1.99, 'milk':5.80}
```

In [6]:
```
prices_lookup['apple']
```

Out[6]:
```
2.99
```

Dictionaries can hold different variables (strings, floats, integers) however, they can also hold lists and other dictionaries! Like so:

In [7]:
```
d = {'k1' : 123, 'k2':[0,1,2], 'k3':{'insideKey':100}}
```

In [9]:
```
d['k2']
```

Out[9]:
```
[0, 1, 2]
```

In [10]:
```
d['k3']
```

Out[10]:
```
{'insideKey': 100}
```

In [11]:
```
d['k3']['insideKey']
```

Out[11]:
```
100
```

In [12]:
```
d['k2'][2]
```

Out[12]:
```
2
```

Pictured right above, we called the key value 'k2' and then index'd the '2' string by indexing [2]

In [13]:
```
d = {'key1' : ['a','b','c']}
```

In [14]:
```
d['key1'][2]
```

Out[14]:
```
'c'
```

In [15]:
```
d['key1']
```

Out[15]:
```
['a', 'b', 'c']
```

In [16]:
```
d['key1'][2].upper()
```

Out[16]:
```
'C'
```

Pictured right above, we were able to stack calls, we retrieved key1, indexed c [2], and also made the C an upper case

If we want to add new key value pairs to a dictionary, we can do so like so:

In [17]:
```
d = {'k1': 100, 'k2': 200}
```

In [18]:
```
d
```

Out[18]:
```
{'k1': 100, 'k2': 200}
```

We have created, new dictionary pictured above, lets add a k3

In [20]:
```
d['k3'] = 300
```

In [21]:
```
d
```

Out[21]:
```
{'k1': 100, 'k2': 200, 'k3': 300}
```

We can also overwrite an existing key pair like so:

In [22]:
```
d['k1'] = 'NEW VALUE'
```

In [23]:
```
d
```

Out[23]:
```
{'k1': 'NEW VALUE', 'k2': 200, 'k3': 300}
```

In [24]:
```
d = {'k1': 100, 'k2': 200, 'k3': 300}
```

If we wish to see all the key pairs/values in a dictionary, we can do so like so:

In [26]:
```
d.keys()
```

Out[26]:
```
dict_keys(['k1', 'k2', 'k3'])
```

In [28]:
```
d.values()
```

Out[28]:
```
dict_values([100, 200, 300])
```

In [29]:
```
d.items()
```

Out[29]:
```
dict_items([('k1', 100), ('k2', 200), ('k3', 300)])
```

Dictionaries - FAQ

Do dictionaries keep an order? How do I print the values of the dictionary in order?

Dictionaries are mappings and do not retain order! If you do want the capabilities of a dictionary but you would like ordering as well, check out the ordereddict object lecture later on in the course

Questions:

Question 1 Is this statement true or false? Dictionaries retain order and are a sequence

A. True B. False

Answer: B. False

Question 2 Given d={'k1':[1,2,3]}

What is the output of d['k1'][1]

A. None, an error occurs B. 1 C. 2 D. 3

Answer: C. 2

Question 3 Is this statement True or False? Dictionaries are immutable.

Choose the answer below.

A. True B. False

Answer: A. False