Dictionaries

In this lecture we will look at a brief introduction to dictionaries:

1.) Constructing a Dictionary 2.) Accessing objects from a dictionary 3.) Nesting Dictionaries 4.) Basic Dictionary Methods

So what are mappings? Mappings are a collection of objects that are stored by a key, unlike a sequence that stored objects by their relative position. This is an important distinction, since mappings won't retain order since they have objects defined by a key.

A Python dictionary consists of a key and then an associated value. That value can be almost any Python object.

So what are mappings? Mappings are a collection of objects that are stored by a key, unlike a sequence that stored objects by their relative position. This is an important distinction, since mappings won't retain order since they have objects defined by a key.

A Python dictionary consists of a key and then an associated value. That value can be almost any Python object.

Constructing a Dictionary

Let's see how we can construct dictionaries to get a better understanding of how they work!

In [1]:
```
# Make a dictionary with {} and : to signify a key and a value
my_dict = {'key1':'value1','key2':'value2'}
```

In [2]:
```
# Call values by their key
my_dict['key1']
```

Out[2]:
```
'value1'
```

Its important to note that dictionaries are very flexible in the data types they can hold. For example:

In [3]:
```
my_dict = {'key1':123,'key2':[12,23,33],'key3':['item0','item1','item2']}
```

In [4]:
```
#Lets call items from the dictionary
my_dict['key3']
```

Out[4]:
```
['item0', 'item1', 'item2']
```

In [5]:
```
# Can call an index on that value
my_dict['key3'][0]
```

Out[5]:
```
'item0'
```

In [6]:
```
#Can we even call methods on that value ? Let's Try
my_dict['key3'][0].upper()
```
Out[6]:
```
'ITEM0'
```

We can effect the values of a key as well. For Example:

In [7]:
```
my_dict['key1']
```
Out[7]:
```
123
```

In [8]:
```
# Add 7 to the value
my_dict['key1'] = my_dict['key1'] + 7
```

In [9]:
```
#Check
my_dict['key1']
```
Out[9]:
```
130
```

We can also create keys by assignment. If we started off with an empty dictionary, we could continually add to it:

In [10]:
```
# Create a new dictionary
d = {}
```

In [11]:
```
# Create a new key through assignment
d['flower'] = 'Rose'
```

In [14]:
```
# Can do this with any object
d['number'] = 11
```

In [15]:
```
#Show
d
```
Out[15]:
```
{'flower': 'Rose', 'number': 11}
```

Nesting with Dictionaries

In [16]:
```
# Dictionary nested inside a dictionary nested in side a dictionary
d = {'key1':{'nestkey':{'subnestkey':'value'}}}
```

Let's see how we can grab that value:

```
In [17]: # Keep calling the keys
         d['key1']['nestkey']['subnestkey']

Out[17]: 'value'
```

A few Dictionary Methods

There are a few methods we can call on a dictionary. Let's get a quick introduction to a few of them:

```
In [18]: # Create a typical dictionary
         d = {'key1':1,'key2':2,'key3':3}
```

```
In [19]: # Method to return a list of all keys
         d.keys()

Out[19]: ['key3', 'key2', 'key1']
```

```
In [20]: # Method to grab all values
         d.values()

Out[20]: [3, 2, 1]
```

```
In [21]: # Method to return tuples of all items  (we'll learn about tuples soon)
         d.items()

Out[21]: [('key3', 3), ('key2', 2), ('key1', 1)]
```

Final Remarks

Now you know how to create a dictionary and how to retrieve values from it.