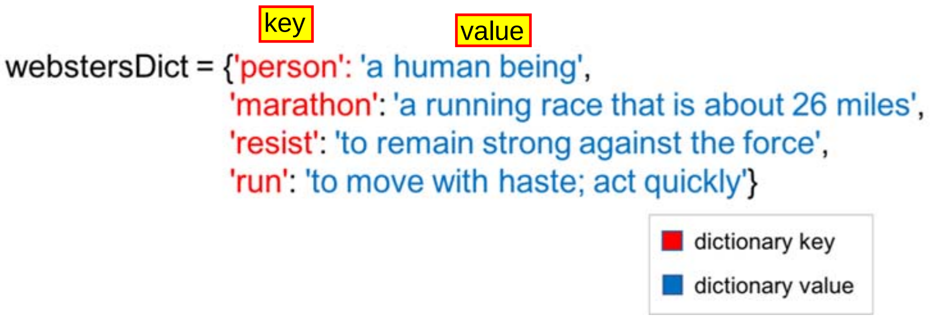
```
Initialize a Dictionary
dict = {'one':1, 'two':'two, 'three':3.5}
```

Python_Dictionary and Dictionary Methods

Michael Galarnyk

Jun 23, 2017


key
value
webstersDict = {'person': 'a human being',
'marathon': 'a running race that is about 26 miles',
'resist': 'to remain strong against the force',
'run': 'to move with haste; act quickly'}
dictionary key
dictionary value

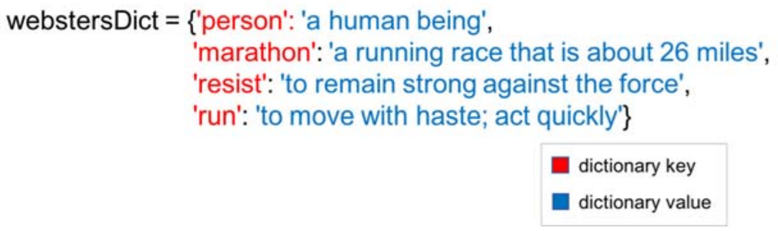
Define a dictionary. Keys are in red. Values are in blue.

This Python Dictionary tutorial covers:

- How to Define a Dictionary
- How to Access Values in a Dictionary
- How to Add, Update, and Delete Keys from a Dictionary
- Dictionary Methods
- How to Iterate through a Dictionary

Define a Dictionary

Dictionaries are written within curly brackets {}.


webstersDict = {'person': 'a human being',
'marathon': 'a running race that is about 26 miles',
'resist': 'to remain strong against the force',
'run': 'to move with haste; act quickly'}
dictionary key
dictionary value

Define a dictionary. Keys are in red. Values are in blue.

```
# Define a dictionary code
websterssDict = {'person': 'a human being',
                'marathon': 'a running race that is about 26 miles',
                'resist': 'to remain strong against the force',
                'run': 'to move with haste; act quickly'}
```

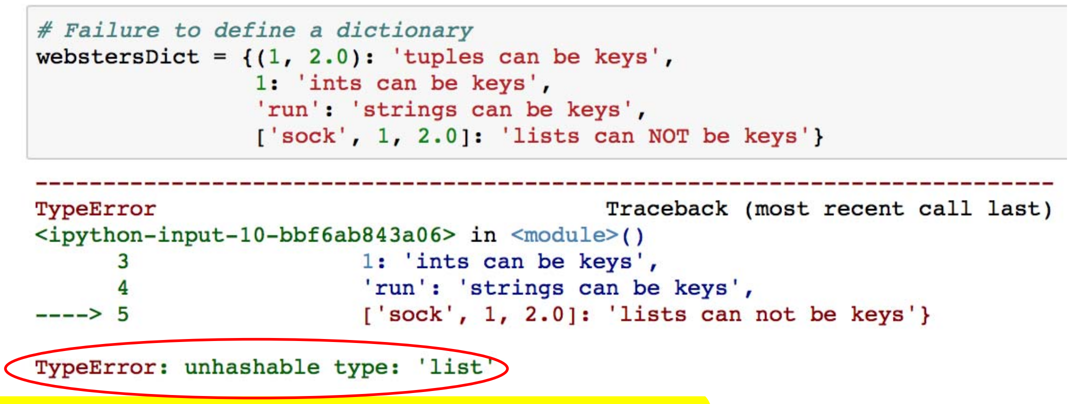
The dictionary webstersDict used strings as keys in the dictionary, but dictionary keys can be any immutable data type (numbers, strings, tuples etc). Dictionary values can be just about anything (int, lists, functions, strings, etc).

For example, the dictionary below, genderDict has ints as keys and strings as values.

```
# Define a dictionary
genderDict = {0: 'male',
              1: 'female'}
```

An important point to emphasize is that if you try to make a key a mutable datatype (like a list), you will get an error.

```
# Failure to define a dictionary
webstersDict = {(1, 2.0): 'tuples can be keys',
                1: 'ints can be keys',
                'run': 'strings can be keys',
                ['sock', 1, 2.0]: 'lists can NOT be keys'}
```



```
# Failure to define a dictionary
webstersDict = {(1, 2.0): 'tuples can be keys',
                1: 'ints can be keys',
                'run': 'strings can be keys',
                ['sock', 1, 2.0]: 'lists can NOT be keys'}

---------------------------------------------------------------------------
TypeError                                 Traceback (most recent call last)
<ipython-input-10-bbf6ab843a06> in <module>()
      3                 1: 'ints can be keys',
      4                 'run': 'strings can be keys',
----> 5                 ['sock', 1, 2.0]: 'lists can not be keys'}

TypeError: unhashable type: 'list'
```

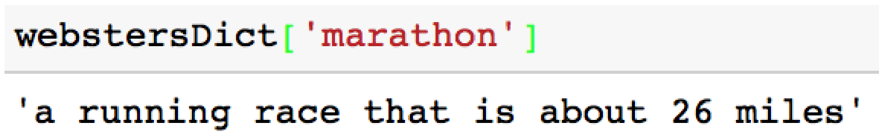
Failure to define a dictionary with a list as a key. Lists are NOT immutable

Access Values in a Dictionary

To access a dictionary value, you can use square brackets [].

For example, the code below uses the key 'marathon' to access the value 'a running race that is about 26 miles'.

```
# Get value of the 'marathon' key
webstersDict['marathon']
```



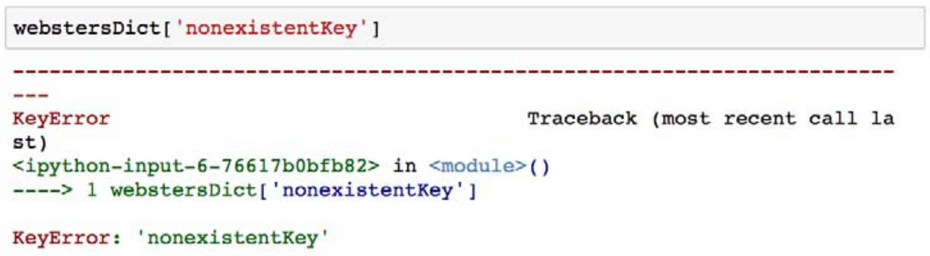
```
webstersDict['marathon']

'a running race that is about 26 miles'
```

Access the key 'marathon'

Keep in mind that you will get a KeyError if you try to access a value for a key that **does not exist**.

```
# Try to get value for key that does not exist
webstersDict['nonexistentKey']
```



```
webstersDict['nonexistentKey']

---------------------------------------------------------------------------
---
KeyError                                  Traceback (most recent call la
st)
<ipython-input-6-76617b0bfb82> in <module>()
----> 1 webstersDict['nonexistentKey']

KeyError: 'nonexistentKey'
```

KeyError will result if you try and look up a key that does not exist.

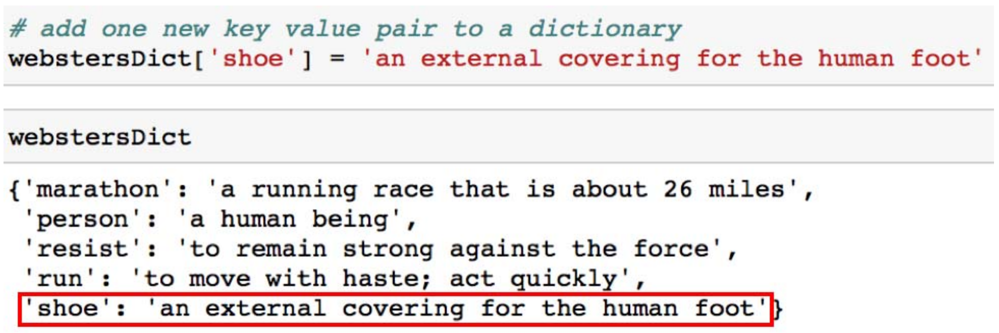
In the Dictionary Methods section, you will see the utility of using the dictionary **method get** to avoid KeyErrors.

Add, Update, and Delete Keys from a Dictionary

Add or Update Key

You can add a new key-value pair.

```
# add one new key value pair to a dictionary
webstersDict['shoe'] = 'an external covering for the human foot'
```



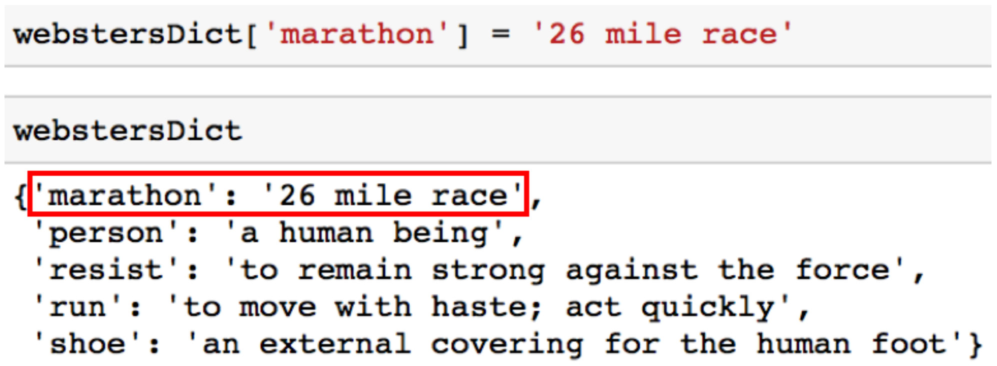
```
# add one new key value pair to a dictionary
webstersDict['shoe'] = 'an external covering for the human foot'
```

```
webstersDict
```

```
{'marathon': 'a running race that is about 26 miles',
 'person': 'a human being',
 'resist': 'to remain strong against the force',
 'run': 'to move with haste; act quickly',
 'shoe': 'an external covering for the human foot'}
```

Add the new key ‘shoe’ to the dictionary. The new key ‘shoe’ is enclosed in the red rectangle.

You can also update a key-value pair.



```
webstersDict['marathon'] = '26 mile race'
```

```
webstersDict
```

```
{'marathon': '26 mile race',
 'person': 'a human being',
 'resist': 'to remain strong against the force',
 'run': 'to move with haste; act quickly',
 'shoe': 'an external covering for the human foot'}
```

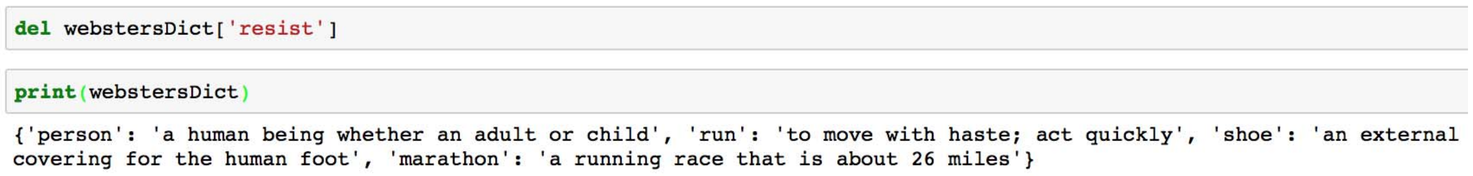
Update the dictionary key ‘marathon’

In the Dictionary Methods section, you will see that you can also add or update multiple key value pairs at a time using the dictionary **update method.**

Delete Keys from Dictionary

It is possible to remove a key and its corresponding value from a dictionary using **del**.

```
# Remove the key 'resist' from the dictionary
del webstersDict['resist']
```



```
del webstersDict['resist']
```

```
print(webstersDict)
```

```
{'person': 'a human being whether an adult or child', 'run': 'to move with haste; act quickly', 'shoe': 'an external
covering for the human foot', 'marathon': 'a running race that is about 26 miles'}
```

Remove the key ‘resist’ from the dictionary webstersDict.

In the Dictionary Methods section, you will see that you can also delete keys using the dictionary **pop method.**

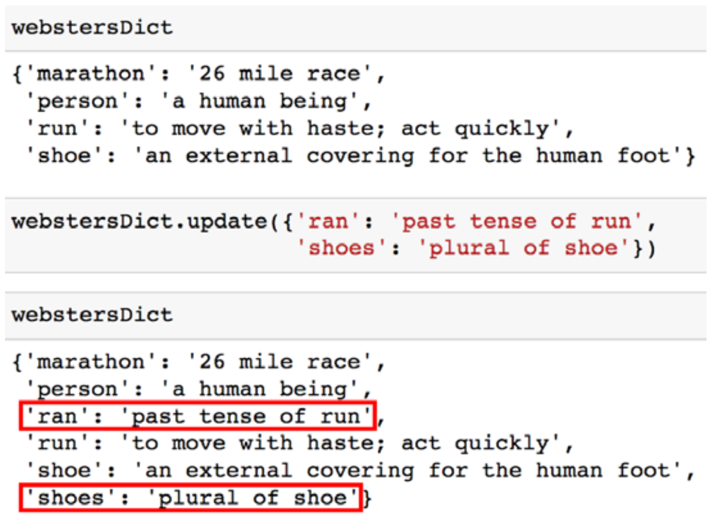
Dictionary Methods

Python dictionaries have different methods that help you modify a dictionary. This section of the tutorial just goes over various python dictionary methods.

update method

The update method is very useful for updating multiple key values pairs at a time. It takes a dictionary as an argument.

```
# Using update method to add two key value pairs at once
websterDict.update({'ran': 'past tense of run',
                    'shoes': 'plural of shoe'})
```



```
webstersDict
```

```
{'marathon': '26 mile race',
 'person': 'a human being',
 'run': 'to move with haste; act quickly',
 'shoe': 'an external covering for the human foot'}
```

```
webstersDict.update({'ran': 'past tense of run',
                     'shoes': 'plural of shoe'})
```

```
webstersDict
```

```
{'marathon': '26 mile race',
 'person': 'a human being',
 'ran': 'past tense of run',
 'run': 'to move with haste; act quickly',
 'shoe': 'an external covering for the human foot',
 'shoes': 'plural of shoe'}
```

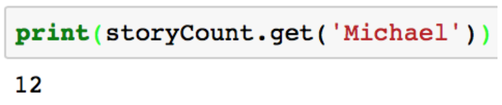
Added the keys ‘ran’ and ‘shoes’ to the dictionary.

get method

```
# Define a dictionary
storyCount = {'is': 100,
              'the': 90,
              'Michael': 12,
              'runs': 5}
```

The get method returns a value for a given key. If a key doesn’t exist, the dictionary will by default return None.

```
# Since the key 'Michael' exists, it will return the value 12
storyCount.get('Michael')
```

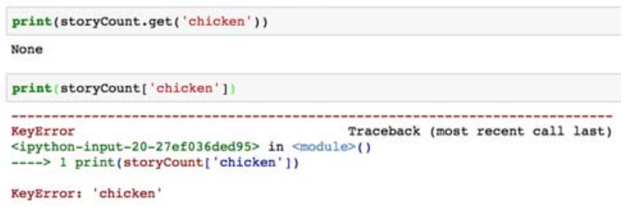


```
print(storyCount.get('Michael'))
```

```
12
```

Since the key ‘Michael’ exists, it returns the value 12. If ‘Michael’ didn’t exist, it would return None.

The method is very useful to look up keys you don’t know are in the dictionary to avoid KeyErrors.



```
print(storyCount.get('chicken'))
```

```
None
```

```
print(storyCount['chicken'])
```

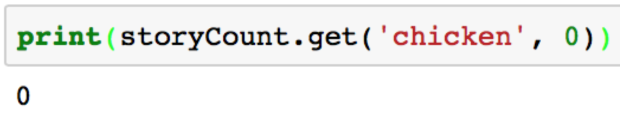
```
---------------------------------------------------------------------------
KeyError                                  Traceback (most recent call last)
<ipython-input-20-27ef036ded95> in <module>()
----> 1 print(storyCount['chicken'])

KeyError: 'chicken'
```

They key ‘chicken’ does not exist.

You can also specify a default value to return if the key doesn’t exist.

```
# Make default value for key that doesn't exist 0.
storyCount.get('chicken', 0)
```



```
print(storyCount.get('chicken', 0))
```

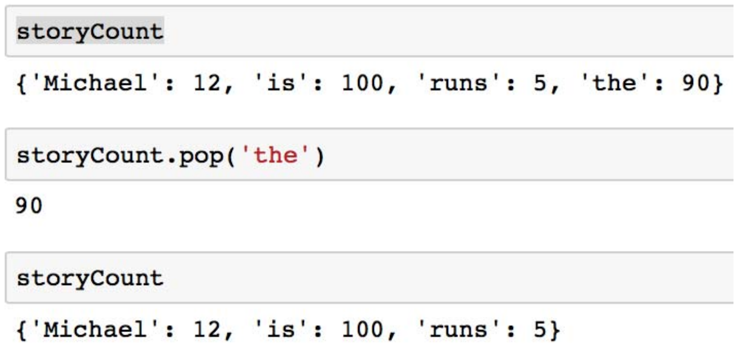
```
0
```

You can see the usefulness of this method if you try a Python Word Count.

pop Method

The pop method removes a key and returns the value.

```
storyCount.pop('the')
```



```
storyCount
```

```
{'Michael': 12, 'is': 100, 'runs': 5, 'the': 90}
```

```
storyCount.pop('the')
```

```
90
```

```
storyCount
```

```
{'Michael': 12, 'is': 100, 'runs': 5}
```

Dictionary before and after removing the key ’the’ from the dictionary.

keys Method

The keys method returns the keys of the dictionary.

```
storyCount.keys()
```

values Method

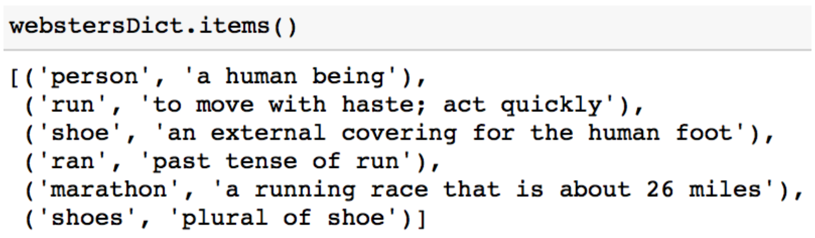
The values method returns the values in the dictionary.

```
storyCount.values()
```

items Method

The items method returns a list like object of tuples where each tuple is of the form (key, value).

```
webstersDict.items()
```



```
webstersDict.items()
```

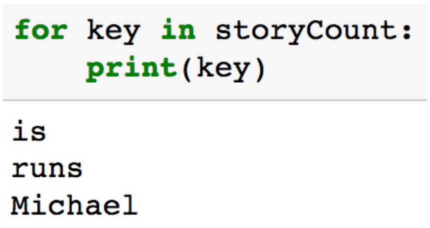
```
[('person', 'a human being'),
 ('run', 'to move with haste; act quickly'),
 ('shoe', 'an external covering for the human foot'),
 ('ran', 'past tense of run'),
 ('marathon', 'a running race that is about 26 miles'),
 ('shoes', 'plural of shoe')]
```

Iterate through a Dictionary

Two iteration methods:
1) forLoop
2) keys method

You can iterate through the keys of a dictionary by using a for loop.

```
for key in storyCount:
   print(key)
```



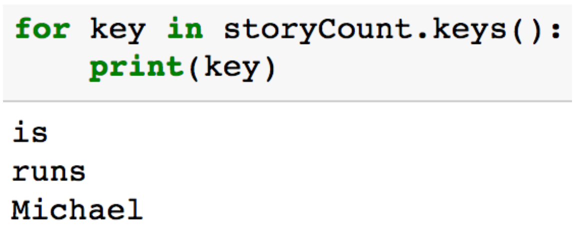
```
for key in storyCount:
    print(key)
```

```
is
runs
Michael
```

Iterate through the keys of the dictionary.

You also iterate through the keys of a dictionary by using the keys method.

```
for key in storyCount.keys():
   print(key)
```



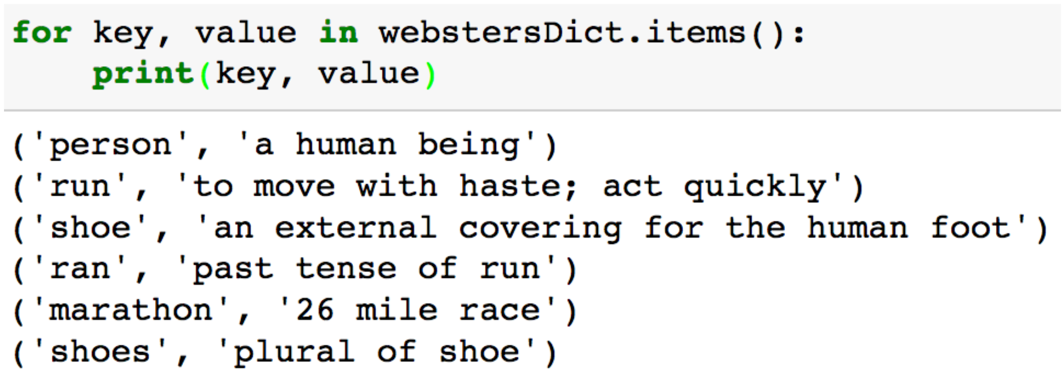
```
for key in storyCount.keys():
    print(key)
```

```
is
runs
Michael
```

Iterate through the keys of the dictionary.

The for loop below uses the items method to access one (key, value) pair on each iteration of the loop.

```
for key, value in webstersDict.items():
    print(key, value)
```



```
for key, value in webstersDict.items():
    print(key, value)
```

```
('person', 'a human being')
('run', 'to move with haste; act quickly')
('shoe', 'an external covering for the human foot')
('ran', 'past tense of run')
('marathon', '26 mile race')
('shoes', 'plural of shoe')
```

Iterate through the key, value pairs of a dictionary.

If you have difficulty understanding this section, I recommend watching the following video.

Closing Remarks

Please let me know if you have any questions either here or through Twitter! The next post, Python Word Count will review dictionary methods, list manipulations, and string manipulations. If you want to learn how to utilize the Pandas, Matplotlib, or Seaborn libraries, please consider taking my Python for Data Visualization LinkedIn Learning course. Here is a free preview video.

https://medium.com/@GalarnykMichael/python-basics-10-dictionaries-and-dictionary-methods-4e9efa70f5b9