Intro_to_dictionaries

April 12, 2018

1 Introduction to Dictionaries

In this notebook you will learn about Python Dictionaries.

1.1 Dictionaries associate `keys` with `values`

```
In [7]: ticket = {
            "date" : "2018-12-28",
            "priority" : "high",
            "description" : """Vehicle did not slow down despite
            SLOW
            SCHOOL
            ZONE    """
        }
```

The `ticket` you see above is a Python `dictionary`. A dictionary is an **unordered** data structure.

With a `list`, data is accessed by **position** (e.g. `my_list[0]`), but dictionaries are unordered. Data in a dictionary is accessed via it's **key** instead of it's position.

In the `ticket` example above, there are three **keys**. You can see what they are by running the code below

```
In [8]: print(ticket.keys())

dict_keys(['date', 'priority', 'description'])
```

Each **key** in a dictionary is associated with a **value**. The code below retrieves the **value** associated with the **key** `description`.

```
In [9]: print(ticket['description'])

Vehicle did not slow down despite
    SLOW
    SCHOOL
    ZONE
```

1.2 Dictionaries are *mutable* (they can be modified)

Dictionaries are **mutable**, which means that they can be *mutated* (modified). Being mutable means:

1. Elements can be **added** to a dictionary.
2. Elements can be **removed** from a dictionary.
3. Elements can be **modified** in a dictionary.

The code below demonstrates how to add, remove, and modify elements in a dictionary.

```
In [10]: # Let's start with an empty dictionary. Eventually this will store
         # English to spanish translations...

         eng_to_spa = {} # this creates an empty dictionary
         print(eng_to_spa)
```

```
{}
```

```
In [11]: # 1. Adding elements to a dictionary.
         #
         #    Elements can be added to a dictionary as follows:

         eng_to_spa['blue']  = 'Azul'
         eng_to_spa['gren']  = 'verde'

         print(eng_to_spa)
```

```
{'blue': 'Azul', 'gren': 'verde'}
```

Oops! I mispelled "green". Let's remove that element from the dictionary...

```
In [12]: # 2. Removing elements from a dictionary
         #
         #    Elements can be removed from a dictionary using the del keyword

         del eng_to_spa['gren']

         print(eng_to_spa)
```

```
{'blue': 'Azul'}
```

It looks like "azul" is capitalized... let's change that!

```
In [13]: # 3. Modifying elements in a dictionary.
         #
         #    Modifying the value associated with a key works just
         #    like adding a new value...
```

```
eng_to_spa['blue'] = 'azul'

print(eng_to_spa)
```

```
{'blue': 'azul'}
```

1.2.1 TODO - Complete `eng_to_spa`

In the code cell below, add elements for a few additional colors so that `eng_to_spa` reflects the following:

English	Spanish
blue	azul
green	verde
pink	rosa
orange	naranja
gray	gris
brown	marron

```
In [2]: eng_to_spa={}
        eng_to_spa['blue'] = 'azul'
        eng_to_spa['green'] = 'verde'

        # YOUR CODE HERE - complete the eng_to_spa
        #   dictionary so it contains all the information
        #   shown in the table above.
        eng_to_spa['pink']='rosa'
        eng_to_spa['orange']='naranja'
        eng_to_spa['gray']='gris'
        eng_to_spa['brown']='marron'

        # Testing code below
        assert(eng_to_spa['blue'] == 'azul')
        assert(eng_to_spa['gray'] == 'gris')
        assert(eng_to_spa['orange'] == 'naranja')
        assert(eng_to_spa['pink'] == 'rosa')
        print("Your english to spanish dictionary is looking good!")
```

```
Your english to spanish dictionary is looking good!
```

1.3 Dictionaries are *unordered*

What does it mean for a dictionary to be **unordered**? Consider the following two tickets.

```
In [3]: dictionary_ticket_1 = {
            "date"       : "2018-12-31",
```

```
        "priority"    : "high",
        "description": "Vehicle made unexpected stop."
    }

    dictionary_ticket_2 = {
        "priority"    : "high",
        "description": "Vehicle made unexpected stop.",
        "date"        : "2018-12-31"
    }

    # these dictionaries contain the same information, but the
    # ordering looks different. Does Python consider them identical?

    print("Does dictionary_ticket_1 == dictionary_ticket_2?")
    print(dictionary_ticket_1 == dictionary_ticket_2)
```

```
Does dictionary_ticket_1 == dictionary_ticket_2?
True
```

1.3.1 Looping through a dictionary

Despite not having a well-defined ordering of its elements, you can still loop through the **keys** of a dictionary…

```
In [5]: # demonstration of dictionary looping
        # demo 1 - keys only

        for i in dictionary_ticket_1:
            print(i)
```

```
date
priority
description
```

```
In [6]: # demonstration of dictionary looping
        # demo 2 - keys and values

        for key in dictionary_ticket_1:
            value = dictionary_ticket_1[key]
            print(key, ':', value)
```

```
date : 2018-12-31
priority : high
description : Vehicle made unexpected stop.
```

1.4 TODO - Exercise: "Reverse" a dictionary

The code below re-defines the `eng_to_spa` dictionary. Your job is to write a function called `reverse_dictionary` which takes, for example, an english-to-spanish dictionary and returns a spanish-to-english dictionary.

```
In [10]: def reverse_dictionary(dictionary):
             new_d = {}
             # TODO - your code here
             for i in dictionary:
                 new_d[dictionary[i]]=i
             return new_d

         eng_to_spa = {
             "red" :    "rojo",
             "blue"   : "azul",
             "green"  : "verde",
             "black"  : "negro",
             "white"  : "blanco",
             "yellow" : "amarillo",
             "orange" : "naranja",
             "pink"   : "rosa",
             "purple" : "morado",
             "gray"   : "gris"
         }

         # TESTING CODE

         spa_to_eng = reverse_dictionary(eng_to_spa)

         assert(len(spa_to_eng) == len(eng_to_spa))
         assert(spa_to_eng['rojo'] == 'red')
         assert(spa_to_eng['azul'] == 'blue')
         assert(spa_to_eng['verde'] == 'green')

         print("Nice work! Your reverse_dictionary function is correct.")
```

```
Nice work! Your reverse_dictionary function is correct.
```

1.5 Spanish to French

In this exercise you will take two dictionaries (one spanish-to-english and one french-to-english) and combine them to make a single spanish-to-french dictionary.

First, let's take a look at the two existing dictionaries.

```
In [11]: eng_to_spa = {
             "red" :    "rojo",
             "blue"   : "azul",
```

```
    "green"  : "verde",
    "black"  : "negro",
    "white"  : "blanco",
    "yellow" : "amarillo",
    "orange" : "naranja",
    "pink"   : "rosa",
    "purple" : "morado",
    "gray"   : "gris"
}

french_to_eng = {
    "bleu"  : "blue",
    "noir"  : "black",
    "vert"  : "green",
    "violet": "purple",
    "gris"  : "gray",
    "rouge" : "red",
    "orange": "orange",
    "rose"  : "pink",
    "marron": "brown",
    "jaune" : "yellow",
    "blanc" : "white",
}

print("there are    ", len(eng_to_spa), "colors in eng_to_spa")
print("but there are", len(french_to_eng), "colors in french_to_eng")
```

```
there are     10 colors in eng_to_spa
but there are 11 colors in french_to_eng
```

In [12]:
```
# don't forget you have the "reverse_dictionary" function!

english_to_french = reverse_dictionary(french_to_eng)

for english_word in english_to_french:
    french_word = english_to_french[english_word]
    print("The french word for", english_word, "is", french_word)
```

```
The french word for blue is bleu
The french word for black is noir
The french word for green is vert
The french word for purple is violet
The french word for gray is gris
The french word for red is rouge
The french word for orange is orange
The french word for pink is rose
The french word for brown is marron
```

```
The french word for yellow is jaune
The french word for white is blanc
```

```
In [18]: def spanish_to_french(english_to_spanish, english_to_french):
             """
             Given an English to Spanish dictionary and an English
             to French dictionary, returns a Spanish to French dictionary.

             If any words appear in one dictionary but NOT the other,
             they should not be included in the resulting dictionary.
             """
             s2f = {}
             #
             # TODO - your code here
             #

             for i in english_to_spanish:
                 if i in english_to_french:
                     s2f[english_to_spanish[i]]=english_to_french[i]

             """
             for english_word in english_to_french:
                 if english_word in english_to_spanish:
                     french_word = english_to_french[english_word]
                     spanish_word = english_to_spanish[english_word]
                     s2f[spanish_word] = french_word
             """

             return s2f


         # TESTING CODE
         S2F = spanish_to_french(eng_to_spa, english_to_french)

         assert(S2F["rojo"] == "rouge")
         assert(S2F["morado"] == "violet")

         print("Nice work! Your spanish to french function works correctly!")
```

```
Nice work! Your spanish to french function works correctly!
```

1.5.1 My Solution

Make sure you've attempted to solve the problem above before continuing!

When I tested my first solution, I ran into an error. Try running my first attempt (below) to see what it was

```
In [ ]: def spanish_to_french_1(english_to_spanish, english_to_french):
            s2f = {}
            for english_word in english_to_french:
                french_word = english_to_french[english_word]
                spanish_word = english_to_spanish[english_word]
                s2f[spanish_word] = french_word
            return s2f

        S2F_1 = spanish_to_french_1(eng_to_spa, english_to_french)
```

I keep getting the following message:

```
KeyError: 'brown'
```

That's because "brown" isn't in my Spanish to English dictionary!

That's okay, I can just do a **membership check** on each key. Compare the code above to the code below.

```
In [19]: def spanish_to_french_2(english_to_spanish, english_to_french):
             s2f = {}
             for english_word in english_to_french:
                 if english_word in english_to_spanish:
                     french_word = english_to_french[english_word]
                     spanish_word = english_to_spanish[english_word]
                     s2f[spanish_word] = french_word
             return s2f

         S2F_2 = spanish_to_french_2(eng_to_spa, english_to_french)
```

1.5.2 A note on the `in` keyword (and "testing for membership")

Pay attention to the if statement in the code above. When we write

```
if X in Y
```

we are "testing for membership". When `Y` is a `dictionary` we are testing to see if `X` is a **key** in that dictionary. If it is, the test returns `True`. Otherwise it returns `False`.

```
In [20]: capitals = {
             "spain" : "madrid",
             "france" : "paris",
             "china" : "beijing"
         }

         print("Is 'china' a key in this dictionary?")
         print("china" in capitals)
         print()

         print("Is 'paris' a key in this dictionary?")
         print("paris" in capitals)
```

```
Is 'china' a key in this dictionary?
True

Is 'paris' a key in this dictionary?
False
```

1.6 Dictionary Summary

Important things to remember about dictionaries!

1. Dictionaries associate `keys` with `values` The following dictionary has 2 keys (`"abc"` and `"def"`) and 2 values (`123` and `456`)

```
d = {"abc":123, "def":456}
```

2. Dictionaries are Unordered Two dictionaries are identical if they have the same keys **and** those keys are associated with the same values. Order doesn't matter.

```
> d1 = {"abc":123, "def":456}
> d2 = {"def":456, "abc":123}
> d1 == d2
True
```

3. Dictionaries are "mutable" This means they can be modified in various ways

3.1 adding new elements

```
> d = {}
> d['k'] = 'v'
> print(d)
{'k': 'v'}
```

3.2 removing elements

```
> del d['k']
> print(d)
{}
```

3.3 changing elements

```
> d['key'] = 'value' # first need to add an element back in
> print(d)
{'key': 'value'}

> d['key'] = 'other value'
> print(d)
{'key' : 'other value'}
```

4. Looping through a dictionary When looping through a dictionary, you loop through the **keys** of that dictionary.

5. Membership Testing Python's `in` keyword can be used to test if something is a **key** in a dictionary.