Last week, you developed a language translation program that established an association between two lists of words. One way to do this is to have two lists x1 and x2 and use the index in one to lookup the corresponding word in the other: **x1[x2.index(word)]** (or vice versa). This technique works because the words are appended in each list in order. like this …

For example, **x1 = ['one','two','three']** and **x2 = ['uno','dos','tres']** .

But what if we could simply write something like: **print(english['uno'])** and get **'one'** ? That's something that we can do using Python dictionaries. We can use dictionaries to map one language into another one. That's not exactly what I'm going to demonstrate below. I leave that simpler exercise up to you. Your assignment is to create a new version of the language translation program you created last week that uses dictionaries instead of lists. Below, I'm going to demonstrate dictionaries with keys that associate with other dictionaries. These dictionaries demonstrate a way to translate from English to multiple other languages. Be sure to read the Starship Example document first. As with all of these code demonstrations, the best way to read them is to be active … have a Python shell open and try out the things I demonstrate.

Let's say we want to translate from English to some other language using dictionaries. One way to do this would be to establish a dictionary where we provide both an English word and another language as keys so the dictionary can return the English word translated into another language.

For example, after executing the statement:

**red\_in\_french = translate['red']['french']**, then

**print(red\_in\_french)** would print '**rouge**'.

Here, the color in English ('red') is a key paired with another dictionary and the language (**'**french**'**) is a key in the dictionary referred to by the 'red' key. The key-value pair for red would look like this:

**'red' : {'french':'rouge'}**

As we add more languages, the dictionary associated with red grows with more key-value pairs …

Adding spanish --> **'red' : {'french':'rouge', 'spanish':'rojo'}** <-- add spanish

Adding german --> **'red' : {'french':'rouge', 'spanish':'rojo','german':'rot'}**

How would the source code look?

Machine generated alternative text:
colors 
colors [ ' red' 
colors [ ' red' 
colors [ ' red' 
colors [ ' red' 
print (colors) 
empty dictionary 
{ } # add key ' red' pairs 
. update ( { ' french ' 
' rouge' } ) 
. update ( { ' spanish ' 
' rojo' 
: 'rot'} ) # 
. update ( { ' german ' 
with empty dictionary 
# update with French 
# update with Spanish 
update with German 

Machine generated alternative text:
{' red' : 
{ ' french' : 
' rouge' , 
' spanish' : 
' rojo', 
' german ' • 
' rot' 

Now that we have one color, we have the recipe for adding other colors like green …

Machine generated alternative text:
colors 
colors [ ' red' 
colors [ ' red' 
colors [ ' red' 
colors [ ' red' 
empty dictionary 
{ } # add key ' red' pairs 
. update ( { ' french ' 
. update ( { ' spanish ' 
. update ( { ' german ' 
{ } # add key 
colors C 'green'] . update ( { ' french ' 
colors C ' green'] . update ( { ' spanish ' 
colors C ' green . update ( { ' german ' 
' rouge' } ) 
' rojo' 
: 'rot'} ) # 
with empty dictionary 
# update with French 
# update with Spanish 
update with German 
add green 
colors C 'green ' 
print (' red' 
colors C ' red'] ) 
print ('green ' 
colors 'green'] ) 
' green' pairs with empty dictionary 
' vert ' } ) # update with French 
' verde ' } ) # update with Spanish 
} ) # update with German 
' grun ' 

Machine generated alternative text:
red { 
green 
' french' : 
' rouge' , 
{ ' french' : 
'vert', 
' spanish' : 
' spanish' : 
' rojo', 
' verde ' , 
' german ' • 
' german ' • 
' rot'} 
' grun ' 

Now add yellow and blue …

Machine generated alternative text:
colors 
colors [ ' red' 
colors [ ' red' 
colors [ ' red' 
colors [ ' red' 
empty dictionary 
{ } # add key ' red' pairs 
. update ( { ' french ' 
' rouge' } ) 
. update ( { ' spanish ' 
' rojo' 
. 'rot'} ) # 
. update ( { ' german ' 
with empty dictionary 
# update with French 
# update with Spanish 
update with German 
add green 
colors [ 'green ' 
colors [ 'green ' 
colors [ 'green ' 
colors [ 'green ' 
add yellow 
colors C ' yellow' 
colors C ' yellow' 
colors C ' yellow' 
colors C ' yellow' 
{ } # add key 
' green' pairs with empty dictionary 
. update ( { ' french' : 'vert ' } ) # update with French 
. update ( { ' spanish' : 'verde ' } ) # update with Spanish 
} ) # update with German 
. update ( { ' german ' 
' grun ' 
{ } # add key 'yellow' pairs with empty dictionary 
' jaune ' 
} ) # update with French 
. update ( { ' french ' 
. update ( { ' spanish ' 
' amarillo' } ) # update with Spanish 
' gel b' } ) # update with German 
. update ( { ' german ' 
{ } # add key 'yellow' pairs with empty dictionary 
add blue 
colors 'blue ' 
colors 'blue ' 
colors 'blue ' 
colors 'blue ' 
# now we can 
in 
for 
color 
. update ( { ' french ' 
. update ( { ' spanish ' 
. update ( { ' german ' 
iterate through the 
colors . keys ( ) 
} ) # update with French 
'bleu' 
' azu1 ' } ) # update with Spanish 
} ) # update with German 
'blau' 
keys to print 
print (color, colors [color] ) 

Note how we use a for loop to iterate through the keys (above) to print the colors …

Machine generated alternative text:
red { ' french' : 
green { ' french' : 
yellow { ' french' : 
blue { ' french' : 
' rouge' , 
'vert', 
' jaune ' 
'bleu', 
' spanish' : 
' spanish' : 
' rojo', 
' verde ' , 
' german ' • 
' german ' • 
' rot'} 
' grun ' 
' spanish' : 
' amari110 ' 
' german ' • 
'blau'} 
' gel b ' 
' spanish' : 
'azul', 
' german ' • 

So, now we've created a multilingual dictionary. See the tests below.

Machine generated alternative text:
>>> colors [ ' red'] C 'german ' ] 
' rot' 
>>> colors [ ' yellow'] [' french'] 
' jaune ' 
>>> colors 'blue'] [ ' spanish' 
' azul' 
>>> colors 'green'] C' french ' 
' vert ' 
>>> 

Now we can create a user interface to do the translation. Prompt the user to obtain a color in English and a language to translate into. Then do the translation …

This is a test run with a simple user interface. Your project will emulate the interface from last week.

Machine generated alternative text:
RESTART: C: / Users/ kb au/Documents/msitCode/coIors 
Language Translator 
Available English words are: blue; green; yellow; red; 
Please enter a color in English: blue 
Available language translations are: german; french; 
Please enter a language from the list: spanish 
The word "blue" in Spanish is "azul" . 
>>> 
spanish; 

Now you have seen how dictionaries can be used to make a simple multilingual translator. Of course, there are problems. The dictionaries use hard-coded data. In general, we would want to be able to expand the vocabulary as you did last week. The same problem will persist … any new vocabulary will be lost when the program is closed. Next week, we’ll learn how to use text files to implement vocabulary that will persist after the program closes.