

All Code Clubs must be registered. Registered clubs appear on the map at codeclubworld.org - if your club is not on the map then visit jump.to/cc/18CpLPy to find out what to do.

Introduction:

In this project you'll learn how to make a program for translating text-speak into sentences.



Activity Checklist

Follow these **INSTRUCTIONS** one by one



Test your Project

Click on the green flag to **TEST** your code



Save your Project

Make sure to **SAVE** your work now

Step 1: Translating words

Let's make a program to convert text-speak to English.

Activity Checklist

1. As you probably already know, a dictionary allows you to look up a word, and see it's meaning. In Python, a dictionary is even more flexible than that - it allows you to map anything (called a key) to anything else! Here's a dictionary that links text-speak words to their meaning: ☐

```
textSpeakDictionary = {  
    "lol"    : "laugh out loud" ,  
    "idk"    : "I don't know"  
}
```

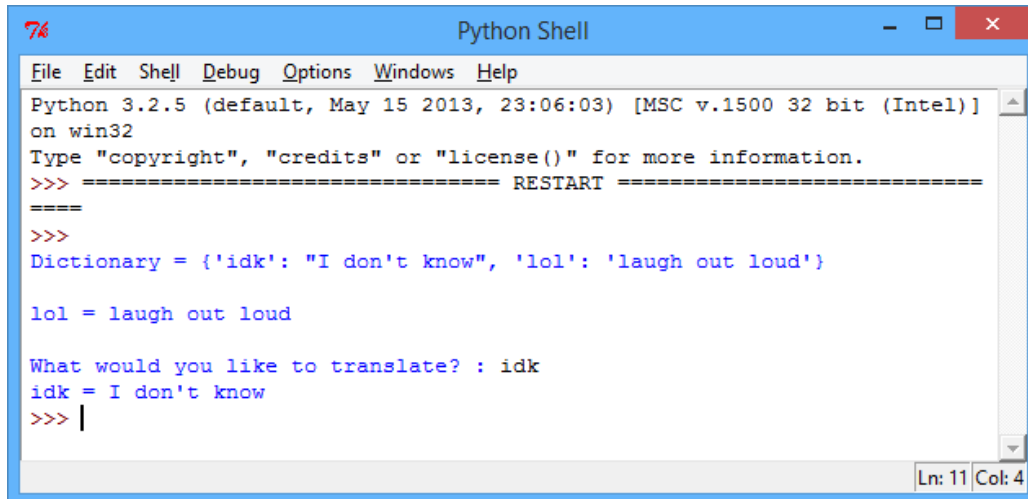
So, in the dictionary above, the key "lol" maps to the text "laugh out loud", and the key "idk" maps to the text "I don't know". You should use a colon (:) to map the text-speak keys to their meanings, and put a comma between each dictionary entry.

2. Getting information out of the dictionary is easy; you just need to add the key after the dictionary variable name, in square brackets. Here's a short program that shows how this works: ☐

```
textSpeakDictionary = {  
    "lol"    : "laugh out loud" ,  
    "idk"    : "I don't know"  
}  
  
#print the entire dictionary  
print( "Dictionary =" , textSpeakDictionary )  
  
#print just the entry for "lol"
```

```
print( "\nlol =" , textSpeakDictionary["lol"] )

#the entry for the user's input
key = input("\nWhat would you like to translate? : ")
print( key , "=" , textSpeakDictionary[key] )
```



```
Python 3.2.5 (default, May 15 2013, 23:06:03) [MSC v.1500 32 bit (Intel)]
on win32
Type "copyright", "credits" or "license()" for more information.
>>> ===== RESTART =====
>>>
>>> Dictionary = {'idk': "I don't know", 'lol': 'laugh out loud'}

lol = laugh out loud

What would you like to translate? : idk
idk = I don't know
>>> |
```

This program prints 3 things: the entire dictionary, the dictionary entry for “lol” and finally the dictionary entry for whatever the user inputs.



Save Your Project

Step 2: Translating sentences

Let's amend your program, so that you can translate whole sentences instead of just single words.



Activity Checklist

1. Run this program, which splits up a sentence into individual words, and then translates each word (if it exists in the dictionary):



```

textSpeakDictionary = {
    "lol"    : "laugh out loud" ,
    "idk"    : "I don't know"
}

#get the sentence to translate
sentence = input("Enter a sentence to translate:
").lower()

#this splits up the sentence into a list of words
wordsToTranslate = sentence.split()

translatedSentence = ""

#loop through each word in the list
for word in wordsToTranslate:

    #add the translated word if it exists in the
    dictionary
    if word in textSpeakDictionary:

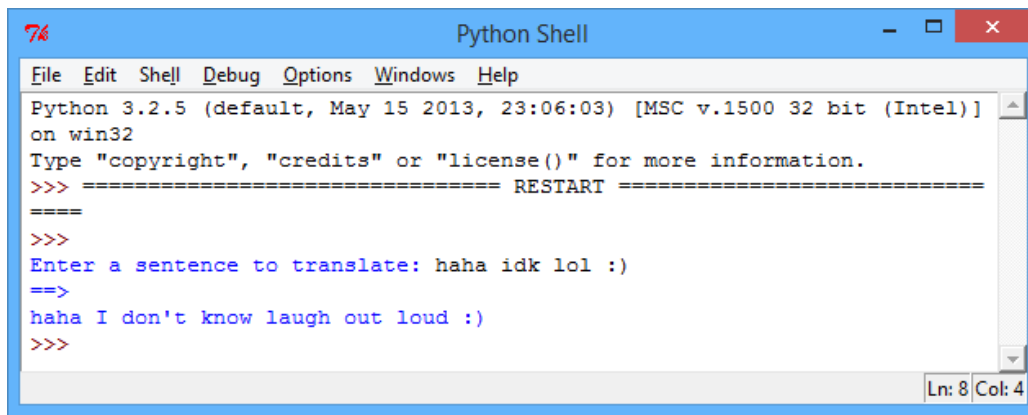
        translatedSentence += textSpeakDictionary[word]
    + " "

    #just keep the original word if there's no
    translation
    else:

        translatedSentence += word + " "

#print the translated sentence
print("==>")
print(translatedSentence)

```



```
Python 3.2.5 (default, May 15 2013, 23:06:03) [MSC v.1500 32 bit (Intel)]
on win32
Type "copyright", "credits" or "license()" for more information.
>>> ===== RESTART =====
>>>
Enter a sentence to translate: haha idk lol :)
==>
haha I don't know laugh out loud :)
>>>
```

Each word is taken in turn, and the program checks whether the word to translate is in the dictionary. If it is, then the translated text added to the `translatedSentence` variable, which is printed at the end of the program. If the word isn't in the dictionary, then just the original word is added to the `translatedSentence` variable.

Notice that whenever a word is added to `translatedSentence`, a space is also added (`+ " "`). What do you think would happen if this space wasn't added?

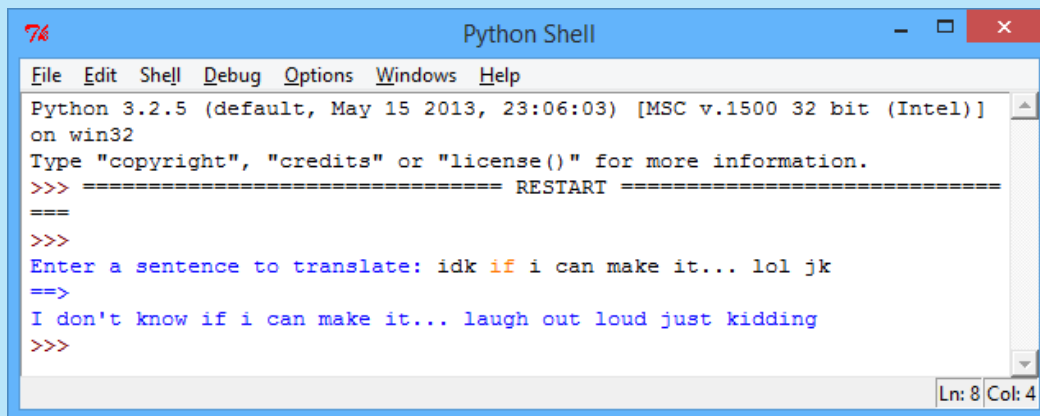
Challenge: Adding translations

- ☐ Add some more translations to the program above. For example:

- "jk" = "just kidding"
- "bc" = "because"

You might need to research some text-speak if you don't know any.

- ☐ Try out the program above, with a number of different sentences, to test that your program works.



```
Python Shell
File Edit Shell Debug Options Windows Help
Python 3.2.5 (default, May 15 2013, 23:06:03) [MSC v.1500 32 bit (Intel)]
on win32
Type "copyright", "credits" or "license()" for more information.
>>> ===== RESTART =====
>>>
Enter a sentence to translate: idk if i can make it... lol jk
==>
I don't know if i can make it... laugh out loud just kidding
>>>
```

- ☐ Did you (or your friends) do anything to break your program? If so, can you fix any problems?

Step 3: Adding and deleting translations

✓ Activity Checklist

1. Just like with your 'compliment generator' program, it would be nice to allow the user to add and remove words from the dictionary. You can do this, by creating a menu system:



```
def displayMenu():
    print("txt spk cnvtr")
    print("=" * 13)
```

```

    print("Menu:")
    print("  c = convert a sentence")
    print("  p = print dictionary")
    print("  a = add a word")
    print("  d = delete a word")
    print("  q = quit")

#-----
-

def convertSentence():
    sentence = input("Enter a sentence to translate:
").lower()
    translatedSentence = ""

    #this splits up the sentence into a list of words
    listOfWords = sentence.split()

    for word in listOfWords:
        #add the translated word if it exists in the
dictionary
        if word in textSpeakDictionary:

            translatedSentence +=
textSpeakDictionary[word] + " "

        #just keep the original word if there's no
translation
        else:

            translatedSentence += word + " "

    #print the translated sentence
    print("==>")
    print(translatedSentence)

#-----
-

def addDictionaryItem():
    txtToAdd = input("Enter the text-speak to add to

```

```

the dictionary: ")
    meaning = input("What does this mean?: ")
    #add the new translation to the dictionary
    textSpeakDictionary[txtToAdd] = meaning

#-----
-

def deleteDictionaryItem():
    txtToDelete = input("Enter the text-speak to delete
from the dictionary: ")
    #delete the translation from the dictionary
    del textSpeakDictionary[txtToDelete]

#-----
-

# main program starts here!
#-----
-

textSpeakDictionary = {
    "lol" : "laugh out loud" ,
    "idk" : "I don't know" ,
    "jk"  : "just kidding" ,
    "bc"  : "because"
}

running = True

displayMenu()

#repeat until the user inputs 'q' to quit
while running == True:

    menuChoice = input(">_").lower()

    #c to convert
    if menuChoice == 'c':
        convertSentence()

    #p to print

```



```

elif menuChoice == 'p':
    print(textSpeakDictionary)

#a to add
elif menuChoice == 'a':
    addDictionaryItem()

#d to delete
elif menuChoice == 'd':
    deleteDictionaryItem()

#q to quit
elif menuChoice == 'q':
    running = False

else:
    print("Invalid menu choice!")

```

```

Python 3.2.5 (default, May 15 2013, 23:06:03) [MSC v.1500 32 bit (Intel)] on win32
Type "copyright", "credits" or "license()" for more information.
>>> ===== RESTART =====
>>>
txt spk cnvtr
=====
Menu:
  c = convert a sentence
  p = print dictionary
  a = add a word
  d = delete a word
  q = quit
>_a
Enter the text-speak to add to the dictionary: rofl
What does this mean?: rolling on the floor laughing
>_p
{'rofl': 'rolling on the floor laughing', 'bc': 'because', 'idk': 'I don't know',
'lol': 'laugh out loud', 'jk': 'just kidding'}
>_c
Enter a sentence to translate: haha idk rofl
==>
haha I don't know rolling on the floor laughing
>_q
>>>
Ln: 23 Col: 4

```

Although this is a loooong program, you've seen most of this code before in other programs. The new bits are just the code to add an item to the dictionary:

```
txtToAdd = input("Enter the text-speak to add to the
```

```
dictionary: ")
meaning = input("What does this mean?: ")
#add the new translation to the dictionary
textSpeakDictionary[txtToAdd] = meaning
```

...and the code to remove an item:

```
txtToDelete = input("Enter the text-speak to delete
from the dictionary: ")
#delete the translation from the dictionary
del textSpeakDictionary[txtToDelete]
```

The code for each of the menu options is also in it's own function, to make the code much easier to read.



Save Your Project

Challenge: Testing your program

Run your program, and try to add a word that already exists in the dictionary. What happens? What happens when you try and remove something that isn't in the dictionary? Can you improve your program so that:

- ☐ you can only add dictionary keys that don't already exist?

```
if itemToAdd not in textSpeakDictionary:  
    #Add your code here!
```

- ☐ you can only delete keys if they already exist in the dictionary?

```
if itemToDelete not in textSpeakDictionary:  
    #Add your code here!
```



Save Your Project

Step 4: Fixing your program



Activity Checklist

1. You've already done lots of testing to improve your program, but there's one more thing that you can fix, to make your program even better. Look what happens when you test your program with the following sentence:



```
> _c
Enter a sentence to translate: hello lol!
==>
hello lol!
> _
```

Ln: 17 Col: 2

It doesn't get translated properly. Try it out for yourself.

2. Why doesn't your program convert the 'lol' in this sentence?
It's because your program splits up the sentence into words, like this:

```
words = [ "hello" , "lol!" ]
```

It then looks up the key `"lol!"` in your dictionary (with the exclamation mark), and can't find a translation for it, because `"lol!"` doesn't exist! One simple way to avoid this problem is to remove some punctuation from the sentence before it is translated. Add this code to your `convertSentence()` function:

```
def convertSentence():
    sentence = input("Enter a sentence to translate: ")
    sentence = sentence.lower()
    translatedSentence = ""

    #take out some punctuation from the sentence
    for char in '?!.,':
        sentence = sentence.replace(char, '')

    #this splits up the sentence into a list of words
    listOfWords = sentence.split()
    ...
```

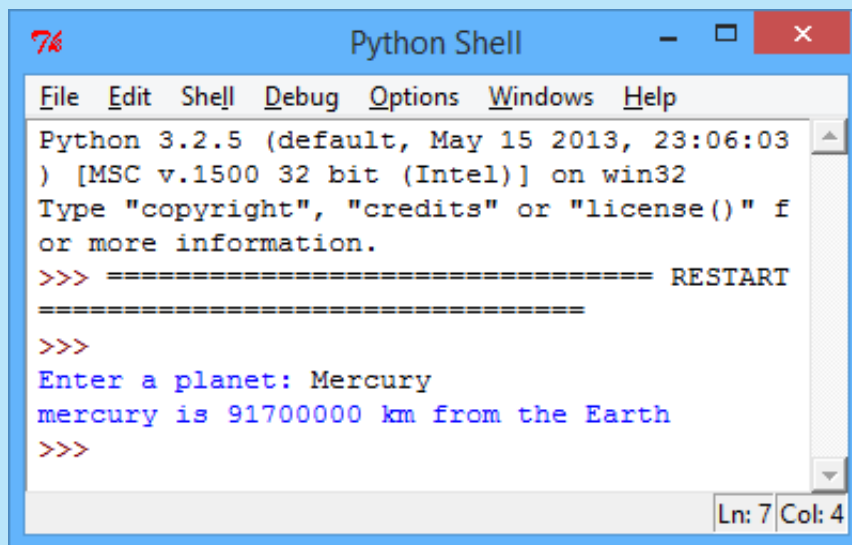
This extra code loops through each of the punctuation marks `?!.,` in turn, and replaces them in the sentence with... nothing! This removes the punctuation from the sentence.

3. After adding in this code to take out the punctuation, try translating `"hello, lol!"` again, to check whether you've solved the problem.



Challenge: Distant planets

Make a program to give the user information about any topic you like. For example planets, and their distances from the Earth. You can store this data in a dictionary that links planets and distances.



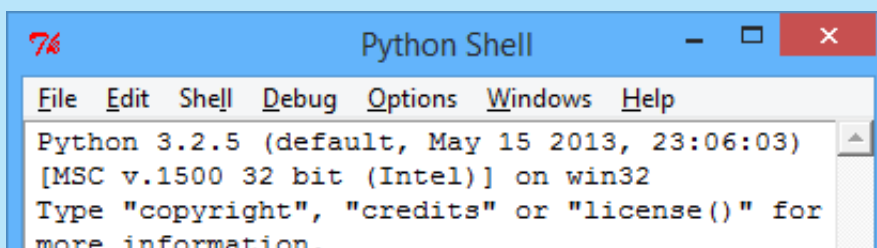
```
Python Shell
File Edit Shell Debug Options Windows Help
Python 3.2.5 (default, May 15 2013, 23:06:03)
[MSC v.1500 32 bit (Intel)] on win32
Type "copyright", "credits" or "license()" for
or more information.
>>> ===== RESTART
>>>
Enter a planet: Mercury
mercury is 91700000 km from the Earth
>>>
```



Save Your Project

Challenge: Password protection

Create a password-protection program, that asks the user for their name and password, and checks a dictionary to see if they've entered the correct details.



```
Python Shell
File Edit Shell Debug Options Windows Help
Python 3.2.5 (default, May 15 2013, 23:06:03)
[MSC v.1500 32 bit (Intel)] on win32
Type "copyright", "credits" or "license()" for
more information.
```

```

>>> ===== RESTART =====
>>>
Super-secret program
=====
Name : sohail
Password : letmein

WELCOME SOHAIL
>>>
Ln: 11 Col: 4

```

Your program will need to check that the user's name exists in the dictionary, and that the correct password for that user has been entered. You can use this code to help you:

```

#check that the name exists, and that the password is
correct
if name in passwordDictionary and password ==
passwordDictionary[name]:
    #add code here!

```

Make sure that your program works, by testing what happens when the user enters valid and invalid names and passwords.

If you feel like it, you could also:

- ☐ add this login code to one of the programs you've already created, so that the program can only be used by your friends.
- ☐ only allow the user 3 attempts to login..., adding 1 to `loginAttempts` whenever access is denied.

```

loginAttempts = 0
while loginAttempts < 3:
    #login code goes here!

```

- ☐ You could even use what you've learnt about dictionaries to create a program for storing your friend's email

to create a program for storing your friends' email addresses, or translating text from one language to another. You could even password-protect this program to make it secure!



Save Your Project