Name: Date:

Lesson 5: Desert Island Dictionary

Dictionaries

In this lesson we're going to learn about one more very important Python type called `dict` which represents a **dictionary**. This is a data structure that has **keys** and **values** just like entries in a real dictionary where the key is the word and the value is the definition of the word. Most programming languages have a dictionary data type – sometimes it's called an *associative array*. An empty Python dictionary is created with braces `{}`. You can also create a dictionary with key-value pairs.

Here is some code to create a dictionary with two keys `'animal'` and `'vegetable'`:

```
>>> d={'animal':'monkey','vegetable':'carrot'}
>>> d.keys()
['animal','vegetable']
>>> d.values()
['monkey','carrot']
```

Keys can be strings or integers. Values can be strings, integers, lists OR even other dictionaries. When you're writing code that operates on complex data, you will spend a lot of time looking at how to best structure that data so you can work with it efficiently. In order to do that it's very important to be really comfortable with the main data types. In Python they are: `str`, `int`, `float`, `list`, `bool`, `dict` which correspond to string, integer, decimal number, list, boolean (True or False) and dictionary data types.

TRY IT OUT #1: Try creating a Python dictionary at the interpreter prompt. It can be be built of any combination of other Python data types you like. List the keys and values.

Let's try creating a dictionary that holds a list of sports as strings and the values are a score associated with that sport. Remember how we used for loops and list comprehensions to iterate every value in a list? We can likewise use a for loop to iterate the dictionary built around a dictionary `.items()` method:

```
>>> d={'football':10,'cricket':5,'rugby':9,'hockey':8}
>>> for k,v in d.items():
...         print(k,v)
('football',10)
('cricket',5)
('rugby',9)
('hockey',8)
```

TRY IT OUT #2: Try iterating over the dictionary you already developed with a `for` loop. How would you rewrite it to print out the reversed value and key? So for instance instead of `('football',10)` in the above example you are printing out `(10,'football')`.

Simple substitution ciphers

A simple substitution cipher is a method of encoding plain text where every letter is replaced with a different character. The Caesar cipher is an example of a substitution cipher where each letter is shifted by an offset. Many simple substitution ciphers have been used throughout history often using strange symbols as in the case of the **Pigpen cipher**. In the days before computers these sort of ciphers were harder to break. Today armed with your Python knowledge, any simple substitution cipher is easily breakable as we will explore in the future.

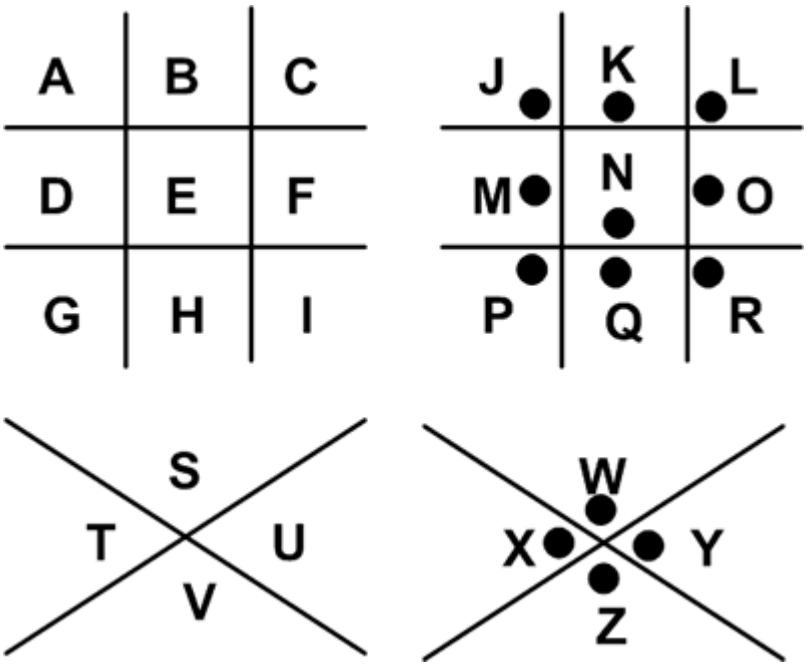
The Pigpen cipher

The Pigpen cipher was used for hundreds of years by the Freemasons for encoding secret message through handwriting. The individual letters of the alphabet are replaced with special symbols as highlighted below. You need to install the "Masonic Cipher" font to view these characters in Word.

A B C D E F G H I J K L M N O P Q R S T U V W X Y Z

A B C D E F G H I J K L M N O P Q R S T U V W X Y Z

The characters are formed from the shapes around the letters in a special arrangement:



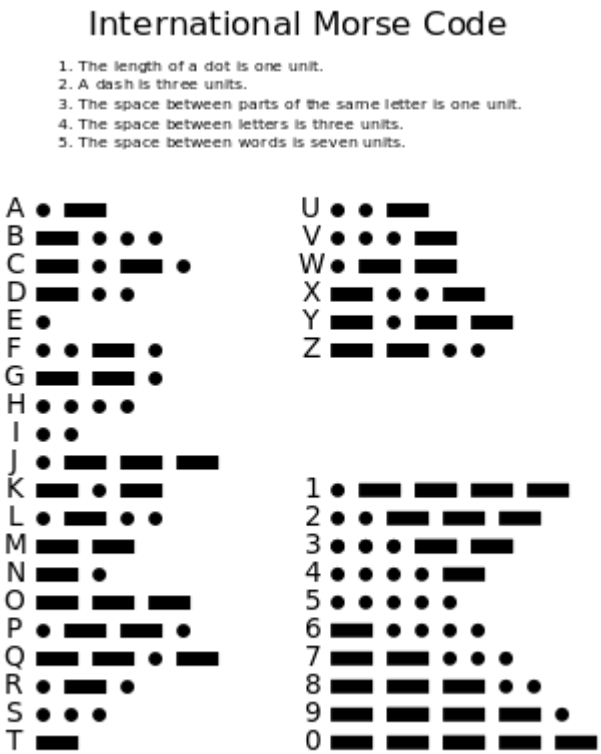
Pigpen character formation grids

Pigpen cipher ring

TRY IT OUT #3: Try writing out the string “hello world” in Pigpen.

Morse Code

Morse code is another example of a simple substitution cipher where text is converted to dots and dashes. It was developed in 1836 and became popular as a system for transmission of messages over the US telegraph system. During WWII Morse code was used to transmit encoded messages via radio telegraphy.



In International Morse Code, the emergency message "SOS" is encoded as "dot dot dot dash dash dash dot dot dot". This was the message transmitted from the Titanic as it was sinking. Here is some code showing how you can use a dictionary in Python to encode 'SOS':

```
morse={'S':'...', 'O':'---'}
s="SOS"
output=' '.join([morse[c] for c in s])
print("%s encodes to '%s'" % (s,output))

---- OUTPUT ----
SOS encodes to '... --- ...'
```

Bringing it all together ….

We can now bring together some of the elements you've learnt about this week to write some code that will convert English to Morse code. The underlying pattern is the same – call some transformation function on each character in string `s` and return the resulting output which in this case will be Morse code representation in ASCII dots and dashes.

SECRET CODE QUEST!

Write an `encodeToMorse()` function that uses a dictionary and list comprehension to convert every character in an English sentence to Morse code. You will need to take care that a space is left between each Morse code character. Can you think why? Can you also think how we could write a `decodeFromMorse()` function?

```
>>> encodeToMorse("Good Luck Boys!”)
' --. --- --- -.. .-.. ..- -.-. -.- -... --- -.-- ... '
```

```
morse={'.-':'a','-...':'b','-.-.':'c','-..':'d','.':'e','..-
.':'f','--.':'g','....':'h','..':'i','.---':'j','-.-':'k','.-
..':'l','--':'m','-.':'n','---':'o','.--.':'p','--.-':'q','.-
.':'r','...':'s','-':'t','..-':'u','...-':'v','.--':'w','-..-
':'x','-.--':'y','--..':'z','.----':'1','..---':'2',
'...--':'3','....-':'4','.....':'5','-....':'6','--...':'7','---
..':'8','----.':'9','-----':'0',' ':' '}

def decodeFromMorse(message):
    codes=message.split()
    english=''
    for code in codes:
        english+=morse.get(code)
    return english

def encodeToMorse(s):
    dico={}
    message=''
    for k,v in morse.iteritems():
        dico[v]=k
    for c in s:
        morseCharacter=dico.get(c.lower())
        if morseCharacter:
            message+=morseCharacter
            message+=' '
        else:
            message+=''
    return message

# Everything under here is the main code which uses the functions
above
if __name__=='__main__':
    #print morse
    message1='...  ---  ...  -...  ---  .- -  ...  .-  -.  -.-  -
--  -.  ..  ...  .-..  .-  -.  -..'
    s=decodeFromMorse(message1)
    print s
    message2='-  ....  .-  -  ...  .--  ....  -.--  -.--  ---
..-  .-  .-.  .  --  -.--  -...  .  ...  -  ..-.  .-.  ..  .  -.
-..'
    s=decodeFromMorse(message2)
    print s
    message3='Good luck boys!'
    s=encodeToMorse(message3)
    r=decodeFromMorse(s)
    print s
    print r
```