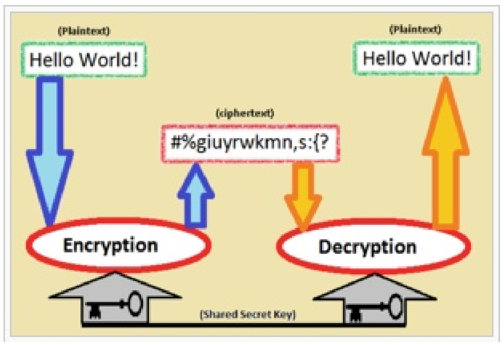
CS150 - Lab Prep 3
Due: Friday Sept. 28 at the beginning of class

For the lab on Friday, we're going to be playing with basic cryptography, which is the study of techniques for securely transmitting and receiving information. For many cryptographic systems, we need to support two keys functions: encryption, where we take a message and encrypt it, and decryption, where we take an encrypted message and give back the original message. To get you ready for lab on Friday, we're going to be going through some key concepts here and play with some of the encryption schemes by hand.

Many, many encryption schemes exists and have been used throughout time. In this lab, we'll be focusing on what are called *private key* encryption schemes. If you have someone that you know you'd like to communicate securely with (i.e. without anyone being able to read it) you decide beforehand on a secret *key* that you both know. Then, when you want to send a message, you encrypt it using the secret key and send the encrypted message to the other person. The other person can then use that same key to decrypt the message. Below is a picture of this process:



http://en.wikipedia.org/wiki/Cryptography

1 Caesar's scheme

One of the simpler and more famous encryption schemes is credited to Julius Ceasar. The encryption scheme is called a "substitution cipher" in that each character is substituted for a different character in the message. To specify a substitution cipher we need to specify the alphabet of characters that our message can contain and then for each of these characters we specify a corresponding substitution character. Caesar's approach was to substitute a letter in the original alphabet with the letter that was some fixed number of letters up in the alphabet. For example, if you chose 2 as your fixed number:

```
alphabet: 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 ‘ ’
key:      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
```

or if we chose 4

```
alphabet: 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 ‘ ’
key:      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
```

Notice that we include the space as a character (written as ‘ ’) and that we wrap around when we’re at the end of the alphabet.

When encrypting, you simply replace each character in your message with the encrypted character (i.e. the character the fixed number up in the alphabet) and to decrypt it you reverse the process.

Do the following based on Caesar’s method and write the answer on a piece of paper to be handed in:

1. Encrypt `‘this is a test’` with a spacing of 2 (to encrypt, substitute letters from “alphabet” to letters in “key”)
2. Decrypt `‘kbnqxgbeubencuu’` with a spacing of 2 (to decrypt, substitute the letters from “key” to letters in “alphabet”)
3. Decrypt `‘stenewjfqqceit’` with a spacing of 5

2 General substitution cyphers

As mentioned, Caesar’s scheme is a specific example of a substitution cipher. In general, a substitution cipher can substitute any letter for any other letter. For example:

```
alphabet: 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 ‘ ’
key:      h v i e k s y r b d a j q w n c x m g u f l t p ‘ ’ o  z
```

4. Decrypt the following message with this substitution and add it to your answers from above: `‘urkzak zbgzvhwhwhg’`

The basic idea when programming something like this is to note that if we store both the original `alphabet` and the encryption `key` as strings, we can find the `index` in the original alphabet string and use that same index in the key to lookup the corresponding encryption character and vice versa during decryption. Specifically:

alphabet:	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	‘ ’
	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26
key:	h	v	i	e	k	s	y	r	b	d	a	j	q	w	n	c	x	m	g	u	f	l	t	p	‘ ’	o	z

Answer the following questions:

5. If we have a variable called ALPHABET initialized as follows:

   ```
   ALPHABET = "abcdefghijklmnopqrstuvwxyz "
   ```

   (again notice the space at the end) and we have a letter stored in a variable called `letter`, write a Python expression that will **find** (*hint*) the index in `ALPHABET` where that letter occurs

6. Finally, if we stored that index in a variable called `index` and the encrypted letters corresponding to `ALPHABET` are stored in a variable called `key`, e.g.

   ```
   key = "hvieksyrbdajqwncxmgufltp oz"
   ```

   write a Python expression that will find the corresponding encrypted letter to the `letter`. Hint: you'll use `index` and it should be something quite simple.

Something to think about: (You don't actually need to answer this) Why aren't substitution cyphers very good? Put another way, why are they relatively easy to break?