Advanced Cryptography

You're all pretty clever now, so we've given you a lot of tasks to do! Feel free to just do the ones that seem fun or challenging. Some of the tasks will require you to do some of the previous tasks.

Task 1: Caesar Ciphers

1. Use the workbook to create a basic Caesar Cipher encryptor/decryptor.
2. Turn your Caesar Cipher into a function. We'll need to use this later!
3. If you like, add some extensions to your Caesar cipher. You can:
   a. Add extensions from the workbook
   b. Make the message retain capitalisation
   c. A simple caesar cipher based on the latin alphabet is easy to crack, so let's confuse the cracker by making an ASCII caesar cipher.
   As it is an ASCII cipher, you need to care about the capitalisation and symbols in the message you will encrypt.
   Hint: for ASCII reference, go to : http://www.ascii-code.com/

Task 2: Vigenere Ciphers

At the moment, your Caesar Cipher encoder takes a single rotation number (e.g. 13 for ROT13) and rotates every letter in the plaintext by the same number. But the Caesar Cipher is vulnerable, because it doesn't do anything to hide the letter frequency of your message.

It's well-known that "e" is the most frequent letter in English. So let's say an eavesdropper managed to get hold of your encrypted message:

```
Uryyb gurer! Gur grkg tvira gb gur Pnrfne Pvcure qrpbqre arrqf gb or
snveyl ybat va beqre gb jbex. Jung vg ernyyl arrqf vf n ybg bs 'r'f,
gb or cerpvfr.
```

They could analyze it, find that "r" is the most common letter, and guess that your cipher converted all the "e"s to "r"s, a rotation of +13. And reversing that rotation could easily give them the plaintext message.

To combat this, we can upgrade the Caesar Cipher to a Vigenere Cipher. The Vigenere Cipher doesn't just use a single modifying number (like +13), but a keyword. Each letter in the keyword is used in turn to encode a single character. For example:

Keyword:	`ab`
Plaintext:	`Here's an example of the Vigenere Cipher.`
Repeated key:	`abab a ba bababab ab aba babababa bababa`
Number to shift by:	`0101 0 10 1010101 01 010 10101010 101010`
Encoded:	`Hfrf's bn fxbmqlf og tie Wiheoese Diqhfr.`

The first character in the keyword ("a") has been used to modify all the odd-position letters, and the second character in the keyword ("b") was used to modify all the even-position letters. The odd letters (the ones under the "a"s in the repeated key) are **unchanged**, because mapping "a" to "a" doesn't change anything. But the even letters have been **shifted up by 1**, since "a" was mapped to "b", a difference of 1 letter.

Keyword:	`abc`
Plaintext:	`Here's an example of the Vigenere Cipher.`
Repeated key:	`abca b ca bcabcab ca bca bcabcabc abcabc`
Number to shift by:	`0120 1 20 1201201 20 120 12012012 012012`
Encoded:	`Hfte't cn fzanrlf qf uje Wkgfpesg Cjrhft.`

This example has a 3-letter keyword. Can you see how each letter in the keyword takes turns modifying the plaintext message?

You can upgrade your Caesar Cipher to a Vigenere Cipher by calculating the new modifying number each time you encode the next letter in a message. An 'a' would shift the letters 0 positions (i.e. no change), a 'b' would shift the letters 1 position, an 'n' would shift 13 positions, and a 'z' would shift 25 positions.

Task 3: Substitution Cipher

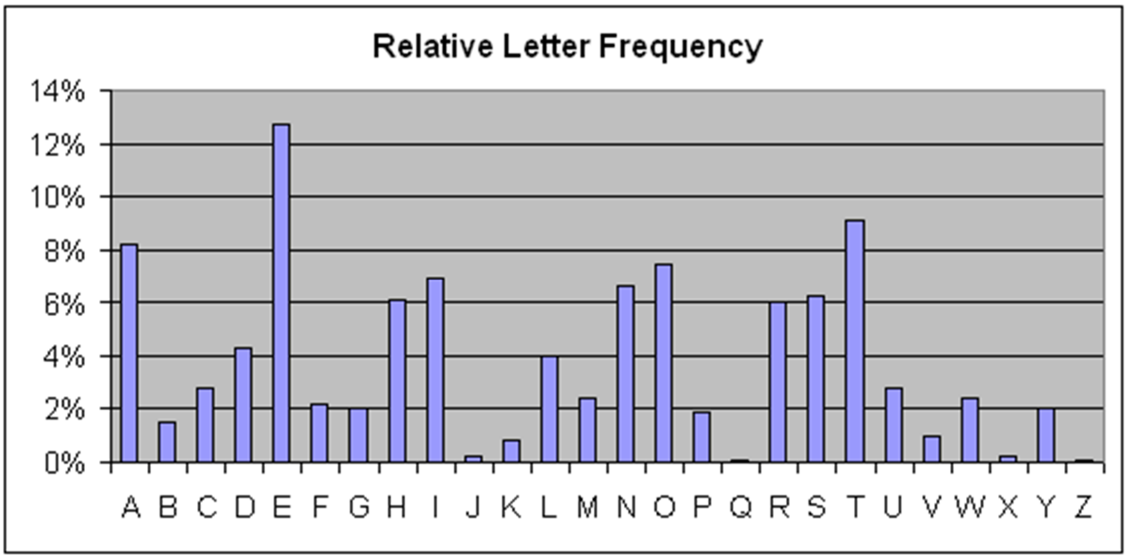
A substitution cipher is just a random mapping between letters. You might always replace “E” with “j”, replace “b” with “t”, etc. Any mapping will work, as long as every letter only maps to one other letter (you can’t replace “b” with “t” and replace “e” with “t”). A substitution cipher is easy to make if you have a dictionary!

Use your knowledge of dictionaries to:

1) Create a dictionary which stores which stores your mapping.
2) Write a program that uses the dictionary to encode a secret message
3) Write a decoder. **Beware**, your dictionary won’t work both ways!

Task 4: Frequency Analysis

A good way to crack someone Caesar Cipher is to do a frequency analysis on it. This means you look at how many times each character appears. This is helpful because in text some letters are more common than other. Like the letter "e" is much more common than the letter "z". Text normally has a distribution that looks like this:



If we can look at our encoded text, we can then see which character is the most common. WE can then assume that this character corresponds to the most common letter in English. the letter "e".

1. Make a tool that counts up the distribution of the letters in the encrypted text and stores it in a dictionary. Get it to print out the maximum occurring character.
2. Visualise your whole dictionary at once using ascii art! You can easily represent your dictionary as an ascii bar chart like this:

```
a | ######
b | ##
c | ###
d | ####
e | ###########
.....
z | ##
```

    But don't just print out one "#" for every time you see a letter!
    That would get realllllllllllllllllllly long for a long text!
    Instead make sure to show your results as a percentage.

```
Only use 100 “#” and split them up between the letters to
represent how many times you saw each character.
```

Task 5: AI Caesar Cipher Cracker

In the previous task we were able to do a frequency analysis on a text, this means we can find out hte most common character and it probably corresponds to the letter “e”. Once you know what it corresponds too, it’s easy to figure out the rotation for the Casesar Cipher! Just find the number of spaces between the letter “e” and the highest frequency character.

But humans are lazy! Why would we bother doing any of these steps when the computer can do it for us!

Let’s create a program that uses frquency analysis to work out the cipher rotation and then cracks the cipher for us!

1. Take the Frequency Analysis code we made above in Task 4.1. We don’t need to be able to print it out, we just need the dictionary so we need are able to return the letter that is the most common.
2. Assume that the most common letter corresponds to the letter “e”.
3. Work out where the most common letter sits in the alphabet. We know where the letter “E” sits already.
4. Take the difference between where “e” sits in the alphabet and where the most common letter sits. This is the rotation key! (You might do this by finding the index of the most common letter in a string like “abcdefg…..wxyz”)
5. Use the rotation key and the decoded text, put it into the Caesar cipher decoding function you made in Task 1.2.
6. Test your code, encode some text using your original Caesar Cipher encoder. See ifyou your AI can guess how to decode the text.

Task 6: AI for Cracking Vigenere Ciphers

Cracking a Viginere ciphers is just like cracking several Caesar Ciphers, when you know how long the key is at least.

Let’s pretend we have a key length of 3. We can then split up our text into 3 strings. We can use our Caesar Cipher Cracker to crack each of the 3 sections.

“this shows you how to split up your encoded text”

Now without the spaces (we don’t care about them!):

```
“thisshowsyouhowtosplitupyourencodedtext”
“t  s  o  y  h  t  p  t  y  r  c  e  e  t”
“ h  s  w  o  o  o  l  u  o  e  o  d  x  ”
“  i  h  s  u  w  s  i  p  u  n  d  t    ”
```

Now we can do a Caesar Cipher crack on each of the 3 separated strings. This works because each one third letter was encoded with the same key, so we are just grouping together everything encoded with the same key. Find the key of each of the 3 parts and we can crack the cipher.

1. Try making a AI Vigenere Cipher Cracker, where you know how long the key is.
2. Now try and make one that cracks a Vigenere when you don’t give it a key. This might involve:
    a. Brute forcing different key lengths
    b. Getting a human to check if the output looks correct for different key lengths.
    c. Testing you text once you have deciphered it. Have one letter words been translated to “a” or “I”. If they are another random letter, it’s probably not right!
    d. Test to see if 2 letter words have been translated into real 2 letter words
    e. Use list of english words to test if most of the words in your decoded text seem to be real words!