COSC 101: Introduction to Computing I
Homework 11
Fall 2014

The goal of this homework is to give you practice with dictionaries, file processing, and string methods and functions.

This homework is **due on Wednesday, November, 19, 2014 11:55pm**.

1 Introduction: cryptanalysis

On a previous assignment, you saw some examples of cryptography, where the sender wants to send a message that only the receiver can read. An eavesdropper may see the message, but because it is "scrambled," the eavesdropper cannot make sense of it. Well, is that really true? The eavesdropper may not have the key, but... could a clever hacker nevertheless figure out what the key is and crack the code? The answer is yes! Your challenge in this assignment is to write a program that breaks the cipher that you wrote in the last assignment!

After you complete this assignment, reward yourself with a trip to the movies to see "The Imitation Game" which is coming out over Thanksgiving break. It's a movie about Alan Turing who used cryptanalysis to break the German's secret messages during World War II. While I can't speak for the movie's version of the tale, the true story of Turing and his accomplishments is really incredible. He is sometimes described as the father of computer science for his many contributions.

2 Getting started

- In the `hw11.zip` file, we include some example files that you can use to test your code.
  - `cipher1.txt`
  - `cipher2.txt`

  If you double click on these files on a Windows machine, it may open these files using Notepad, in which case they do not display correctly. I suggest opening them in IDLE. For fun, we include two more test files,
  - `mystery1.txt`
  - `mystery2.txt`

  These files are fairly large (`mystery1.txt` has 13,000 lines of text). Don't worry about these until your program works on the smaller test examples.
- Read about Your Task and the background on cracking codes.
- Read Section 5 to find out exactly what you need to do.

- After you finish, review the checklist and, if you wish, try the challenge problem!

3 Your Task

This homework builds directly on top of the previous homework. Ultimately, you should turn in a *single* program that completes the tasks outlined in this homework and the previous one.

If you did not manage to complete the previous homework, you have options.

- Move on from hw10. Write a program that fulfills *only* the requirements outlined in this document. We will grade `hw10.py` and `hw11.py` separately.
- Revise hw10. Write a single program that fulfills the requirements of both this homework and the previous homework. We will grade `hw11.py` only and assign grades for both homework 10 and 11 based on it.

4 Cracking the Caesar cipher

An eavesdropper who does not know the key can try to break the ciphertext by inferring, or guessing, what key was used to make it. For reasonably long messages, one of the easiest strategies is to use letter frequency. The letter E is often the most frequently ocurring letter in a passage of English text.

Here's an example. Suppose the ciphertext is `'EQORWVGT UEKGPEG KU VJG DGUV'`. To crack the Caesar cipher, perform these steps:

1) First, figure out which letter occurs most frequently in the ciphertext. Here, that's the letter G which occurs 5 times.
2) Figure out the key: we make the educated guess that E must have been mapped to G, therefore the key is 2.
3) Decipher the text. Here's a neat trick for that: deciphering is the same process as ciphering (which was explained in the previous homework), you just need to adjust the key. In this example, the fact that the key is 2 means that each letter was rotated right by 2 places. To decipher, we need to rotate left 2 places. However, rotating left two places is equivalent to rotating right $26 - 2 = 24$ places. So, to decipher we can simply cipher with the key of $26 - 2 = 24$. If we do this, the resulting text is `'COMPUTER SCIENCE IS THE BEST'`.

Here's a second example. Suppose the ciphertext is `'BNLOTSDQ RBHDMBD HR SGD ADRS'`. To crack the Caesar cipher, perform these steps:

1) First, figure out which letter occurs most frequently in the ciphertext. Here, that's the letter D which occurs 5 times.
2) Figure out the key: E must have been mapped to D, therefore the key is -1 or equivalently 25 (rotating left one letter is equivalent to rotating right 25 letters).
3) Decipher the text. If ciphertext was created with the key of 25, then to decipher, we can simply cipher with the key of $26 - 25 = 1$. If we do this, the resulting text is, once again, `'COMPUTER SCIENCE IS THE BEST'`.

Important detail: when counting the most frequently occurring characters, characters such as spaces, punctuation, etc. should *not* be counted – otherwise space will probably be the most frequent. In addition, counts should not distinguish between upper and lower case: e.g., "Aardvark" has three occurrences of the letter "a."

5 Program requirements

Your task is to write a program that allows the user to either cipher, decipher, or crack the code. Your program should work exactly as described in homework 10 with two exceptions:

- The first line should ask if the user wants to cipher, decipher, or crack the code.
- If the user wants to crack the code, it should ask for the name of the file containing ciphertext and print out the best guess for the key using the strategy outlined above. (The user can run your program again to actually decipher it using the guessed key.)

As stated in homework 10, you must have a function called `main`. This is where you should prompt the user to choose between ciphering and deciphering and cracking. Depending on their response you ask for additional information (see hw10 and new examples below).

Of course, as with the previous homework, you are expected to write additional "helper" functions to support the `main` function. We leave the program design up to you. All functions should have appropriate docstrings.

6 Examples

Here are some examples of the new functionality of cracking the code:

```
Do you want to (c)ipher or (d)ecipher or (cr)ack? cr
Name of file to read: cipher1.txt
I cracked the code! The key is 2
```

You should then be able to use your program to decipher:

```
Do you want to (c)ipher or (d)ecipher or (cr)ack? d
Which cipher (c)aeser or (v)ignere? c
Enter key: 2
Name of file to read: cipher1.txt
Name of file to write: deciphered1.txt
```

If your program works correctly, you should see that `deciphered1.txt` contains the text:

```
COMPUTER   SCIENCE
IS  THE  BEST
```

If you run your program again and this time try to crack `cipher2.txt`, you should get this:

```
Do you want to (c)ipher or (d)ecipher or (cr)ack? cr
Name of file to read: cipher1.txt
I cracked the code! The key is 25
```

7 Checklist

- So that we can more easily test your code, make sure that you ask the user for inputs in *exactly* the same order as shown in the examples above and in the examples provided in the previous homework.
- Once your program works on the smaller examples, try your program on the larger ciphered files `mystery1.txt` and `mystery2.txt`.
- Double check the checklist from the previous asssingment!

8 Challenge problem

The file `mystery_challenge.txt` was written using the Vigenère cipher whose key length is 8 characters. Embellish your main program to ask users if they want you to crack a file encrypted using the Vigenère cipher.

When your program works correctly, it should produce an output like this:

```
Do you want to (c)ipher or (d)ecipher or (cr)ack? cr
Which cipher (c)aeser or (v)ignere? v
Name of file to read: mystery_challenge.txt
I cracked the code! The key is ????????
```

where `????????` is replaced with a eight character string representing the best guess at the key.

Challenge challenge problem Above, we give the key length. But what if you didn't know it? Could you guess it? To try to crack the key length using the Friedman test, described here `http://goo.gl/M6ZsT`.