DM 536 Introduction to Programming

Fall 2013 Re-exam Project

Department of Mathematics and Computer Science
University of Southern Denmark

November 20, 2013

Introduction
The purpose of the project for DM536 is to try in practice the use of programming techniques and knowledge about the programming language Python on small but interesting examples.

Please make sure to read this entire note before starting your work on this part of the project. Pay close attention to the sections on deadlines, deliverables, and exam rules.

Exam Rules
This project is the 2nd quarter re-exam for the course.

Thus, the project must be done individually, and no cooperation is allowed beyond what is explicitly stated in this document.

Both parts of the project have to be passed to pass the overall project.

Deliverables
A short project report (at least 6 pages without front page and appendix) has to be delivered. This report has to contain the following 7 sections:

- **front page** (course number, name, section, date of birth)
- **specification** (what the program is supposed to do)
- **design** (how the program was planned)
- **implementation** (how the program was written)
- **testing** (what tests you performed)
- **conclusion** (how satisfying the result is)
- **appendix** (complete source code)

The report has to be delivered as a single PDF file electronically using Blackboard's SDU Assignment functionality.

Deadline

January 10, 2014, 23:59

Part 1

Julius Caesar scrambled his messages by shifting every letter of a word by a certain number of steps in the alphabet. We have two functions, *encrypt* that scrambles a message into a seemingly senseless text and *decrypt* that unscrambles a given text into the original message.

The key in this scrambling method is the number of steps a letter is shifted. If we know this number, we can unscramble by shifting all letters back by the same number. Consider the following examples:

- shifting b by 5: $b \to c \to d \to e \to f \to g$ (result is g)
- shifting w by 6: $w \to x \to y \to z \to a \to b \to c$ (result is c)
- shifting g back by 5: $g \to f \to e \to d \to c \to b$ (result is b)

Your task in this part of the project is to write a Python program that reads a text from an input file given as a first argument, asks the user for a key (the number of shifts) and whether to decrypt or encrypt. The resulting text is then printed to the screen. If an output file is given as a second argument, the resulting text should additionally be written to that output file.

The Input

The input is a normal message containing only words over the alphabet $a, b, \ldots, z$ separated by whitespace. Thus, our input might look like this:

```
galia est
pacata
```

The Output

The output of your solver should also be a text containing only words over the alphabet $a, b, \ldots, z$ separated by whitespace. Thus, our output for a shift of 25 might look like:

```
fzkhz
drs
ozbzsz
```

The Task

Implement a function that handles the input and output from and to files and from and to the user. Then implement two functions `encrypt(message, key)` and `decrypt(text, key)` that scramble and unscramble as described above. Use these functions in order to achieve the required behaviour.

Part 2

The Caesar cipher used in the first part of the project is not very secure. There are two reasons for this. First, there are only 26 possible shifts (and one does not change the plaintext). So it is possible to go through the 25 interesting shifts and look for the plain text.

Second, and more importantly, the letter frequency distribution of a natural language text is not random. In English, for example, the letter "e" is by far the most common letter. Thus, when you are sure the plaintext is in English, analyzing the letter frequency distribution of the encrypted text will easily reveal the key used.

Your task in this part of the project is to write a Python program that reads an encrypted English text from an input file given as a first argument and a letter frequency distribution for English as the second argument. Then, the English plaintext corresponding to the encrypted English text should be printed to the screen. To this end, you will have to determine the key that was used to encrypt the text. The key should be determined *without user interaction*. If an output file is given as a third argument, the plaintext should additionally be written to that output file.

The Input

The first input is an encrypted message containing arbitrary text, that is, any kind of characters are allowed. You may assume, though, that only the letters $A, B, \ldots, Z$ and $a, b, \ldots, z$ have been encrypted. Thus, our input might look like this:

> Puzaybjavyz jhu zluk lthps av hss vy zlsljalk pukpcpkbhs Bzlyz, Zabkluaz, Nyvbwz, Alhjopun Hzzpzahuaz, Puzaybjavyz vy Vizlyclyz pu h Jvbyzl. Myvt h Ishjrivhyk Slhyu jvbyzl, lthpsz jhuuva il zlua av hufvul dov pz uva h tltily vm aol jvbyzl.

The second input is a letter frequency distribution for English available as `engelsk.dat` from the home page.

The Output

The output of your solver should be the English plain text corresponding to the encrypted message:

> Instructors can send email to all or selected individual Users, Students, Groups, Teaching Assistants, Instructors or Observers in a Course. From a Blackboard Learn course, emails cannot be sent to anyone who is not a member of the course.

The Task
Implement a function that handles the inputs from files and the output to the screen (and possibly to a file).

Write a class `Histogram` that uses a dictionary to keep frequency data. There should be functions or an `__init__` method to build histograms both from the given letter frequency data as well as from a given encrypted text.

Write a method for comparing the histogram obtained from the encrypted text to the histogram from the given letter frequency data. This method should return as a result the best guess for the key needed to decrypt the given encrypted text. For the above example, this method should return 7.

Write a method that returns a list of letters (and possibly frequencies) ordered by the frequency such that the most frequent letter comes first. This method should be used by the above method to determine the key.

You decide which method to use for determining a potential key. For inspiration, see the Wikipedia page on the Caesar cipher:

`http://en.wikipedia.org/wiki/Caesar_cipher`

Your implementation should at least be able to correctly decrypt two of the three texts `engelsk1.crypt, engelsk2.crypt, engelsk3.crypt`, which you can download from the course home page:

`http://www.imada.sdu.dk/~petersk/DM536/#project`