This is the report for Security assignment 1 to show how I did decrypt the messages using Python 2.7 with an extra English word checking library PyEnchant.

Program structure

Each message will be represented by a file containing cryptography method (e.g. CaesarCrypto.py, ColumnarCrypto.py, etc..). In those files will store defined methods to solve and decrypt the messages. There will be a Main.py file that will store all the messages in string and each one will be executed using the suitable method from other files.

There will be a word checking file that use additional library (PyEnchant) to check English word. This will reduce the time it will print out to the screen unnecessary decrypted messages. A tutorial will be shown below for demonstration how to install.

PyEnchant installation

In order to install pyenchant, python must firstly be installed. Because I'm using linux operating system, I already installed Python 2.7.

- Download the source code from PyEnchant website http://pythonhosted.org/pyenchant/download.html
- Enter to the directory of downloaded file, the *setup.py* file will be in there.
- Run the command *python setup.py install*

I'm using PyCharm IDE, I can then use the library when making new project.

Message 1

Ceasar Cipher is all about rotating the alphabet by a number of positions which the key is also unknown in this case. So, I look at the Assignment requirement pdf file and I see the alphabet as listed below:

ABCDEFGHIJKLMNOPQRSTUVWXYZ .,:;()-!?$'"\n0123456789

It is said that this alphabet to be assumed to use for all messages. So I created a string to store this alphabet and converted it to python list using the method list(object)

Next, I defined a method call cipher_table(number) which would convert the original alphabet to the cipher alphabet based on what number I wanted to shift. A number 1 would shift each character to one position, number 2 would shift to 2 positions and so on. It then will return the list containing the cipher table.

Finally, I defined the method decrypt_caesar(message, boolean) which will take the message as a parameter and boolean as if user wanted to use English word checking or not, the method would print out the possible solutions. In this method, I used the method defined above and loop through all possible shifting number based on how long the original alphabet was. Then, I compared each characters in the message with the corresponding character in the original table and get the index of it to match with the shifted table and concatenated them to a string. Next, use the defined SpellCheckingEn.py to check the words in the decryped message to see if it contained English words. If it was (even there was only 1 character that has meaning), I printed out to the screen to check again manually. If it wasn't (all the words are false), then it would not print out.

At the result, I got three messages printing out to the screen. Those were shifting number 11, 39 and 44. The 11 and 44 got printed out because those two might contain something that were valid as "English word" but the whole message had no meaning. So, only the shifting number 39 had meaning. The decrypted message is shown below:

EVERYTHING YOU WANTED TO KNOW ABOUT THE SECURITY-FOCUSED BLACKPHONE

BARCELONA, SPAIN

HERE AT MOBILE WORLD CONGRESS, SILENT CIRCLE AND GEEKSPHONE HAVE JUST ANNOUNCED MORE DETAILS ON THE BLACKPHONE, A PHONE FOCUSED ON SECURITY AND PRIVACY. "BLACKPHONE" SEEMS TO BE BOTH A PRODUCT AND A COMPANY, AS IN THE COMPANY BLACKPHONE WILL PROVIDE UPDATES AND SUPPORT FOR THE PRODUCT BLACKPHONE. HAVING BEEN COFOUNDED BY PHIL ZIMMERMAN, THE CREATOR OF PGP E-MAIL ENCRYPTION, THE COMPANY HAS TONS OF SECURITY TALENT. BLACKPHONE WAS ANNOUNCED ABOUT A MONTH AGO, BUT THIS IS THE FIRST TIME WE'RE GETTING DETAILS ON JUST WHAT THE BLACKPHONE IS AND HOW IT WORKS.

Message 2

In this message, the algorithm was using Columnar Transposition without knowing the key. So, by reading it, I remembered this was quite similar to the one in the lecture’s slide. Remembering the way to solve it in class, I thought I must first find the factors of the length of the message.

Firstly, I defined the method called get_factor() which take “number” as a parameter. This method will take an input as a number and return the factors of it as python list, simple and straightforward. The algorithm for it would be finding the 0 remainder of all possible numbers from 2 (because cannot divide 0 and 1 !) up to the length of the message. Note that this method would skip the first factor (which was 1) and the last factor (which was the length) because it wasn’t different than the message itself when decrypting!

Next, I defined a method called decrypt_columnar() which take “message” as a parameter. This method would receive the message which I wanted to decrypt and print out to the screen all possible answers. I tried to write down on papers what if I made a table with 4x3 and a table with 3x4. Would the message be different ? The answer was “yes”. When writing down the message in the row and encrypting using column, I had different answers. So there would be 2 answers for each 2 factors. For example, assuming the length of the message was 32. The factors would be 1, 2, 4, 8, 16, 32. The get_factor() method will skip the first and last factor. So the whole factors were 2, 4, 8, 16. The possible combinations are (2,16) (4,8). Each of them will have 2 answers (because I needed to reverse the row and column like I said before). So the total possible answers were 4. The loop in later lines just tried to make all possible tables, put the messages in horizontal lines and decrypted using vertical lines. In python, I use list in list which acted as 2D arrays to perform the decryption. I also used the Spell Checking class (which was declared in message 1) to check for english words.

As a result, I got 3 messages printed out to the screen. There were (40,4), (32,5) and (20,8) which were column-row in order. The first two might have some valid English words among the whole messages so when I checked again it wasn’t the real one. The last one clearly had meaning. The decrypted message is showed below:

NEW IOS FLAW MAKES DEVICES SUSCEPTIBLE TO COVERT KEYLOGGING, RESEARCHERS SAY

PROOF-OF-CONCEPT APP IN APPLE'S APP STORE SENT KEYSTROKES TO REMOTE SERVER.

Message 3

The initial method (failed)

In this message, the algorithm is random substitution cipher and the key is also unknown. So, I firstly thought of making a method that count the frequency of each character in the message. The method receive the message as a parameter and return 2D list that have the letter and the frequency of each. So I got the frequency in the following order:

0 : 80 times

X : 54 times

“ : 39 times

W : 36 times

………..

Then, according to this website

http://www.math.cornell.edu/~mec/2003-2004/cryptography/subs/frequencies.html

The most frequent letters are (in order) : E, T, A, O, I, ….

I tried to replace the ‘0’ with ‘E’ due to the fact it repeats a lot. Then, I also replaced the letter ‘X’ with ‘T’. However, I then got into dead end after spending 5-6 hours try to guess and decrypt by hand.

The better method (worked)

After failing by using the first method, I tried to rework by using different approach. I found out that because the message had got messed up due to the fact that I hadn’t known what the spaces were. I thought that breaking the space out was the most important to know frequent two-letter or three-letter words to have higher chance to guess.

I copied the message and paste to a text editor (such as notepad++ running on wine because it has marking which is useful in my opinion). I then tried to replace the most frequent characters in the message with the ‘space’ to see if words were broken down evenly. I found that the letter ‘0’ seemed like it was the ‘space’ because words were broken down like shown below:

```
I" L'0N 0QVWI0"880WI3XX0"',! 606X"3'90 XQ0I"3LQ"3X0LNX' 4W0:VT0'N:WQ"3X$EXWWX30'N:WQ"3X0QN!
8L0I"BX0IX81XL0WIX0,N'WJ0E!W0WIX0'N:WQ"3X0I"' 4W07I" 6XL0,!7IR$$E"37X8N "J0'1"V
$QIV8X0WIX06"8"TF0'H0Q"'0X"'V8F0WIX0,N'W0V,1N3W" W0WIV 60"  N! 7XL0"W0"',! 64'0,NEV8X0QN38L07N 63X"013X"0XBX
WJ0WIX07N,1" F0"8'N0'INQXL0N::0WI3XX0 XQ0',"3W0Q"W7IX'R0"',! 60LX,NXL0WIX06X"30)J0WIX06X"30)0 XNJ0"
L0WIX06X"30:VWR0WIX0(6"8"TF(0E3" LV 60V'06N X0EX7"!'X0WIX0Q"W7IX'0 N08N 6X303! 0" L3NVLDWIXF4BX0'QVW7IXL0WN0"',!
64'0WV2X 0N'R
```

I highlighted the letter ‘0’ because if I replaced it with Space, It might mix up with the real space. Then, I traced the message and found that the word ‘WIX’ was repeated the most, 11 times. According to the tips I found in this website, https://sites.google.com/site/codesforscouts/tips-for-solving-cryptograms, the most frequent three-letter word was THE. I tried to replace ‘W’ -> ‘T’, ‘I’ -> ‘H’, ‘X’ -> ‘E’ and it seemed reasonable.

Now I saw that the word th3ee is nearly complete, I guess it was the word ‘three’ so i replaced 3 -> r.

Next, there was a word “88 which I guessed was too, all or see. However, e and t were already replaced. So all was the most appropriate since the word behind it was “three” which “all three” was reasonable. So I replaced “ -> a , 8 -> l . Now the message looked like this:

```
ha L'0N 0QVth0all0three0'a,'! 606ear'90 eQ0harLQare0LNe' 4t0:VT0'N:tQare$Eetter0'N:tQare0QN!
lL0haBe0hel1eL0the0,N'tJ0E!t0the0'N:tQare0ha' 4t07ha 6eL0,!7hR$$Ear7elN aJ0'1aV
$QhVle0the06alaTF0'H0Qa'0ea'VlF0the0,N't0V,1Nrta t0thV 60a  N! 7eL0at0'a,'! 64'0,NEVle0QNrlL07N 6re"01re"0eBe
tJ0the07N,1a F0al'N0'hNQeL0N::0three0 eQ0',art0Qat7he'R0'a,'! 60Le,NeL0the06ear0)J0the06ear0)0 eNJ0a
L0the06ear0:VtR0the0(6alaTF(0Era LV 60V'06N e0Ee7a!'e0the0Qat7he'0 N0lN 6er0r! 0a LrNVLDtheF4Be0'QVt7heL0tN0'a,'!
64'0tV2e 0N'R
```

Then, I saw the word QVth seem like the word with which suit the phase “with all three”. So I replaced Q-> w, V -> i.

Now, I saw the word harLware which hah 99% chance to be “hardware”, so I replaced L -> d

```
ha d'0N 0with0all0three0'a,'! 606ear'90 ew0hardware0dNe' 4t0:iT0'N:tware$Eetter0'N:tware0wN!
ld0haBe0hel1ed0the0,N'tJ0E!t0the0'N:tware0ha' 4t07ha 6ed0,!7hR$$Ear7elN aJ0'1ai
$while0the06alaTF0'H0wa'0ea'ilF0the0,N't0i,1Nrta t0thi 60a  N! 7ed0at0'a,'! 64'0,NEile0wNrld07N 6re"01re"0eBe
tJ0the07N,1a F0al'N0'hNwed0N::0three0 ew0',art0wat7he'R0'a,'! 60de,Ned0the06ear0)J0the06ear0)0 eNJ0a
d0the06ear0:itR0the0(6alaTF(0Era di 60i'06N e0Ee7a!'e0the0wat7he'0 N0lN 6er0r! 0a drNidDtheF4Be0'wit7hed0tN0'a,'!
64'0ti2e 0N'R
```

Now I see wat7he'R and 'wit7hed which might have 7 as c and ‘ as s. So i replaced 7 -> c, ‘ to s.

The word sN:tware is definitely *software*. I replaced N -> o, : -> f.

The word <u>has 4t</u> had high chance to be *hasn't*. I replaced *space* -> n, 4 -> '

The word <u>chan6ed</u> was *changed.* I replaced 6 -> g

```
hands0on0with0all0three0sa,s!ng0gears90new0hardware0doesn't0fiT0software$Eetter0software0wo!
ld0haBe0hel1ed0the0,ostJ0E!t0the0software0hasn't0changed0,!chR$$EarcelonaJ0s1ain
$while0the0galaTF0sH0was0easilF0the0,ost0i,1ortant0thing0anno!nced0at0sa,s!ng's0,oEile0world0congress01ress0eBen
tJ0the0co,1anF0also0showed0off0three0new0s,art0watchesR0sa,s!ng0de,oed0the0gear0)J0the0gear0)0neoJ0an
d0the0gear0fitR0the0(galaTF(0Eranding0is0gone0Eeca!se0the0watches0no0longer0r!n0androidDtheF'Be0switched0to0sa,s!n
g's0ti2en0osR
```

Now it looked like shown below:

The phase *<u>,ost i,1ortant thing</u>* which likely to be *most important thing.* I replaced , -> m 1-> p.

The word <u>anno!nced</u> was *announced*. I replaced ! -> u

The word <u>moEile</u> was mobile. I replaced E -> b

The word <u>companF</u> was company. I replaced F -> y

The word <u>galaTy</u> and <u>FiT</u> which had the word T most likely to be x. I replaced T -> x

The phase <u>haBe helped</u> which was *have helped.* I replaced B -> v

The phase <u>watchesR samsung</u> had the letter R stand between the space and the word "watches". Also, the letter R appeared at the end of the message. I guessed R -> .

The word *(galaxy(* seemed like it had double quotes around it. I replaced (-> "

Now I see that much.$$barcelonaJ spain had J which most suited for the letter "," . Because it seemed like it was written like that in a newspaper or something. I replaced J -> ,

Since *space* -> n, I thought it was ok to replace 0 -> *space.* The message now looked like this:

```
hands on with all three samsung gears9 new hardware doesn't fix software$better software wou
ld have helped the most, but the software hasn't changed much.$$barcelona, spain
$while the galaxy sH was easily the most important thing announced at samsung's mobile world congress press even
t, the company also showed off three new smart watches. samsung demoed the gear ), the gear ) neo, an
d the gear fit. the "galaxy" branding is gone because the watches no longer run androidDthey've switched to samsun
g's ti2en os.
```

Since 9 and D appeared only once, I guessed either of them was either ! or ; . If this message was in the newspaper, the letter 9 was probably the symbol "-" because it was likely to appear on the title of a topic in a newspaper. The letter D was probably ";" because it separated to sentences.

Now, I see that <u>$</u> was probably a line break (\n) since it was suitable to make the paragraph look nicer. I replaced $ -> \n

Then, I got stuck figuring out the rest because It had limited information such as “ H “ (what number ? s2, s3, s4 ???), “) “ (what number is this symbol?), “ 2 “ (what possible letter it had ?). I then researched online using some of the words in this message and I got some information in the description of a youtube video. The link of the youtube video is shown below:

https://www.youtube.com/watch?v=MK64Uj8EFME

I decrypted the message as shown below:

HANDS ON WITH ALL THREE SAMSUNG GEARS-NEW HARDWARE DOESN'T FIX SOFTWARE
BETTER SOFTWARE WOULD HAVE HELPED THE MOST, BUT THE SOFTWARE HASN'T CHANGED MUCH.

BARCELONA, SPAIN

WHILE THE GALAXY S5 WAS EASILY THE MOST IMPORTANT THING ANNOUNCED AT SAMSUNG'S MOBILE WORLD CONGRESS PRESS EVENT, THE COMPANY ALSO SHOWED OFF THREE NEW SMART WATCHES. SAMSUNG DEMOED THE GEAR 2, THE GEAR 2 NEO, AND THE GEAR FIT. THE "GALAXY" BRANDING IS GONE BECAUSE THE WATCHES NO LONGER RUN ANDROID;THEY'VE SWITCHED TO SAMSUNG'S TIZEN OS.

Message 4

For this message, the algorithm is unknown but the key is 12. So the best way is to traverse from other algorithms in the previous questions. So I started with Caesar cryptography. I refactored a little bit in the CaesarCrypto.py, created another method that decrypt Caesar with a specific key (not random like in message 1).

The method was made easily by using the same logic as decrypting in message 1. However, in this time, instead of looping through all possible ways to make cipher table, it only used the key as passing by parameter. The new method was *decrypt_caesar_with_key(message, key, boolean),* the *boolean* indicated if I wanted to use the SpellChecking python class or not. In addition, this method would use the key in both direction (left and right). For example, If the key is 3, It would loop 2 times to transpose in both directions. So, a ->d, b -> e, c -> f (transpose to the right) as first cipher table and a -> x, b -> w, c -> v(transpose to the left), something like that.

Surprisingly, the answer was found when the program used the key that transpose to the left. So, this message used Caesar cryptography to encrypt.

The full message is shown below:

YOU HAVE TO SHOW YOUR WORK FOR EVERY QUESTIONIF YOU SOLVE THE PROBLEM

"BY HAND", YOU HAVE TO EXPLAIN HOW YOU SOLVED THE PROBLEMIF YOU WROTE

A PROGRAM TO SOLVE THE PROBLEM, YOU HAVE TO SUBMIT YOUR SOURCE CODE WITH

AN EXPLANATION OF HOW YOU USED IT TO SOLVE THE PROBLEMYOU MUST WRITE

YOUR OWN PROGRAMS: YOU CANNOT DOWNLOAD SOFTWARE FROM THE INTERNET OR

"BORROW" SOURCE CODE FROM OTHER STUDENTSPLAGIARISM RULES WILL BE APPLIED

VERY STRICTLY IN THE MARKING OF THIS ASSIGNMENT

Message 5

Message 6

Message 7