Project 1: Breaking the unbreakable Code

The middle ages made great progress in many human arts, not excluding the art of cryptography. One of the breakthroughs was an improvement on the Caesar cipher, that was made popular by Vigenère as the “unbreakable code”.

Caesar’s Cipher

We first take a look at Caesar’s code and how to program it in Python. As is traditional, we convert each text into one containing only capital letters. The Caesar cipher uses as a secret a single letter. It conceptually then places two copies of the alphabet over each other, where the “A” is arranged over the secret letter. If the secret letter is N, then

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
N	O	P	Q	R	S	T	U	V	W	X	Y	Z	A	B	C	D	E	F	G	H	I	J	K	L	M

This table defines the encoding. The letters in the cleartext are then substituted by the letters of which they are placed. For example, “GAULASAWHOLEISDIVIDEDINTOTHREEPARTS” becomes “TNHYNFNJUBYRVFQVIVQRQVAGBGUERRCNEGF”.

There are many ways in which we can implement this transition. One possibility is to use the encoding of ASCII letters. As we have learned, there exist encodings for letters. In Python, we can use the function ord, that gives the encoding for a letter as a decimal integer. Here is the encoding of the capital letters.

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
65	66	67	68	69	70	71	72	73	74	75	76	77	78	79	80	81	82	83	84	85	86	87	88	89	90

The reverse function is chr. You can see this in action with a simple for loop.

```
for letter in "ABCDEFGHIJKLMNOPQRSTUVWXYZ":
        print(letter, ord(letter))

for num in range(65,91):
        print(num, chr(num))
```

If we want to encode the preceding table, we take the difference between the encoding of a letter and the encoding of ‘A’. For example, if the letter is ‘J’, then we obtain `ord(’J’)-ord(‘A’)=9`. This just means that ‘J’ is the ninth letter of the alphabet. We add the difference to the code of the key: `ord(‘J’) - ord(‘A’)+ord(’N’) = 87`, which is the encoding of the cipher for ‘J’, namely ‘W’. Unfortunately, we need to be careful about going beyond ‘Z’. If we want to encode ’T’, the same calculation yields

```
ord(’T’) - ord(‘A’) + ord(’N’) = 84-65+78 = 97.
```

What we need to do is to subtract 26 to obtain 71, which is the encoding of the correct letter ‘G’. We can algebraically express this by subtracting ord(‘A’), taking the remainder modulo 26, and then add ord(‘A’) to it. This gives us the transformation function:

```
global_offset = ord('A')
global_length = ord('Z')-ord('A')+1

def change(letter, pwd):
    return chr( (ord(letter)+ord(pwd)-2*global_offset)%global_length+global_offset)

def unchange(letter, pwd):
    return chr( (ord(letter)-ord(pwd)+global_length)%global_length + global_offset)
```

In this implementation, I chose to make the constants global variables so that they do not need to be recalculated every time.

The Vigenère Cipher

The Vigenère cipher consists of several Caesar ciphers where neighboring letters are encoded using different secret letters. The Vigenère cipher uses a secret word or phrase. As an example, we use the phrase “LOYOLACHICAGO”. Then the first letter of the cleartext is encoded using “L”, the second with “O”, the third with “Y”, etc. When we reach the end of the secret phrase, we start over again.

Attacks against Caesar’s Cipher

Breaking Caesar’s cipher is trivial. Since there are only 26 secrets, we can just try out all possibilities and select the one that yields a readable text. Alternatively, we can do a frequency count, since the most frequent letter in English text is ‘E’. In fact, a frequency analysis will give us a good idea about the language in which the message was written. For example, in Spanish, the number of ‘E’s and ‘A’s are almost the same, whereas in English, the most frequent letter is 25% more frequent than the next frequent letter. If we have decided that the most frequent letter in the cipher text is — let’s say ‘V’, then we know that

```
    ord('V') = (ord(secret)-ord('A')+ord('E')-ord('A'))%26+ord('A')
```

so that the secret letter is ‘R’.

Attacks against the Vigenère Cipher

The weakness of the Vigenère cipher is the small size of the secret. Our example secret has only 13 letters. This means that every 13th letter is encoded with the same letter. In fact, three of every 13 letters are encoded with secret “O”. Since the frequency of letters in any actual language is far from uniformly distributed, this observation is the key to breaking the code.

If the length of the secret is *n* then every *n*-th letter is encoded with the same letter and hence the chances that a letter in the cipher text is equal to the letter in the cipher removed by *n* is higher than for all other displacements.

As an example, I encrypted the cleartext with this secret, and then counted the number of times that a letter in the resulting cipher text was equal to the letter at distance *n* for various distances *n*. The following table shows the result.

4	17057
5	16572
6	16400
7	16857
8	16270
9	17232
10	16658
11	18083
12	15123
13	28411
14	15143
15	17679
16	16371
17	17346
18	16512
19	16385
20	16183
21	16713
22	17486
23	16443
24	17958
25	15049
26	28522
27	15140
28	17711
29	16563
30	17474

As you can see see, there is a jump at 13 and another one at 26 with a number of coincidences almost twice as large as for the other displacements. This allows us to find the length of the secret phrase. Of course, encrypting with “LOYOLACHICAGO” has the same effect as encrypting with “LOYOLACHICAGOLOYOLACHICAGO”, so there is a noticeable jump in the number of coincidences for all multiples of the secret length as well.

Once we know the length of the secret to be 13, we know that every 13th letter are encrypted with the same letter. Thus, letter 0, letter 13, letter 26, … are all encrypted with the same letter. I collect the letters 0, 13, 26, 39, … in the cipher and determine which one is the most frequent one. This turns out to be ‘P’. Now, since the most frequent letter in English is “E”, we know that “E” is encrypted by “P”. This means that “D” is encrypted with “O”, “C” with “N”, “B” with “M”, and “A” with “L”. I thus recovered successfully the first letter of the secret. For the second letter, I look at the letters 1, 14, 27, 40, … of the cipher. The most frequent letter is “S”, and since “E” goes to “S”, “A” has to go to “O”. Applying this method a total of 13 times, the secret is successfully recovered and the cipher can now be decoded.

We have broken the “unbreakable code”.

Description of the Code

You will find in the adjacent program several functions. The function `clean` changes a text file to one where all letters are changed to capital letters and all punctuation marks, digits, and white spaces are removed. The result is a file with a single line in it. The functions `cipher` and `decipher` encrypt and decrypt using a Vigenère cipher. You will need to write the functions `displacement` and `find`.

Deliverables

Decrypt the file `cipher.txt` by determining the secret phrase. You should submit on paper the code to obtain the secret phrase, the secret itself, a description of the plaintext, and the Python functions used to obtain the secret. You can work in groups of up to three people.