V-Cipher(Vigenere Cipher)

Problem: We encrypted a file using a vigenere xor cipher. Can you get the flag?

Given: vfile.enc

Hint: Frequency

Info: A Vigenere cipher is a repeating key cipher, in this case it was xored with the plain text. To learn more about how a vegenere cipher at https://en.wikipedia.org/wiki/Vigen%C3%A8re_cipher

Steps:
1) First thing to start with is opening the given file and reading it into a variable. Then we make sure it is an even length so that we can decode the hex version. Since there is a new line, we strip the last character from the file. The following is a screenshot of how it would look. We also save the length of the text.

```
f=open('vfile.enc')
ct=f.read()
# remove new line at end
ct=ct[0:len(ct)-1]
ct=ct.decode('hex')
tl=len(ct)
```

2) The second part and most important is finding the max repeating patterns in the cipher text and finding the largest shifting point. This can be done in many ways, but the rough idea is to take a pattern of a string such as ‘abc’ and see if ‘abc’ is repeated again later in the cipher text. We repeat this for strings from length 2 to 16 in this problem. The following screenshots show the code for analyzing the patterns. The three columns in the output are the key length, the maximum number of occurrences of a repeating pattern, and the mostly occurred repeating pattern (in hex)

```
# find the shifting and the max repeating
print '###### find repeating pattern ######'
rpcnt={}
rpmax={}
for shft in range(5,16):
    rpcnt[shft]=0
    rpmax[shft]=''
    rpt=''
    for i in range(0,tl-20):
        if ct[i]==ct[i+shft]:
            rpcnt[shft] += 1
            rpt += ct[i]
        else:
            if len(rpt) > len(rpmax[shft]):
                rpmax[shft] = rpt
            rpt=''
    print shft, rpcnt[shft], rpmax[shft].encode('hex')
```

```
###### find repeating pattern ######
5 84 2a
6 182 3b37
7 95 72
8 87 3b3a
9 72 3126
10 68 213d
11 91 383b
12 229 63263135
13 85 382620
14 65 3c
15 67 373a
```

3) It can easily be seen that 12 and 6 have the most repeating patterns, meaning one of them is most likely the key length. Usually it is better to start at the smaller factor number, meaning try 6 first. Hence, we guess the key size is equal to 6. Then we regroup the cipher text based on the key size. We put the cipher text into smaller sections(columns) of the cipher text with the length of 6. Then we use the 1 gram frequency for the alphabet and swap the characters accordingly to the English alphabet frequency. Normally in a frequency chart. The most used character is not a space because the normal frequency is only 0-25 which is A-Z. In our cipher text we want to assume that the most used character is a space(‘ ‘). We then analyze the cipher text blocks that we created to swap the character based on their high frequencies. This can be seen in the following screenshot.

```
# guess key length is ?#
keysize=6

# regroup ciper text
ctg={}
for i in range(0,keysize):
    ctg[i]=''
    for j in range(i,tl,keysize):
        ctg[i]+=ct[j]

# 1-gram each group
ctgf={}
for i in range(0,keysize):
    ctgf[i]={}
    for c in ctg[i]:
        if c in ctgf[i]:
            ctgf[i][c]+=1
        else:
            ctgf[i][c]=1

# analyze each group
print '###### crack each group ######'
ptg={}
for i in range(0,keysize):
    maxc=''
    maxk=0
    for k in ctgf[i]:
        if ctgf[i][k] > maxk:
            maxk=ctgf[i][k]
            maxc=k
    # assume ' ' is the most frequent char
    xk = ord(maxc) ^ ord(' ')
    ptg[i]=''
    for c in ctg[i]:
        ptg[i] += chr(ord(c)^xk)
    print i, ptg[i]
```

4) Now after we have analyzed each group with our guessed key length as 6. We can now begin to put the cipher text back together which will now be converted into our plain text. Since we had our blocks of text we need to merge them back together which will become our plain text. This can be done like the following:

```
# put plaintext together
pt=''
for i in range(0,len(ptg[0])):
    for j in range(0,keysize):
        if i<len(ptg[j]):
            pt += ptg[j][i]

print '###### cracked ######'
print pt
```

5) Now, we make a full script to find the flag...