

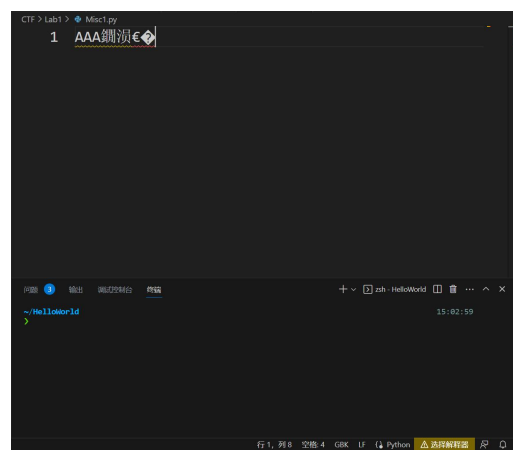
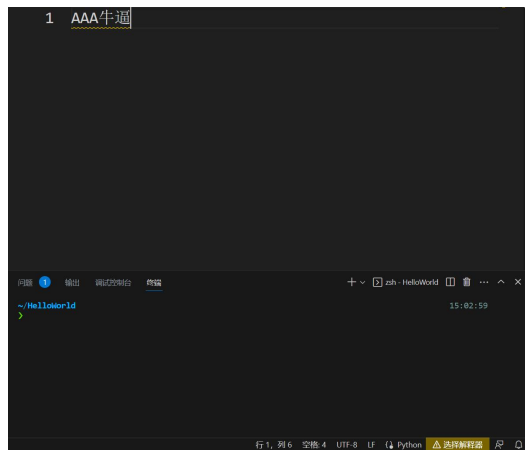
# Misc Lab1

## 3220102732 周伟战

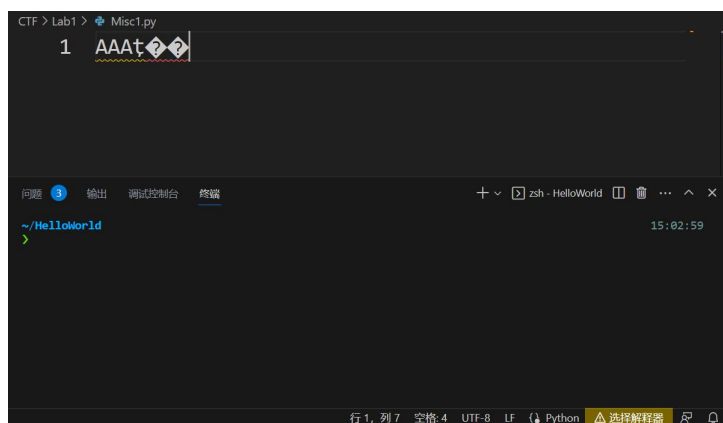
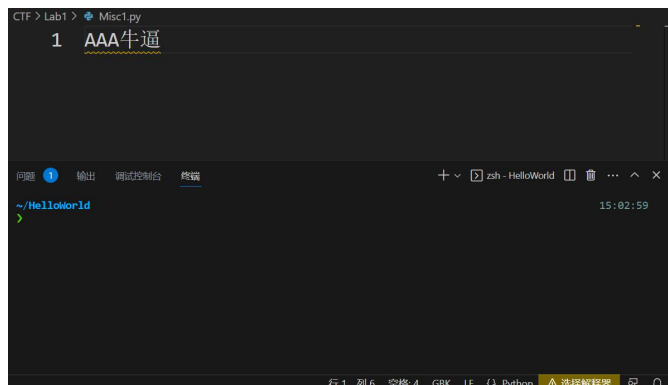
### Task1

#### 1.1 复现 6 种乱码情况

1) 用 GBK 解码 UTF-8 编码的文本



2) 用 UTF-8 解码 GBK 编码的文本



3) 用 latin-1 解码 UTF-8 编码的文本

Input

AAA牛逼

REC 5 1

Raw Bytes

LF

Output

AAAç•••é•%|

REC 9 1

2ms

ISO-8859-1 Latin 1 We...

LF

4) 用 latin-1 解码 GBK 编码的文本

Input

AAA牛逼

REC 5 1

Simplified Chinese GBK

LF

Output

AAAÀÊ±ä

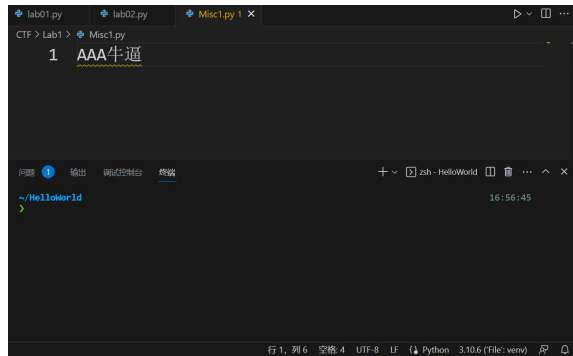
REC 7 1

2ms

ISO-8859-1 Latin 1 We...

LF

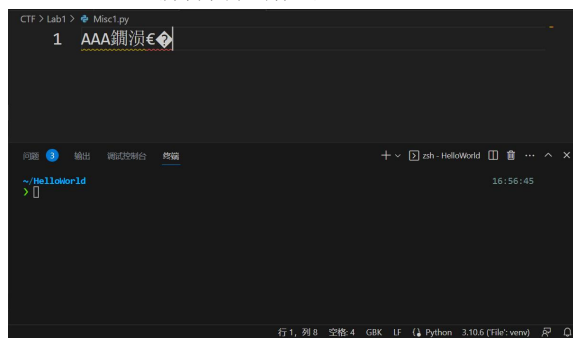
5) 先用 GBK 解码 UTF-8 编码的文本，再用 UTF-8 解码前面的结果  
UTF-8 的原码



```
CTF > Lab1 > Misc1.py
1 AAA牛逼

行 1, 列 6  空格 4  UTF-8  LF  Python  3.10.6 (File view)
```

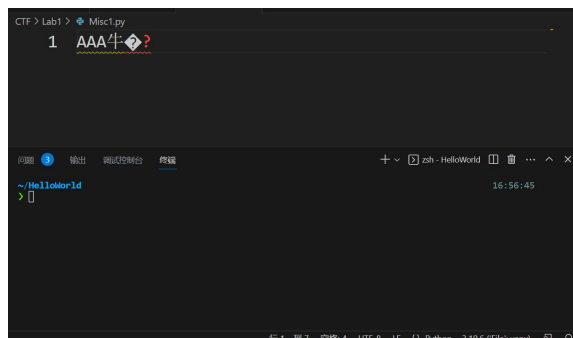
通过 GBK 重新打开后解码



```
CTF > Lab1 > Misc1.py
1 AAA𠄎𠄎

行 1, 列 8  空格 4  GBK  LF  Python  3.10.6 (File view)
```

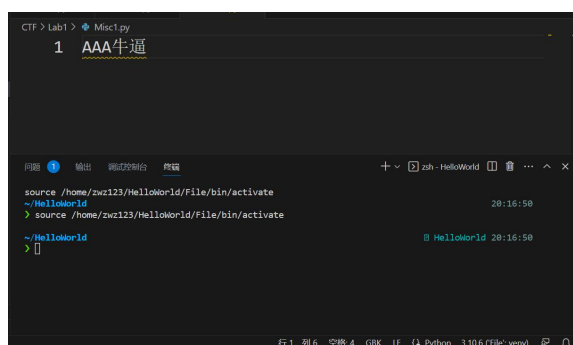
然后再保存后重新利用 UTF-8 打开



```
CTF > Lab1 > Misc1.py
1 AAA牛逼

行 1, 列 7  空格 4  UTF-8  LF  Python  3.10.6 (File view)
```

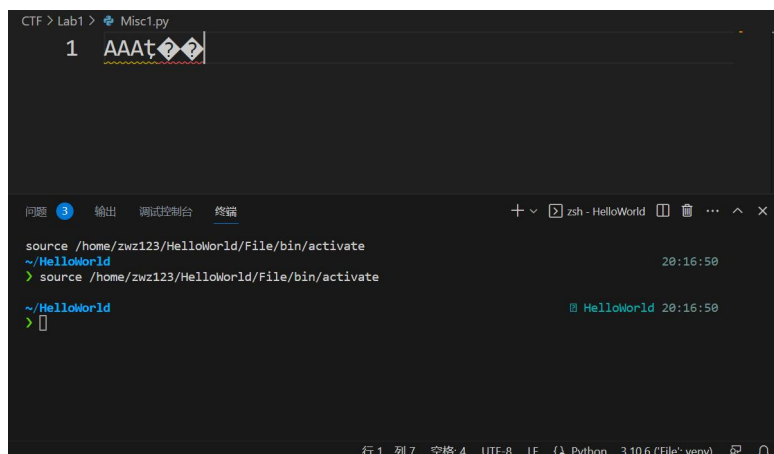
6) 先用 UTF-8 解码 GBK 编码的文本，再用 GBK 解码前面的结果



```
CTF > Lab1 > Misc1.py
1 AAA牛逼

~/.HelloWorld
> source /home/zwz123/HelloWorld/File/bin/activate
~/.HelloWorld 20:15:50
>
```

再重新用 **UTF-8** 编码打开该 **GBK** 文件

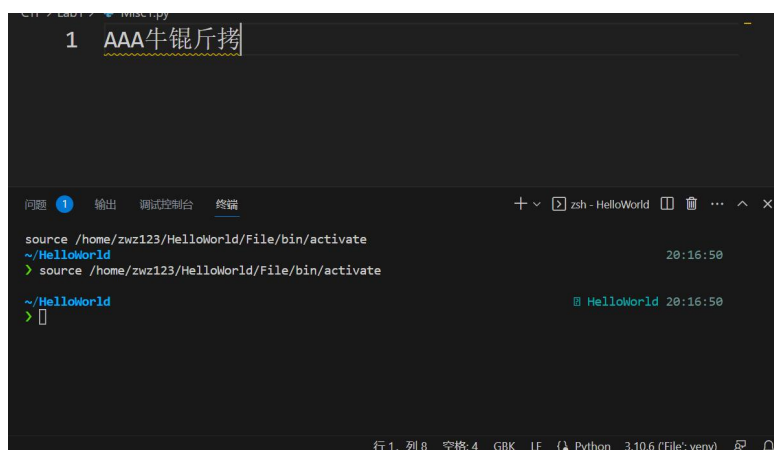


```
CTF > Lab1 > Misc1.py
1 AAA??

问题 3 输出 调试控制台 终端
+ zsh - HelloWorld
source /home/zwz123/HelloWorld/File/bin/activate
~/HelloWorld 20:16:50
> source /home/zwz123/HelloWorld/File/bin/activate
~/HelloWorld 20:16:50
> []

行 1, 列 7 空格 4 UTF-8 LF Python 3.10.6 (File: venv)
```

再进行重新编码保存，并用 **GBK** 重新打开得到 **AAA 牛银斤拷**



```
CTF > Lab1 > Misc1.py
1 AAA牛银斤拷

问题 1 输出 调试控制台 终端
+ zsh - HelloWorld
source /home/zwz123/HelloWorld/File/bin/activate
~/HelloWorld 20:16:50
> source /home/zwz123/HelloWorld/File/bin/activate
~/HelloWorld 20:16:50
> []

行 1, 列 8 空格 4 GBK LF Python 3.10.6 (File: venv)
```

## 1.2.1

**Q:**在自行研究了 **GB** 系列编码后，请阐述 **GB** 系列是如何实现三个版本兼容的？

**A:** **GB** 系列实现的是 **GB18030** 兼容的三个版本

**1:** **GBK**

**2:** **GB2312**

**3:** **ASCII**

(1) 首先是 **GB2312** 是可以叫做区位码，由两个字节组成，第一个

字节是区，第二个字节是位。（分别是高低位，“高位字节”使用了 **0xA1 - 0xF7**（把 **01 - 87** 区的区号加上 **0xA0**），高位字节有部分是空区，“低位字节”使用了 **0xA1 - 0xFE**（把 **01 - 94** 加上 **0xA0**））他们分别选取了 **94** 个 bit，那么 **GB2312** 就可以包含 **94\*94=8836** 个字符。**GB2312** 为了兼容 **ASCII** 码，把小于等于 **127** 的字符，表示为 **ASCII**，单个字节，依旧以 **0** 开头，与原本的意义相同。但其他的字符都是两个字节，而且都以 **1** 开头，为了与 **ASCII** 码区分开来。

如：我叫 **ABC** 这句话用 **GB2312** 码来表示：

**11001110 11010010**（我） **10111101 11010000**（叫） **01000001**（A）  
**01000002**（B） **01000003**（C）

这样就达到了兼容 **ASCII** 码的效果；

（2）**GBK** 兼容了 **GB2312** 和 **ASCII** 码：**GBK** 编码是微软在 **GB2312** 的编码规则基础上，利用 **GB2312** 未编完的空间，共收录 **21886** 个汉字和图形符号，其中汉字（包括部首和构件）**21003** 个，图形符号 **883** 个。这样下来，**GBK** 编码向下完全兼容 **GB2312** 编码，也等同于兼容 **ASCII** 码。

（3）**GB18030** 兼容 **GBK** 编码：采用变长多字节编码，每个字可以由 **1** 个、**2** 个或 **4** 个字节组成。具有这样的特点，使得其可以包含 **161w** 的字符。**1** 个字节的是保留了 **ASCII** 码的编码规则，而 **2** 个字节的是对于 **GBK** 编码的保留与拓展，而 **4** 字节的是用于包含所有 **Unicode** 编码下的 **UTF** 四字节区段。从而做到了 **GB18030** 兼容 **3** 个版本。

## 1.2.2

**Q:** 针对六种乱码情况，哪些是你觉得可以恢复的，哪些是不可以恢复的？

**A:** 我觉得以上只有两种可恢复，**3**，**4** 两种。以 **4** 举例，因为对于 **Latin-1** 是对于 **ASCII** 码的扩展，可以表示任何单字节的字符，任何字节流都可以用其解码。（**ASCII** 最高位为 **0**，而 **Latin-1** 最高位为 **1**，这样将 **GBK** 编码下的字符转换成 **Latin-1**，可以看成将多个两字节的字符，分别拆开都拆成了 **2** 个单字节的 **Latin-1** 字符。）这样的好处是所有单字节都可以涵盖掉。然后，只要是该 **Latin-1** 字符串都是由 **GBK** 而来，那么，重新回到 **GBK** 就可以实现恢复的效果。对于 **UTF-8** 编码也是如此。所以我认为 **3**，**4** 可恢复，而 **1**，**2**，**5**，**6** 不可恢复。原因是会出现编码损失，无法像 **Latin-1** 转化成单字符可以完全覆盖，当 **GBK** 和 **UTF-8** 很多的编码区域是没有交集的，就导致有些乱码无法重新恢复回去。

## 1.2.3

“锼斤拷”，这三个词是由于 **UTF-8** 编码是来源于 **Unicode** 字符集的转变问题。就比如 **Unicode** 在和 **GBK** 编码转化的过程中，**GBK** 编码的字符肯定是有一部分 **Unicode** 无法表示，**Unicode** 官方用了一个占位符来表示这些文字，这就是：**U+FFFD REPLACEMENT CHARACTER**。

**U+FFFD** 的 **UTF-8** 编码是 **0xEFBFBD**，如果重复多次形成：

**EFBFBDEFBFBDEFBFBDB**。那么将转化后的 Unicode 重新变成了 GBK 编码的时候。GBK 中的除了 ASCII 码，其他都是两个字节的编码，**EFBF,BDEF,BFBD**.就分别对应了“锷”“斤”“拷”三个字符。1.1.6 中的“6)先用 UTF-8 解码 GBK 编码的文本，再用 GBK 解码前面的结果”AAA 牛逼---->AAA 牛锷斤拷，就是由于这个原因。“逼”这个字，从 GBK 变为 UTF-8 的编码是 Unicode 无法表示的。

## Task2

### Challenge1

### Vigenere 解码

以下是解题代码：

Step1:先求解密钥长度

代码在 Misc.zip 中的 Vigenere-decrypt(1).py 中

```
1 #Vigenere-decrypt1 use for get the length of key
2 from random import randrange
3
4 text_list=' !"%$%&'()*+,-./0123456789:;<=>?@ABCDEFGHIJKLMNPOQRSTUVWXYZ[\]^_`abcdefghijklmnopqrstuvwxyz
5 #key 是从text_list中随机选择15-30个的字符 生成的密钥
6 def TextRead(file):
7     #将txt文件转化为字符串
8     with open(file,"r") as f :
9         TextContent = f.read()
10    return TextContent
11
12 def Num(s):
13     #已经是分组之后取出的一组字符串 ， 计算单组中的重合指数
14     count=[0]*97
15     Num = 0
16     for i in range(len(s)):
17         index=text_list.index(s[i])
18         count[index]+=1
19     for i in range(97):
20         if count[i]!=0:
21             Num+=count[i]*(count[i]-1)
22     Num/=len(s)*(len(s)-1)
23     return Num
24
25 def Ic(s,k):
26     #用来统计不同k值下的重合指数
27     Ic=[]
28     str=""
29     for i in range(len(s)//k):#组数
30         for x in range(i,len(s),k):
31             str+=s[x]
32         p = Num(str)
33         Ic.append(p)
34     return Ic
```

```

35
36 T=TextRead("vigenere.txt")
37
38 for k in range(15,31):
39     p=0 #p用于改变存放每个k值下的Ic值总合
40     for x in range(k):
41         p+=Ic(T,k)[x]
42     p/=k
43     print (p)
44 #C[i]中存放的是Ic的均值

```

```

0.03459784961078679
0.036255956720073736
0.03344268791214023
0.03609708271185117
0.03520982960248611
0.034196049891084775
0.03383462158072611
0.034262435691612865
0.03830053002394148
0.036043758485237894
0.03647059166610666
0.0347398741452819
0.03250182725535983
0.0338630034887915

```

0.03771683757058582

0.034791778064769595

这些是输出后的结果，发现 k=29 的时候，算出的 Ic 最接近 0.038。由此可以推断出 key 的长度为 29。

## Step2:利用 key=29 来解密 Vigenere 乘法加密

这是第一组的字母出现

u>; +00 +;0qn^5 EEiT6Td!!ds!;h!EETT1T }!O16 } QSETEd s1 ^0T}T+^ 11 O;&0&;!1d+ Td ;E;^ m;s^E T+1+ndh15&0T^0; }^ T^T^OO0; } !& ;;;TT0m^

Vigenere-decrypt3.py 用于获得该分组后的字符出现频率

```

Vigenere-decrypt3.py > -
1 text ='>; +00 +;0qn^5 EEiT6Td!!ds!;h!EETT1T }!O16 } QSETEd s1 ^0T}T+^ 11 O;&0&;!1d+ Td ;E;^
2 frequency = {key:0 for key in text} #初始化一个空字典
3
4 # 遍历文本中的每个字符
5 for x in text:
6     if x in frequency:
7         frequency[x] += 1
8     else:
9         frequency[x] = 1
10 for i in frequency:
11     frequency[i]/=len(text)
12
13 # 输出字符频率
14 sorted_frequency=dict(sorted(frequency.items(),key = lambda x:x[1],reverse=True))
15 for char, count in sorted_frequency.items():
16     print(f"Character '{char}' appears {count} ")
17
18

```



用以上的代码统计频率从大到小排序

Character ' ' appears 0.1555555555555556  
Character 'T' appears 0.1111111111111111  
Character ';' appears 0.0962962962962963  
Character '^' appears 0.07407407407407407  
Character 'O' appears 0.06666666666666667  
Character 'E' appears 0.05925925925925926  
Character '1' appears 0.05925925925925926  
Character '!' appears 0.05185185185185185  
Character '+' appears 0.044444444444444446  
Character 'd' appears 0.044444444444444446  
Character '}' appears 0.037037037037037035  
Character 'O' appears 0.02962962962962963  
Character '&' appears 0.02962962962962963  
Character 's' appears 0.02222222222222223  
Character 'n' appears 0.014814814814814815  
Character '5' appears 0.014814814814814815  
Character '6' appears 0.014814814814814815  
Character 'h' appears 0.014814814814814815  
Character 'm' appears 0.014814814814814815  
Character 'u' appears 0.007407407407407408  
Character '>' appears 0.007407407407407408  
Character 'q' appears 0.007407407407407408  
Character 'i' appears 0.007407407407407408  
Character 'Q' appears 0.007407407407407408  
Character 'S' appears 0.007407407407407408

推断出来第一个字符是 **key[0]=** 这个一开始还没破译出来，需要等最后爆破

对于 i=1 即第二组字符串推断可知：频率分布

Character ' ' appears 0.23648648648648649  
Character '}' appears 0.11486486486486487  
Character 'G' appears 0.08108108108108109  
Character 'E' appears 0.05405405405405406  
Character '9' appears 0.05405405405405406  
Character 'T' appears 0.0472972972972973  
Character ',' appears 0.0472972972972973  
Character 'o' appears 0.04054054054054054  
Character 'a' appears 0.04054054054054054  
Character ':' appears 0.033783783783783786  
Character 'q' appears 0.033783783783783786  
Character '+' appears 0.033783783783783786  
Character 'r' appears 0.02702702702702703  
Character 'F' appears 0.02702702702702703  
Character '~' appears 0.02702702702702703

Character '\*' appears 0.02027027027027027  
Character '8' appears 0.02027027027027027  
Character 'p' appears 0.013513513513513514  
Character 'L' appears 0.006756756756756757  
Character 'i' appears 0.006756756756756757  
Character '.' appears 0.006756756756756757  
Character 'V' appears 0.006756756756756757  
Character 'H' appears 0.006756756756756757  
Character 'w' appears 0.006756756756756757  
Character 'N' appears 0.006756756756756757

可以推测“j”所对应的大概率是字母“e” 此时的密钥 key[1]=''

Vigener-decrypt4.py 中来利用字母出现的频率逆推 key

```
1 Text_List=' !"#%&\'()*+,-./0123456789:;<=>?@ABCDEFGHIJKLMNPQRSTUVWXYZ[\\]^_`abcdefghijklmnopqrstuvwxyz  
2 for key in range(97):  
3     if (Text_List.index('a')*key)%97==Text_List.index('Q'):  
4         print(Text_List[key])
```

有代码推得 key[1]=''

同理，看第三组

Character ' ' appears 0.1925925925925926  
Character 'y' appears 0.08888888888888889  
Character '\_' appears 0.08888888888888889  
Character 'p' appears 0.07407407407407407  
Character 'Z' appears 0.05925925925925926  
Character '3' appears 0.05185185185185185  
Character 'V' appears 0.05185185185185185  
Character 'c' appears 0.05185185185185185  
Character 'l' appears 0.044444444444444446  
Character 'h' appears 0.044444444444444446  
Character '%' appears 0.037037037037037035  
Character '@' appears 0.037037037037037035  
Character '7' appears 0.02962962962962963  
Character '\*' appears 0.02222222222222223  
Character 'M' appears 0.014814814814814815  
Character '}' appears 0.014814814814814815  
Character '.' appears 0.014814814814814815  
Character '2' appears 0.014814814814814815  
Character '|' appears 0.007407407407407408  
Character 'K' appears 0.007407407407407408  
Character 'f' appears 0.007407407407407408  
Character '{' appears 0.007407407407407408  
Character 'u' appears 0.007407407407407408  
Character 'O' appears 0.007407407407407408  
Character ')' appears 0.007407407407407408  
Character ';' appears 0.007407407407407408  
Character 'Q' appears 0.007407407407407408

Key[2]=6

同理得到后面的 X.6z cK1rP1lAy~XXXho2aXX &DGX\'X  
(具体推理过程在 u.docx 文件中)

uy thz mid-nin^;a/3j8 oent'i^> the =erm "ic^J5D 8Q" enP{||;d thz AmericaQ i?3S}Q^e, \*G< ice  
NQs still /&t( 6W^inn.vX to ab(ect the (</- {( ord.vv+y ci5]zens in ;-/ Q:]}ed K2V0es. -+e ice tV[R/  
S|W{ wiPY 0he gefwth of +4

<4E Lce 6'\ used ]n hotel?w T@u7:(s, SvW hosp{=als, anu 4@ 4{c6 foAqV+d-loL`ing citw R/@>7:+  
in /|;sh mzQt, fresz p

.( \$ Gnd \*G<0er.Ab=er the d4=

5 sQx (1sjFq1865k2 as ice i%X ?\W" to i7nrigeQte frei0E

>2|IN it 'z^o ca|W into h:\*q/9 z% use( m5en bz(ore 1884w #@>x ?he .Q7 sold ]n New Y:]ZQ  
f8]aadeu?8Ea, ad% Baltim:]aQ 2:% oneQ28Erd ob that so  
(

3 lf+ton 'i ChilQgo, wen(

h ~Vc%liey xTr thz]r own u?x0 M(yl had <76ome 4fssible BxC?,47 G ne6 8TusehL1d  
conveQ4al>H\$ ?he .Q7dox, y precurs:] h; <+6 mo9{|| refe]erator/ -?= 6W6n i#\\7!ted.83aking aQ  
a+;^w]6nt .Q7dox wyl not as x%X( Vl we LnXst noN supposeb Al j8W earu: linetzWnth cen(\*b@u  
<+6 kn[qz;dge L( the phwL<7. {( heaPZ hwich ras esseQ;<?5 <f a sbn7!ce ob refrigeV[

2l\$ {as AGWEmenty:y. The +/xp2l\W(se #\*<Eon t2Qt the b^L

8=7"<x wS~ Tne t2Qt preveQ;a3 j8W ice /|Tm me3=ing was /p > }:+e m.~<1ken, (or it wYL  
T9H c6lti#D Tf thz ice tha( D//~{:ued PY7 cool{4g.

NeveV;-/5H\IN eaAL^ effoe=s to ec:@5p8\*7 %ce .vw}uded rapping ;-/ ^wW in \*LV!kets# which  
k^

-(7 %ce Hi{O doidZ its joBU \*2j o(til v71r thz end of (Ea 3^:W?eenPY 6entueu did in[x&T2\_  
Gchiq\7 the 1Wlicate B[i?3=7 <f i#~}}atiod and cir+\*i?~^4 nee9{W for y4 effici^@

8=7"<x.

HSVQ:AAAO]\_like\_T4xp=Bug:B\_M3'YaS!}

B~= as ear G ?. FD=3, Sv Engen{fus Mary [ &3 ~V:uer, 98Tmas tfore, hau 4/<l f( thq  
|Eght 5:ack. He /Ll<s Q farL Vdout 5renty mi  
xq 2?<l%de PY7 city ff WashiQV

h3b (<r wWnws the @illage :g F< |Z6tow# =1s thz market +x&T<\_E Ahen Y7 used Qn icebo- 5+ (yl  
own f7^ign 5f transp:]

9^\\ [uttqi 0o mae`et, he G/.l= <+Gt c'~<Tmers rould pa?L P1 <+6 ra2nW}y me3=ing stuGg

3 <+6 tu\*~ Tf hiM competi(/bX j{ Pay S [+emiul price f:] #84 "StteAZ ^till (resh anu  
-?/s ]( neS2\$ one-4fund bri+sql!0:W advSv<1ge ob his iceB/[Q %f{xe en?z1ined# was tha( p?/t7:+  
wo'lw no lL4ger hav^

h j|Qgel P\* Oarke5 at nigh( <l |%6r t[ 9;ep t2Wir prodr9a > {1v

PeAYV&s thz most ob[45P. =QB arPn\0ic ceWation r^gi/>j\ row 2{{&le l{ @e is by Q<\  
|](g tW{ ;nvirL4ment - (Ea 42<Wxialy Vld tel+nologie? %L@^zQ[le P\* 1 cul5ore. StoQxs \*  
{%N trq{ dark, Tlay, anu q?3s Qxe gqv7+ally Qvailabl^ x?-H|]Gls. u: addi5]on, dep^@R

3S f( thq zTcali5u, other ]aX2?|T6s mS: de aciWssible: L-/5>\2 hor#~\$ gold# copper, [&3 4y1ger.  
98; difbWrent us^L T2 =+%ch y\*wEetieM put the?x p@j7:%als '|; of {4terest (/ ?3j8:<pol[Dy^ts  
w2f may as5w +2\_ W.ampu{\$ why 4Wople ch:/q/ j{ Sse bLVm and 4ot copp^] H9H: [oth n<;ms  
aeW availaBbal Dz=rougW <sere y:e no coQ9iP.^]W ans6{|^ yet# the way 4& \*(yTr a y\*wEety  
f]ews its x&L8\_{4uent n\ some5]mes appY]al- y4 its Q8Tice y4d use oG %\^=\%c mS27+ialsT The  
use /p >H|=Gin L<1ls, bfr examp  
xs 42^ [e rq~7+ved bfr cerem:@<?5 {"9ecty {n spel]al impoV;%l>HE cr tW{ delieb in  
the ?\*D//IV=Sral ?{hers L( a ston^ 5\j|W6 maE w1use y sculptoV  
h h7 +ens.2y5e to =hat mat^]<?5m@Brat .~ &artilolarly m^]&  
3Sxoa to ':OhropL1ogist i?  
#< |WGlizS2yTn thy= althou0E T9H cGter.'z^ ava{1able to [ X2=yW?y mS: 0o so|W extent b<p8j fx  
inHL];nce N+at it cY@ 32 V:~ist.QV]ly, 5+e materc[iX h^ (o mq':^ detz:mine whY;  
. Wf(e. iY^ do t2W artist? <l EV7Gnesq \Tciet rake saQ(  
3j{ Pattqi:~; an1 the artcL  
X ^: >oma# \Tciet melt saQ( T2 xfxm gu'\^? MoeWover, e[x& \*(74 the ~VOe ma5Wrial is  
\*q/= y4 the ~VOe wa  
by memb^]q 2~ %%ffeA{:0 soc{Wties, ttx +2\_Z <r sP:z; of 5+e work [[b  
<4 W(orm[G\}y frLc cultur^  
h =}1?ure( + sociz=y may scQDt( w+<ose 2{ reprzlent obj^9  
X | Phen[a7!a thy= are im#/bT@I< ?o iP~ &opuly=ion. An x{?4^:Q?ion \*x the y:t of th^ }  
=szW Agey <;lls ~l somethc@ @h{o? thq Z;dievy1 preoccr %T8 : {ith 28;olog{Tal  
doctV4&/s N4 add.2yTn to :evealin0  
#< [:~marE wTncerdl of a s:9</-U\$ ?he b\*:0ent L( that s:9</-Ucl art aVm alsL reflect ;-/ =}1?ure\$~  
^ocia3  
stratific9%T8 :5

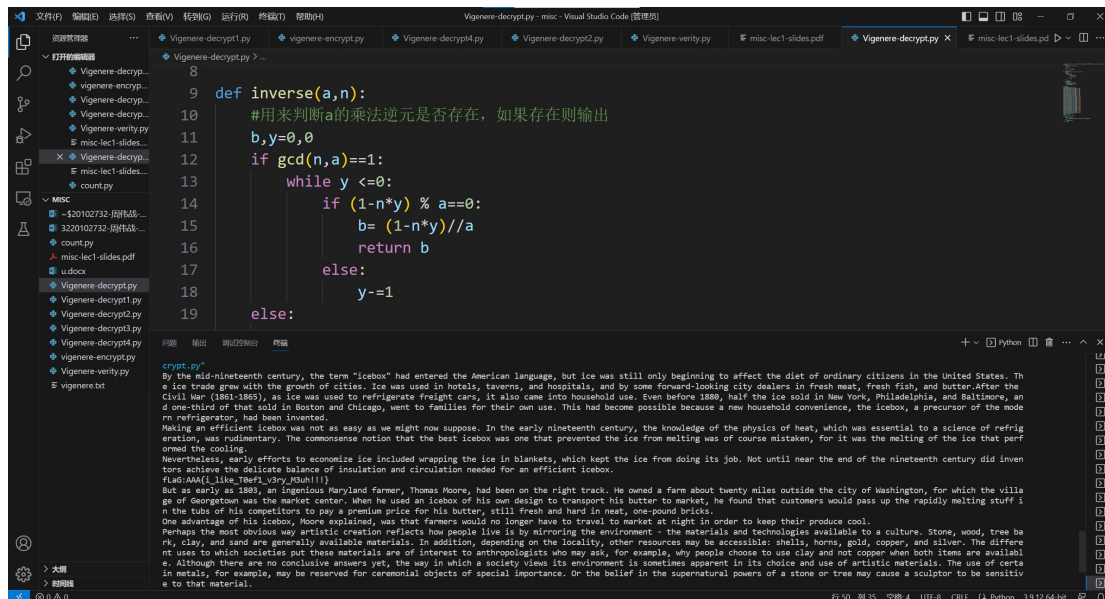
## 开始爆破

1. 第一位 uy-->By 利用 Vigenere-decrypt4.py 中的函数 推得 key[0]='S'
2. 第 27 位为'enPer'---->"enter" key[26]='X'
3. 第 29 位为'ente|'---->"enter" key[28]='a'
4. 第 23 位为'oentury'--->'century' key[22]='4'
5. 第 15, 16, 17 位利用 mid-nineteenth 来破解

得到 key[14]='\\',key[15]='P'.key[16]='-'

最后收尾完成所有密码的破译

得到 key='S.6zck1r,P1IAy~\\p-h02aR4&DGX\'a'(注意转义符)



具体的解题过程在 u.docx 中（分别是每一组的字母出现以及频率）

解题代码在（emm 分成了 6 个小程序，没合并）

Vigener-decrypt.py （用于恢复原文）

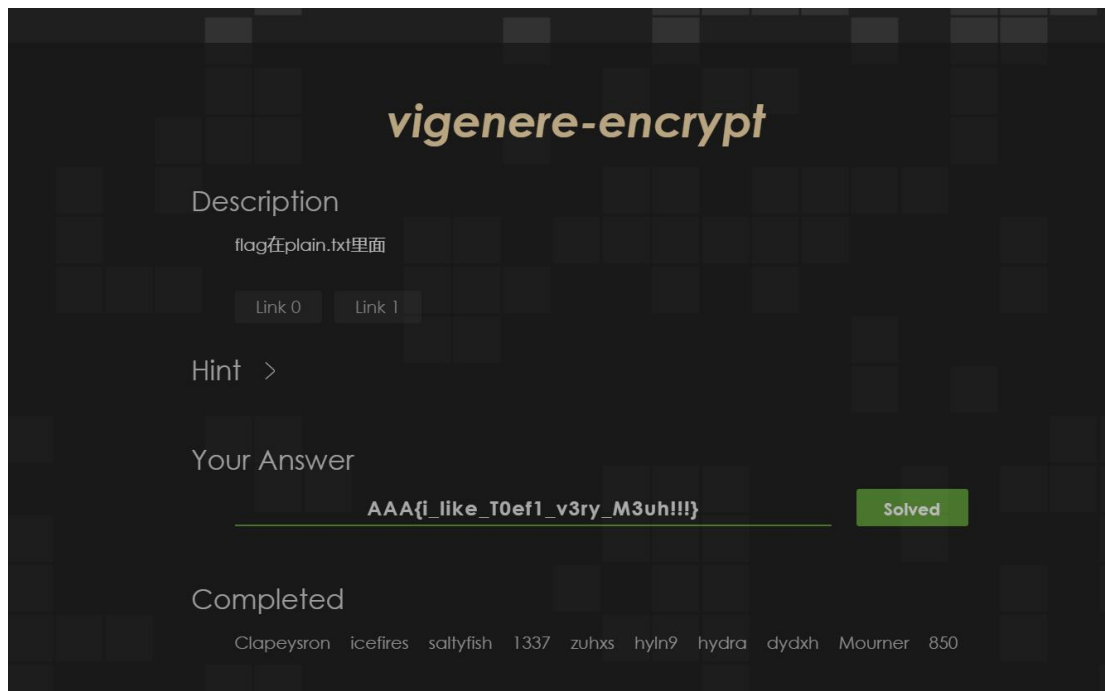
Vigener-decrypt1.py(用于爆破重合指数，获得 key 的长度为 29)

Vigener-decrypt2.py（用于取得各组字符串的内容）

Vigener-decrypt3.py(用于获得每个分组字符串部分的字母出现频率)

Vigener-decrypt4.py（用于反推密钥）

Vigener-verity.py(用于验证该密钥的正确性，通过验证前几位频率较高的字母出现，有无符号出现等来确认正确性)



Flag 是 **AAA{i\_like\_T0ef1\_v3ry\_M3uh!!!}**

（下一页附上原文）

By the mid-nineteenth century, the term "icebox" had entered the American language, but ice

was still only beginning to affect the diet of ordinary citizens in the United States. The ice trade grew with the growth of cities. Ice was used in hotels, taverns, and hospitals, and by some forward-looking city dealers in fresh meat, fresh fish, and butter. After the Civil War (1861-1865), as ice was used to refrigerate freight cars, it also came into household use. Even before 1880, half the ice sold in New York, Philadelphia, and Baltimore, and one-third of that sold in Boston and Chicago, went to families for their own use. This had become possible because a new household convenience, the icebox, a precursor of the modern refrigerator, had been invented.

Making an efficient icebox was not as easy as we might now suppose. In the early nineteenth century, the knowledge of the physics of heat, which was essential to a science of refrigeration, was rudimentary. The commonsense notion that the best icebox was one that prevented the ice from melting was of course mistaken, for it was the melting of the ice that performed the cooling. Nevertheless, early efforts to economize ice included wrapping the ice in blankets, which kept the ice from doing its job. Not until near the end of the nineteenth century did inventors achieve the delicate balance of insulation and circulation needed for an efficient icebox.

**flaG:AAA{i\_like\_T0ef1\_v3ry\_M3uh!!!}**

But as early as 1803, an ingenious Maryland farmer, Thomas Moore, had been on the right track. He owned a farm about twenty miles outside the city of Washington, for which the village of Georgetown was the market center. When he used an icebox of his own design to transport his butter to market, he found that customers would pass up the rapidly melting stuff in the tubs of his competitors to pay a premium price for his butter, still fresh and hard in neat, one-pound bricks.

One advantage of his icebox, Moore explained, was that farmers would no longer have to travel to market at night in order to keep their produce cool.

Perhaps the most obvious way artistic creation reflects how people live is by mirroring the environment - the materials and technologies available to a culture. Stone, wood, tree bark, clay, and sand are generally available materials. In addition, depending on the locality, other resources may be accessible: shells, horns, gold, copper, and silver. The different uses to which societies put these materials are of interest to anthropologists who may ask, for example, why people choose to use clay and not copper when both items are available. Although there are no conclusive answers yet, the way in which a society views its environment is sometimes apparent in its choice and use of artistic materials. The use of certain metals, for example, may be reserved for ceremonial objects of special importance. Or the belief in the supernatural powers of a stone or tree may cause a sculptor to be sensitive to that material.

What is particularly meaningful to anthropologist is the realization that although the materials available to a society may to some extent limit or influence what it can do artistically, the materials by no means determine what is done. Why do the artists in Japanese society rake sand into patterns; and the artists in Roman society melt sand to form glass? Moreover, even when the same material is used in the same way by members of different societies, the form or style of the work varies enormously from culture to culture. A society may simply choose to represent objects or phenomena that are important to its population. An examination of the art of the Middle Ages tells us something about the medieval preoccupation with theological doctrine. In addition to revealing the primary concerns of a society, the content of that society's art may also reflect the culture's social stratification.

