In [1]:
```python
# dictionary of alphabets
dict1 = {}
dict2 = {}

count = 0
for i in range(65,91):
    dict1[chr(i+32)] = count
    dict2[chr(i)] = count
    count+=1
```

In [2]:
```python
key_list1 = list(dict1.keys())
val_list1 = list(dict1.values())

key_list2 = list(dict2.keys())
val_list2 = list(dict2.values())

print(key_list2)
print()
print(val_list2)
```

```
['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']

[0, 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15, 16, 17, 18, 19, 20, 21, 22, 2
3, 24, 25]
```

Q.1 Implement the following message "cryptography is a very simple subject to understand" using Additive/shift/Caesar with key 19.

In [3]:
```python
def caesar_cipher(choice):

    # choice = int(input("Enter 1 for encryption or 2 for decryption: "))

    if choice == 1:
        #encryption
        #
        # p = input("Enter the plain text: ")
        # k = int(input("Enter the key: "))

        p = "cryptography is a very simple subject to understand"
        k = 19

        c = ""
        p = p.upper()

        for i in p:
            if i == " " :
                c = c + " "
                continue
            z = (k + dict2[i]) % 26

            c = c + key_list2[val_list2.index(z)]
        print(f"CIPHER TEXT: {c}")
    elif choice == 2:

        #decryption

        C = input("Enter the CIPHER TEXT: ")
        K = int(input("Enter the key: "))

        P = ""
        C = C.upper()
        for i in C:
            if i == " ":
                P = P + " "
                continue
            x = dict2[i] - K
            if x < 0 :
                x = x + 26
            P = P + key_list1[val_list1.index(x)]
        print(f"plain text: {P}")

    else:
        print("Invalid choice")
        print("Try again")
        caesar_cipher()
```

In [4]:
```
if __name__ == "__main__":
    caesar_cipher(1)
```

```
CIPHER TEXT: VKRIMHZKTIAR BL T OXKR LBFIEX LNUCXVM MH NGWXKLMTGW
```

Q.2 Write a program to perform a brute force attack on the following encrypted message.

In [5]:
```
def brute_force_attack(p):
    # p = input("Enter the text: ")
    p = p.lower()
    for k in range(1,26):
        c = ""
        for i in p:
            if i == " ":
                c = c + " "
                continue

            x =  dict1[i] - k
            if x < 0:
                x = x + 26
            c = c + key_list1[val_list1.index(x)]

        print(c)
        print()
```

In [6]:
```
if __name__ == "__main__":
    x =  "IOL LIJHXIG MYHNYHWY AYHYLUNIL WUH BYFJ SIO WIGY OJ QCNB ZOH LUHXIG MYHNY
    brute_force_attack(x)
```

hnk khigwhf lxgmxgvx zxgxktmhk vtg axei rhn vhfx ni pbma yng ktgwhf lxgmxgvxl yhk tg
r inkihlx cnlm xgmxk rhnk itktfxmxkl bgmh hnk mhhe tgw lxx pahim bm vtg vhfx ni pbma

gmj jghfvge kwflwfuw ywfwjslgj usf zwdh qgm ugew mh oalz xmf jsfvge kwflwfuwk xgj sf
q hmjhgkw bmkl wflwj qgmj hsjsewlwjk aflg gmj lggd sfv kww ozghl al usf ugew mh oalz

fli ifgeufd jvekvetv xvevirkfi tre yvcg pfl tfdv lg nzky wle ireufd jvekvetvj wfi re
p gligfjv aljk vekvi pfli grirdvkvij zekf fli kffc reu jvv nyfgk zk tre tfdv lg nzky

ekh hefdtec iudjudsu wuduhqjeh sqd xubf oek secu kf myjx vkd hqdtec iudjudsui veh qd
o fkhfeiu zkij udjuh oekh fqhqcujuhi ydje ekh jeeb qdt iuu mxefj yj sqd secu kf myjx

djg gdecsdb htcitcrt vtctgpidg rpc wtae ndj rdbt je lxiw ujc gpcsdb htcitcrth udg pc
n ejgedht yjhi tcitg ndjg epgpbtitgh xcid djg idda pcs htt lwdei xi rpc rdbt je lxiw

cif fcdbrca gsbhsbqs usbsfohcf qob vszd mci qcas id kwhv tib fobrca gsbhsbqsg tcf ob
m difdcgs xigh sbhsf mcif dofoashsfg wbhc cif hccz obr gss kvcdh wh qob qcas id kwhv

bhe ebcaqbz fragrapr trarengbe pna uryc lbh pbzr hc jvgu sha enaqbz fragraprf sbe na
l checbfr whfg ragre lbhe cenzrgref vagb bhe gbby naq frr jubcg vg pna pbzr hc jvgu

agd dabzpay eqzfqzoq sqzqdmfad omz tqxb kag oayq gb iuft rgz dmzpay eqzfqzoqe rad mz
k bgdbaeq vgef qzfqd kagd bmdmyqfqde uzfa agd faax mzp eqq itabf uf omz oayq gb iuft

zfc czayozx dpyepynp rpypclezc nly spwa jzf nzxp fa htes qfy clyozx dpyepynpd qzc ly
j afcazdp ufde pyepc jzfc alclxpepcd tyez zfc ezzw lyo dpp hszae te nly nzxp fa htes

yeb byzxnyw coxdoxmo qoxobkdyb mkx rovz iye mywo ez gsdr pex bkxnyw coxdoxmoc pyb kx
i zebzyco tecd oxdob iyeb zkbkwodobc sxdy yeb dyyv kxn coo gryzd sd mkx mywo ez gsdr

xda axywmxv bnwcnwln pnwnajcxa ljw qnuy hxd lxvn dy frcq odw ajwmxv bnwcnwlnb oxa jw
h ydayxbn sdbc nwcna hxda yjajvncnab rwcx xda cxxu jwm bnn fqxyc rc ljw lxvn dy frcq

wcz zwxvlwu amvbmvkm omvmzibwz kiv pmtx gwc kwum cx eqbp ncv zivlwu amvbmvkma nwz iv
g xczxwam rcab mvbmz gwcz xiziumbmza qvbw wcz bwwt ivl amm epwxb qb kiv kwum cx eqbp

vby yvwukvt zlualujl nlulyhavy jhu olsw fvb jvtl bw dpao mbu yhukvt zlualujlz mvy hu
f wbywvzl qbza lualy fvby whyhtlalyz puav vby avvs huk zll dovwa pa jhu jvtl bw dpao

uax xuvtjus yktzktik mktkxgzux igt nkrv eua iusk av cozn lat xgtjus yktzktiky lux gt
e vaxvuyk payz ktzkx euax vgxgskzkxy otzu uax zuur gtj ykk cnuvz oz igt iusk av cozn

tzw wtusitr xjsyjshj ljsjwfytw hfs mjqu dtz htrj zu bnym kzs wfsitr xjsyjshjx ktw fs
d uzwutxj ozxy jsyjw dtzw ufwfrjyjwx nsyt tzw yttq fsi xjj bmtuy ny hfs htrj zu bnym

syv vstrhsq wirxirgi kirivexsv ger lipt csy gsqi yt amxl jyr verhsq wirxirgiw jsv er
c tyvtswi nywx irxiv csyv teveqixivw mrxs syv xssp erh wii alstx mx ger gsqi yt amxl

rxu ursqgrp vhqwhqfh jhqhudwru fdq khos brx frph xs zlwk ixq udqgrp vhqwhqfhv iru dq
b sxusrvh mxvw hqwhu brxu sdudphwhuv lqwr rxu wrro dqg vhh zkrsw lw fdq frph xs zlwk

qwt tqrpfqo ugpvgpeg igpgtcvqt ecp jgnr aqw eqog wr ykvj hwp tcpfqo ugpvgpegu hqt cp
a rwtrqug lwuv gpvgt aqwt rctcogvgtu kpvq qwt vqqn cpf ugg yjqrv kv ecp eqog wr ykvj

pvs spqoepn tfoufodf hfofsbups dbo ifmq zpv dpnf vq xjui gvo sboepn tfoufodft gps bo
z qvsqptf kvtu foufs zpvs qbsbnfufst joup pvs uppm boe tff xipqu ju dbo dpnf vq xjui

```
our ropndom sentence generator can help you come up with fun random sentences for an
y purpose just enter your parameters into our tool and see whopt it can come up with

ntq qnomcnl rdmsdmbd fdmdqzsnq bzm gdko xnt bnld to vhsg etm qzmcnl rdmsdmbdr enq zm
x otqonrd itrs dmsdq xntq ozqzldsdqr hmsn ntq snnk zmc rdd vgnos hs bzm bnld to vhsg

msp pmnlbmk qclrclac eclcpyrmp ayl fcjn wms amkc sn ugrf dsl pylbmk qclrclacq dmp yl
w nspnmqc hsqr clrcp wmsp nypykcrcpq glrm msp rmmj ylb qcc ufmnr gr ayl amkc sn ugrf

lro olmkalj pbkqbkzb dbkboxqlo zxk ebim vlr zljb rm tfqe crk oxkalj pbkqbkzbp clo xk
v mromlpb grpq bkqbo vlro mxoxjbqbop fkql lro qlli xka pbb telmq fq zxk zljb rm tfqe

kqn nkljzki oajpajya cajanwpkn ywj dahl ukq ykia ql sepd bqj nwjzki oajpajyao bkn wj
u lqnlkoa fqop ajpan ukqn lwnwiapano ejpk kqn pkkh wjz oaa sdklp ep ywj ykia ql sepd

jpm mjkiyjh nziozixz bzizmvojm xvi czgk tjp xjhz pk rdoc api mviyjh nziozixzn ajm vi
t kpmkjnz epno ziozm tjpm kvmvhzozmn dioj jpm ojjg viy nzz rcjko do xvi xjhz pk rdoc
```

Q.3. Write a program to perform a statistical attack on the following encrypted message

”NBYLY ULY U FIN IZ JINYHNCUF OMYM ZIL IOL LUHXIG MYHNYHWY AYHYLUNIL ZLIG AYNNCHA NBY WLYUNCPY DOCWYM ZFIQCHA NI MYLPCHA UM CHMJCLUNCIH ZIL FSLCWM MWLCJNM IL VLUCHMNILGCHA IZ UHS MILN NBYLY CM HI MBILNUAY IZ OMY WUMYM MIGYNCGYM QY UFF HYYX U ECWEMNULN NI AYN NBIMY WLYUNCPY DOCWYM ZFIQCHA MCHWY SIO HYPYL EHIQ QBUN GUS MJULE SIOL HYRN VLCFFCUHN CXYU NBCM NIIF YRJUHXM IH NBY LUHXIG QILX AYHYLUNIL VS LYNOLHCHA U ZOFF MYHNYHWY”

Assume that Additive Cipher is used for encrypting this message.

In [7]:
```python
def occurances(cipher):
    totals = []
    for letter in key_list2:
        totals.append((cipher.count(letter),letter))
    totals.sort()
    print(totals)
    return totals

# cipher = input("Enter Cipher Text")
cipher = "NBYLY ULY U FIN IZ JINYHNCUF OMYM ZIL IOL LUHXIG MYHNYHWY AYHYLUNIL ZLIG

cipher = cipher.upper()
inputDensity = occurances(cipher)
k = dict2[inputDensity[-1][1]]-dict2["E"]
M = ""

for i in cipher:
    if (i == " "):
        M = M + " "
        continue
    x = dict2[i] - k
    if (x < 0):
        x+=26
    M = M + key_list1[val_list1.index(x)]
print(M)
```

```
[(0, 'K'), (0, 'T'), (2, 'D'), (2, 'R'), (3, 'V'), (4, 'E'), (4, 'P'), (6, 'J'), (6,
'Q'), (6, 'S'), (6, 'X'), (7, 'G'), (8, 'B'), (9, 'O'), (9, 'Z'), (11, 'A'), (11,
'W'), (12, 'F'), (24, 'U'), (25, 'C'), (27, 'M'), (29, 'H'), (31, 'I'), (33, 'N'),
(34, 'L'), (44, 'Y')]
there are a lot of potential uses for our random sentence generator from getting the
creative juices flowing tr serving ops inspiration for lyrics scripts or brainstormi
ng of any sort there is no shortage of use cases sometimes we all need a kickstart t
r get those creative juices flowing since you never know what may spark your next br
illiant idea this tool expands on the random word generator by returning a full sent
ence
```

In [8]:
```
print(inputDensity[-1])
print(inputDensity[-1][1])
print(inputDensity[-1][0])
```

```
(44, 'Y')
Y
44
```