Question 3

This question is to implement some functions useful to performing classical cipher attacks.

1. Implement a Sage function that performs frequency attacks on a mono-alphabetic substitution ciphers. This function should take a ciphertext string compute a histogram of the incidence each letter (ignoring all non alphabet characters.) And return a list of pairs (letter, incidence percentage) sorted by incidence percentage.
2. Implement a Sage function that takes a partial mono-alphabetic substitution and a ciphertext and returns a potential plaintext. The partial mono-alphabetic substitution should be specified as follows: As a 26 character string where the character at position i is the substitution of ith character of the alphabet, OR an underscore ‘\_’ if the corresponding substitution is unknown. The potential plaintext should be the ciphertext with values specified by the mono-alphabetic substitution replaced by the lower-case plaintext. If the corresponding character is unknown (i.e. ‘\_’ in the monoalphabetic substitution cipher) print the cipher text as an uppercase character.)
3. Use your functions from (a) and (b) to decrypt the following cipher text:  
   "ztmn pxtne cfa peqef kecnp cjt tmn zcwsenp ontmjsw ztnws tf wsvp xtfwvfefw, c feb fcwvtf, xtfxevqea vf gvoenwk, cfa aeavxcwea wt wse rntrtpvwvtf wscw cgg lef cne xnecwea eymcg."

Solution to Question 3

This question is to implement some functions useful to performing classical cipher attacks.

1. Implement a Sage function that performs frequency attacks on a mono-alphabetic substitution ciphers. This function should take a ciphertext string compute a histogram of the incidence each letter (ignoring all non alphabet characters.) And return a list of pairs (letter, incidence percentage) sorted by incidence percentage.

def incidence\_pairs\_key(x):

return x[1]

def compute\_incidence(ciphertext):

incidence\_counts = [0 for j in xrange(26)]

alphabetic\_char\_count = 0

for j in xrange(len(ciphertext)):

c = ciphertext[j]

if is\_alphabetic\_char(c):

x = char\_to\_num(c)

incidence\_counts[x] += 1

alphabetic\_char\_count += 1

incidence\_pairs = [(num\_to\_char(j), float(incidence\_counts[j])/float(alphabetic\_char\_count)) \

for j in xrange(26)]

incidence\_pairs.sort(key=incidence\_pairs\_key, reverse=True)

return incidence\_pairs

1. Implement a Sage function that takes a partial mono-alphabetic substitution and a ciphertext and returns a potential plaintext. The partial mono-alphabetic substitution should be specified as follows: As a 26 character string where the character at position i is the substitution of ith character of the alphabet, OR an underscore ‘\_’ if the corresponding substitution is unknown. The potential plaintext should be the ciphertext with values specified by the mono-alphabetic substitution replaced by the lower-case plaintext. If the corresponding character is unknown (i.e. ‘\_’ in the monoalphabetic substitution cipher) print the cipher text as an uppercase character.)

def monoalphabetic\_substitution(text, subs):

outtext = ""

for j in xrange(len(text)):

c = text[j]

oc = c

if is\_alphabetic\_char(c):

x = char\_to\_num(c)

tc = subs[x]

if ('\_' == tc):

oc = c.upper()

else:

oc = tc.lower()

outtext += oc

return outtext

1. Use your functions from (a) and (b) to decrypt the following cipher text:  
   "ztmn pxtne cfa peqef kecnp cjt tmn zcwsenp ontmjsw ztnws tf wsvp xtfwvfefw, c feb fcwvtf, xtfxevqea vf gvoenwk, cfa aeavxcwea wt wse rntrtpvwvtf wscw cgg lef cne xnecwea eymcg."

sage: cipher text = "ztmn pxtne cfa peqef kecnp cjt tmn zcwsenp ontmjsw ztnws tf wsvp xtfwvfefw, c feb fcwvtf, xtfxevqea vf gvoenwk, cfa aeavxcwea wt wse rntrtpvwvtf wscw cgg lef cne xnecwea eymcg."

sage: compute\_incidence(ciphertext)

[('e', 0.12587412587412589),

('w', 0.1048951048951049),

('f', 0.097902097902097904),

('t', 0.097902097902097904),

('c', 0.090909090909090912),

('n', 0.076923076923076927),

('v', 0.062937062937062943),

('a', 0.048951048951048952),

('p', 0.04195804195804196),

('s', 0.04195804195804196),

('x', 0.04195804195804196),

('g', 0.027972027972027972),

('m', 0.027972027972027972),

('z', 0.02097902097902098),

('j', 0.013986013986013986),

('k', 0.013986013986013986),

('o', 0.013986013986013986),

('q', 0.013986013986013986),

('r', 0.013986013986013986),

('b', 0.006993006993006993),

('l', 0.006993006993006993),

('y', 0.006993006993006993),

('d', 0.0),

('h', 0.0),

('i', 0.0),

('u', 0.0)]

sage: #the top two incidences are 'e' and 'w' so we can conclude that these are 'e' and 'w' respectively

sage: subs1 = '\_\_\_\_e\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_t\_\_\_'

sage: monoalphabetic\_substitution(ciphertext, subs1)

'ZTMN PXTNe CFA PeQeF KeCNP CJT TMN ZCtSeNP ONTMJSt ZTNtS TF tSVP XTFtVFeFt, C FeB FCtVTF, XTFXeVQeA VF GVOeNtK, CFA AeAVXCteA tT tSe RNTRTPVtVTF tSCt CGG LeF CNe XNeCteA eYMCG.'

sage: # notice that we have tT, so that T most probably is o

sage: # and a three letter word tSe, so that S = h

sage: # and we have a one letter word C, so that C = a

sage: subs2 = '\_\_a\_e\_\_\_\_\_\_\_\_\_\_\_\_\_ho\_\_t\_\_\_'

sage: monoalphabetic\_substitution(ciphertext, subs2)

'ZoMN PXoNe aFA PeQeF KeaNP aJo oMN ZatheNP ONoMJht ZoNth oF thVP XoFtVFeFt, a FeB FatVoF, XoFXeVQeA VF GVOeNtK, aFA AeAVXateA to the RNoRoPVtVoF that aGG LeF aNe XNeateA eYMaG.'

sage: # we have aNe, so probably N = r

sage: # we also have a three letter aFA, and a two letter word oF

sage: # so probably F = n, and A = d

sage: # also we have aJo, so probably J = g

sage: subs3 = 'd\_a\_en\_\_\_g\_\_\_\_\_\_\_\_ho\_\_t\_\_\_'

sage: monoalphabetic\_substitution(ciphertext, subs3)

'ZoMN PXoNe and PeQen KeaNP ago oMN ZatheNP ONoMght ZoNth on thVP XontVnent, a neB natVon, XonXeVQed Vn GVOeNtK, and dedVXated to the RNoRoPVtVon that aGG Len aNe XNeated eYMaG.'

sage: # we have a word aGG, so probably G = l

sage: # we have a word aNe, so probably N = r

sage: # we have a word neB, so probably B = w

sage: # we also have natVon, and thVP, so probably V = i and P = s

sage: subs4 = 'dwa\_enl\_\_g\_\_\_r\_s\_\_ho\_it\_\_\_'

sage: monoalphabetic\_substitution(ciphertext, subs4)

'ZoMr sXore and seQen Kears ago oMr Zathers OroMght Zorth on this Xontinent, a new nation, XonXeiQed in liOertK, and dediXated to the RroRosition that all Len are Xreated eYMal.'

sage: subs5 = 'dwa\_enl\_\_gymurbsvpho\_itcqf'

sage: monoalphabetic\_substitution(ciphertext, subs5)

'four score and seven years ago our fathers brought forth on this continent, a new nation, conceived in liberty, and dedicated to the proposition that all men are created equal.'