Question 1

The purpose of this question is to implement Sage functions for encryption/decryption with the Caesar cipher, as well as attacks.

1. Implement Sage encryption/decryption functions that take a key (as an integer in 0, 1,2, … , 25), and a string. The function should only operate on the characters ‘a’, ‘b’, … ‘z’ (both upper and lower case), and it should leave any other characters, unchanged.
2. Implement a function that performs a brute force attack on a ciphertext, it should print a list of the keys and associated decryptions. It should also take an optional parameter that takes a substring and only prints out potential plaintexts that contain that decryption.
3. Show the output of your encrypt function (part a) on the following (key, plaintext) pairs:
   * + - k = 6 plaintext = "Get me a vanilla ice cream, make it a double."
       - k = 15 plaintext = "I don't much care for Leonard Cohen."
       - k = 16 plaintext = "I like root beer floats."
4. Show the output of your decrypt function (part a) on the following (key, ciphertext) pairs:
   * + - k = 12 ciphertext = 'nduzs ftq buzq oazqe.'
       - k = 3 ciphertext = "fdhvdu qhhgv wr orvh zhljkw."
       - k = 20 ciphertext = "ufgihxm uly numnys."
5. Show the output of your attack function (part c) on the following ciphertexts, if an optional keyword is specified, pass that to your attack function:
   * + - ciphertext = 'gryy gurz gb tb gb nzoebfr puncry.' keyword = 'chapel'
       - ciphertext = 'wziv kyv jyfk nyve kyv tpdsrcj tirjy.' keyword = 'cymbal'
       - ciphertext = 'baeeq klwosjl osk s esf ozg cfwo lgg emuz.' no keyword

Question 1 (solutions)

The purpose of this question is to implement Sage functions for encryption/decryption with the Caesar cipher, as well as attacks.

1. Implement Sage encryption/decryption functions that take a key (as an integer in 0, 1,2, … , 25), and a string. The function should only operate on the characters ‘a’, ‘b’, … ‘z’ (both upper and lower case), and it should leave any other characters, unchanged.

def CaesarEncrypt(k, plaintext):

ciphertext = ""

for j in xrange(len(plaintext)):

p = plaintext[j]

if is\_alphabetic\_char(p):

x = (k + char\_to\_num(p)) % 26

c = num\_to\_char(x)

else:

c = p

ciphertext += c

return ciphertext

def CaesarDecrypt(k, ciphertext):

plaintext = ""

for j in xrange(len(ciphertext)):

c = ciphertext[j]

if is\_alphabetic\_char(c):

x = (char\_to\_num(c) - k) % 26

p = num\_to\_char(x)

else:

p = c

plaintext += p

return plaintext

1. Implement a function that performs a brute force attack on a ciphertext, it should print a list of the keys and associated decryptions. It should also take an optional parameter that takes a substring and only prints out potential plaintexts that contain that decryption.

def BruteForceAttack(ciphertext, keyword=None):

for k in xrange(26):

plaintext = CaesarDecrypt(k, ciphertext)

if (None==keyword) or (keyword in plaintext):

print "key", k, "decryption", plaintext

return

1. Show the output of your encrypt function (part a) on the following (key, plaintext) pairs:
   * + - k = 6 plaintext = "Get me a vanilla ice cream, make it a double."
       - k = 15 plaintext = "I don't much care for Leonard Cohen."
       - k = 16 plaintext = "I like root beer floats."

sage: k = 6; plaintext = 'Get me a vanilla ice cream, make it a double.'

sage: CaesarEncrypt(k, plaintext)

'mkz sk g bgtorrg oik ixkgs, sgqk oz g juahrk.'

sage: k = 15; plaintext = "I don't much care for Leonard Cohen."

sage: CaesarEncrypt(k, plaintext)

"x sdc'i bjrw rpgt udg atdcpgs rdwtc."

sage: k = 16; plaintext = "I like root beer floats."

sage: CaesarEncrypt(k, plaintext)

'y byau heej ruuh vbeqji.'

1. Show the output of your decrypt function (part a) on the following (key, ciphertext) pairs:
   * + - k = 12 ciphertext = 'nduzs ftq buzq oazqe.'
       - k = 3 ciphertext = "fdhvdu qhhgv wr orvh zhljkw."
       - k = 20 ciphertext = "ufgihxm uly numnys."

sage: k = 12; ciphertext = "nduzs ftq buzq oazqe."

sage: CaesarDecrypt(k, ciphertext)

'bring the pine cones.'

sage: k = 3; ciphertext = "fdhvdu qhhgv wr orvh zhljkw."

sage: CaesarDecrypt(k, ciphertext)

'caesar needs to lose weight.'

sage: k = 20; ciphertext = "ufgihxm uly numnys."

sage: CaesarDecrypt(k, ciphertext)

'almonds are tastey.'

1. Show the output of your attack function (part c) on the following ciphertexts, if an optional keyword is specified, pass that to your attack function:
   * + - ciphertext = 'gryy gurz gb tb gb nzoebfr puncry.' keyword = 'chapel'
       - ciphertext = 'wziv kyv jyfk nyve kyv tpdsrcj tirjy.' keyword = 'cymbal'
       - ciphertext = 'baeeq klwosjl osk s esf ozg cfwo lgg emuz.' no keyword

sage: ciphertext = 'gryy gurz gb tb gb nzoebfr puncry.'

sage: BruteForceAttack(ciphertext, 'chapel')

key 13 decryption tell them to go to ambrose chapel.

sage: ciphertext = 'wziv kyv jyfk nyve kyv tpdsrcj tirjy.'

sage: BruteForceAttack(ciphertext, 'cymbal')

key 17 decryption fire the shot when the cymbals crash.

sage: ciphertext = 'baeeq klwosjl osk s esf ozg cfwo lgg emuz.'

sage: BruteForceAttack(ciphertext)

key 0 decryption baeeq klwosjl osk s esf ozg cfwo lgg emuz.

key 1 decryption azddp jkvnrik nrj r dre nyf bevn kff dlty.

key 2 decryption zycco ijumqhj mqi q cqd mxe adum jee cksx.

key 3 decryption yxbbn hitlpgi lph p bpc lwd zctl idd bjrw.

key 4 decryption xwaam ghskofh kog o aob kvc ybsk hcc aiqv.

key 5 decryption wvzzl fgrjneg jnf n zna jub xarj gbb zhpu.

key 6 decryption vuyyk efqimdf ime m ymz ita wzqi faa ygot.

key 7 decryption utxxj dephlce hld l xly hsz vyph ezz xfns.

key 8 decryption tswwi cdogkbd gkc k wkx gry uxog dyy wemr.

key 9 decryption srvvh bcnfjac fjb j vjw fqx twnf cxx vdlq.

key 10 decryption rquug abmeizb eia i uiv epw svme bww uckp.

key 11 decryption qpttf zaldhya dhz h thu dov ruld avv tbjo.

key 12 decryption posse yzkcgxz cgy g sgt cnu qtkc zuu sain.

key 13 decryption onrrd xyjbfwy bfx f rfs bmt psjb ytt rzhm.

key 14 decryption nmqqc wxiaevx aew e qer als oria xss qygl.

key 15 decryption mlppb vwhzduw zdv d pdq zkr nqhz wrr pxfk.

key 16 decryption lkooa uvgyctv ycu c ocp yjq mpgy vqq owej.

key 17 decryption kjnnz tufxbsu xbt b nbo xip lofx upp nvdi.

key 18 decryption jimmy stewart was a man who knew too much.

key 19 decryption ihllx rsdvzqs vzr z lzm vgn jmdv snn ltbg.

key 20 decryption hgkkw qrcuypr uyq y kyl ufm ilcu rmm ksaf.

key 21 decryption gfjjv pqbtxoq txp x jxk tel hkbt qll jrze.

key 22 decryption feiiu opaswnp swo w iwj sdk gjas pkk iqyd.

key 23 decryption edhht nozrvmo rvn v hvi rcj fizr ojj hpxc.

key 24 decryption dcggs mnyquln qum u guh qbi ehyq nii gowb.

key 25 decryption cbffr lmxptkm ptl t ftg pah dgxp mhh fnva.