program that can perform a letter frequency attack on an additive cipher without human intervention. Your software should produce possible plaintexts in rough order of likelihood. It would be good if your user interface allowed the user to specify “give me the top 10 possible plaintexts.”

from collections import Counter

import math

EN\_FREQ = [8.167,1.492,2.782,4.253,12.702,2.228,2.015,6.094,6.966,0.153,0.772,4.025,2.406,6.749,7.507,1.929,0.095,5.987,6.327,9.056,2.758,0.978,2.361,0.150,1.974,0.074]

def shift\_decrypt(text, shift):

out = []

for ch in text:

if 'A' <= ch <= 'Z':

out.append(chr((ord(ch)-65-shift) % 26 + 65))

elif 'a' <= ch <= 'z':

out.append(chr((ord(ch)-97-shift) % 26 + 97))

else:

out.append(ch)

return ''.join(out)

def chi\_squared(text):

letters = [c.upper() for c in text if c.isalpha()]

N = len(letters)

if N == 0:

return float('inf')

obs = Counter(letters)

chi2 = 0.0

for i,pct in enumerate(EN\_FREQ):

letter = chr(65+i)

expected = pct/100.0 \* N

observed = obs.get(letter,0)

exp = expected if expected>1e-8 else 1e-8

chi2 += (observed-expected)\*\*2/exp

return chi2

def attack(ciphertext, top\_n=10):

candidates = []

for shift in range(26):

plain = shift\_decrypt(ciphertext, shift)

score = chi\_squared(plain)

candidates.append((shift, score, plain))

candidates.sort(key=lambda x: x[1])

return candidates[:top\_n]

if \_\_name\_\_ == "\_\_main\_\_":

ct = input("Enter ciphertext:\n")

tn = input("Top how many candidates? (default 10): ").strip()

try:

top\_n = int(tn) if tn else 10

except:

top\_n = 10

results = attack(ct, top\_n)

print(f"{'Rank':>4} {'Shift':>5} {'Score':>10} Plaintext")

for i,(shift,score,plain) in enumerate(results,1):

snippet = plain if len(plain)<=100 else plain[:97]+"..."

print(f"{i:>4} {shift:>5} {score:10.3f} {snippet}")

