Checkpoint 1 — Caesar Cipher (Brute Force)

Objective

Break the given Caesar cipher by trying all possible shifts (0–25).

Problem

Cipher text: **odroboewscdrolocdcwkbdmyxdbkmdzvkdpybwyeddrobo**

Task: Write a program to decrypt this Caesar cipher and recover the original plaintext.

Method:

1. A Caesar cipher shifts every letter by a fixed number. Example with shift 3: A→D, B→E, C→F.

2. There are only 26 possible shifts, so I try them all (0 to 25).

3. For each shift I decrypt and print the result, then pick the line that makes real English.

Formula I used (in my C++ code):

decrypted_char = ((cipher_char - 'a' - shift + 26) % 26) + 'a';

+26 keeps the value positive when wrapping around the alphabet.

Result:

Shift 0: odroboewscdrolocdcwkbdmyxdbkmdzvkdpybwyeddrobo
Shift 1: ncqnandvrbcqnknbcbvjaclxwcajlcyujcoxavxdccqnan
Shift 2: mbpmzmcuqabpmjmabauizbkwvbzikbxtibnwzuwcbbpmzm
Shift 3: laolylbtpzaolilzazthyajvuayhjawshamvytvbaaolyl
Shift 4: kznkxkasoyznkhkyzysgxziutzxgizvrgzluxsuazznkxk
Shift 5: jymjwjzrnxymjgjxyxrfwyhtsywfhyuqfyktwrtzyymjwj
Shift 6: ixliviyqmwxlifiwxwqevxgsrxvegxtpexjsvqsyxxlivi
Shift 7: hwkhuhxplvwkhehvwvpduwfrqwudfwsodwiruprxwwkhuh
Shift 8: gvjgtgwokuvjgdguvuoctveqpvtcevrncvhqtoqwvvjgtg
Shift 9: fuifsfvnjtuifcftutnbsudpousbduqmbugpsnpvuuifsf
Shift 10: ethereumisthebestsmartcontractplatformoutthere

Shift 11: dsgdqdtlhrsgdadrsrlzqsbnmsqzbsokzsenqlntssgdqd
Shift 12: crfcpcskgqrfczcqrqkypramlrpyarnjyrdmpkmsrrfcpc
Shift 13: bqebobrjfpqebybpqpjxoqzlkqoxzqmixqclojlrqqebob
Shift 14: apdanaqieopdaxaopoiwnpykjpnwyplhwpbknikqppdana
Shift 15: zoczmzphdnoczwznonhvmoxjiomvxokgvoajmhjpooczmz
Shift 16: ynbylyogcmnbyvymnmgulnwihnluwnjfunzilgionnbyly
Shift 17: xmaxkxnfblmaxuxlmlftkmvhgmktvmietmyhkfhnmmaxkx
Shift 18: wlzwjwmeaklzwtwklkesjlugfljsulhdslxgjegmllzwjw
Shift 19: vkyvivldzjkyvsvjkjdriktfekirtkgcrkwfidflkkyviv
Shift 20: ujxuhukcyijxuruijicqhjsedjhqsjfbqjvehcekjjxuhu
Shift 21: tiwtgtjbxhiwtqthihbpgirdcigprieapiudgbdjiiwtgt
Shift 22: shvsfsiawghvspsghgaofhqcbhfoqhdzohtcfacihhvsfs
Shift 23: rgurerhzvfgurorfgfznegpbagenpgcyngsbezbhggurer
Shift 24: qftqdqgyueftqnqefeymdfoazfdmofbxmfradyagfftqdq
Shift 25: pespcpfxtdespmpdedxlcenzyeclneawleqzcxzfeespcp

Checkpoint 2 — Substitution Ciphers

Objective

Break two substitution ciphers using letter frequency analysis and mapping.

What is a substitution cipher

Each plaintext letter is replaced by another letter (a 26-letter permutation). It is not a fixed shift like Caesar. The weakness: it does not hide the frequency pattern of letters, so we can use English letter frequencies to guess.

English letter frequencies (used idea from the manual table)

Common letters are E (~12%), T (~10%), A (~8%), O (~7–8%). Rare letters include J, Q, X, Z. I used this idea to match cipher letters to likely English letters by rank.

Method (what my notebook does)

1. **Count letters** in the cipher (ignore spaces/punctuation).
2. **Sort** cipher letters from most frequent to least.
3. **Map by rank** to English frequency order (E, T, A, O, I, N, …).
4. **Decrypt** the text with this mapping and print the result.
5. (If needed) small manual fixes could be added, but my submitted code is automatic.

This follows the lab's frequency-analysis guidance.

Code summary

- `calculate_frequency(text)`: counts letters and computes percentages.
- `create_mapping(freq)`: builds cipher→plain mapping by frequency rank.
- `decrypt(text, mapping)`: replaces letters; keeps spaces and punctuation.
- `break_cipher(...)`: prints top-10 cipher frequencies, builds mapping, decrypts, shows result for Cipher-1 and Cipher-2.

Why it works

Substitution ciphers leak letter statistics. With enough text, the cipher's frequency shape looks like English, so rank-based mapping gives a readable plaintext.

Which was easier

Cipher-2 is usually easier because it is longer. More text → better frequency accuracy → better automatic mapping. (Longer texts more closely match standard English frequencies.)

How I ran it

- Environment: **Google Colab**, **Python** notebook.
- The notebook prints the top-10 cipher letter frequencies and the decrypted text for both ciphers.