Encryption and decryption of MTP:

File : key_mtp.py

Generates a cryptographically secure key of 2 power 20 bits and stores it in a file named otp_secret

File : encrypt_mtp.py

python encrypt_mtp.py plaintext.txt ciphertext.txt
Reads OTP from the otp_secret file. Reads plaintext. The otp_secret is sliced to match the number of bits in the plaintext.
Writes the result of otp_secret xor plaintext to ciphertext

File : decrypt_mtp.py
python decrypt_mtp.py ciphertext.txt viola.txt
Reads OTP from otp_secret file. Reads ciphertext
Writes result of otp_secret xor ciphertext to viola.txt

Problem 2b:

Tries to deduce the location at which space would be present in the plaintext. Once we get the location we can use properties of xor to retrieve a part of the key.
Assuming we have 10 cipher texts encrypted using same key (MTP)
c1,c2,....,c10

-> Since c1 = b1 xor key, c2 = b2 xor key etc. If we xor c1 and c2 we get a part of plaintext in form of b1 xor b2.
-> Prepare a list of values we could get if we xor english alphabets with ascii value of space.
-> For discovering possible space locations in b1,
- c1 xor c2, c1 xor c3, c1 xor c4 ..... c1 xor c10
-Compare the results with the list prepared above.
- If we find a match keep track of it by incrementing the counter of that value. This is done as there can be different combinations that would result in values given in the list.
- Once we have all possible locations, we ignore the one's with low threshhold(Some % of number of CTs available). This is done so that other possible combinations can be ignored.
- With the remaining locations, each location of cipher text c1 is xored with space to iteratively discover a part of key.
-> The above procedure is repeated for all ciphertexts available.