CSCI 357 - Cryptography
Ryan Kulyassa
HW #4 - Vigenère cipher implementation

For my encryption function, I start by iterating through the plaintext. Since the key may be shorter than the plaintext, we have to index the characters of the key by taking the mod of our current iteration with the length of the key (`key[i % len(key)]`), which essentially cycles through it. Then, if the plaintext character is a letter (skip otherwise), we proceed with the encryption, finding the alphabetic index of the plaintext char and handling uppercase/lowercase.

Handling the key is a little less trivial, as in the assignment we use the specific key "CSCI357", and the numbers have to be handled differently than the alphabetic characters. My approach is to just take the raw ordinal of the numbers (`ord(key_char)`), meanwhile applying the usual uppercase/lowercase offset logic for alphabetic characters. There are other approaches that can be used here, such as evaluating the numbers literally (0 maps to 0, 1 to 1, etc.), but in my opinion this is ultimately an arbitrary choice since the numbers 0-9 are still going to produce a deterministic mapping. Furthermore, it's my understanding that most traditional implementations of the Vigenère cipher do not allow for non-alphabetic characters in the key, so there isn't much precedent to follow either.

Finally, we get the shifted character by adding the alphabetic indices of both plaintext and key characters mod 26, which is the core feature of the Vigenère cipher.

Decryption starts by iterating and converting characters to alphabetic indices the same way. The only difference is, when converting to the plaintext character index, we subtract the key's alphabetic index rather than adding it, which I like to think of as shifting "down" rather than shifting "up" as when encrypting.