1. In the beginning, I import the txt files, then split the characters by using flatmap lambda function. I will get all the characters, these include whitespaces, punctuations, numbers, and alphabets. I do not alter any symbols in the original text file, the only thing I change is the alphabet by using shift key function I build for later use. After getting all the characters, I filter out the whitespaces, and count how many alphanumeric and punctuation symbols in the text files. Sort the count by doing sortBy lambda function, and do it in ascending way. Since it is ascending, the first element must be the most common letters in the ciphertext. According to English language, the letter ‘e’ is the most common letter. I assume the most common letter in ciphertext is equivalent to ‘e’ after decrypted, so I use ascii value to find the potential shift key by doing the most common letter in ciphertext index in ascii minus 65. Since we are using an upper class, ascii table for ‘A’ stars from 65. Now we get the key, we can start decrypting the ciphertext. I simply used a for loop to shift every alphabet to where it supposes to be. Inside a for loop, I have a nested if statement. If statement is to make sure numbers and punctuation will not get altered. Print out the value, and now we have a decrypted text file. I used “langdetect” extension to check if the text is English language in the end, and also to make sure if the text file is properly decrypted.

2. Finding the key function

Find the shifting key

In [11]:
```
index_of_most_common_letter = 4

charsCountSorted = charsCount.sortBy(lambda a: a[1], ascending=False)

def find_Common_Letter_in_cipher(x):

    commonLetter = ''

    for (alphabet, frequency) in charsCountSorted.take(1):

        commonLetter = alphabet

        return commonLetter

    print(commonLetter)

common_letter_in_cipher = find_Common_Letter_in_cipher(charsCountSorted)

common_Letter_in_cipher_Index = ord(common_letter_in_cipher) - 65

key = common_Letter_in_cipher_Index - index_of_most_common_letter
```

Decrypt function

Decrypt the txt file

In [18]:
```
def Decrypt(message, key):

    translated = ""

    for symbol in message.collect():

        if symbol.isalpha():
            num = ord(symbol)
            num -= key

            if num < 65:
                num += 26

            translated += chr(num)

        else:
            translated += symbol
    return translated

decryptData = Decrypt(chars, key)
print(decryptData)
```

3. All 3 txt files decrypted without any issue by using this method. By using the “langdetect”, the average percentage of showing is English language is around 98%