MI Assignment(0)

Problem (5) Idea

Name	Basma Hatem Elhoseny
Sec	1
BN	16
Code	9202381

Basic Implementation:

As the brute force has tried all the 26 possbile solutions this causes time limit to be exceeded no

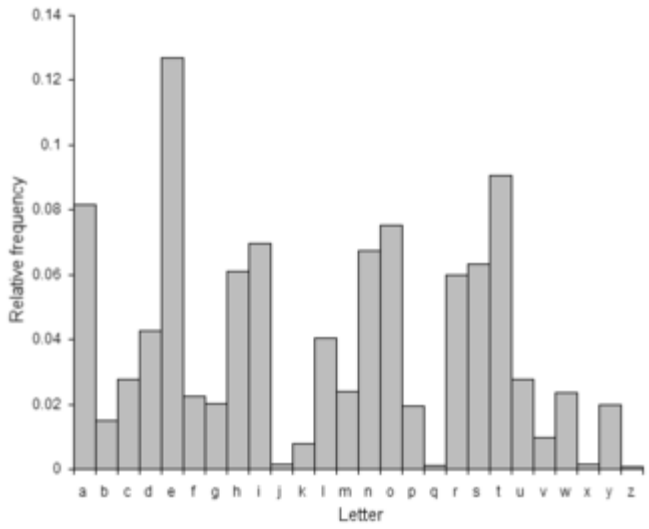
```
    for i in range(26): # Brute Force
    # for i in possible_shifts:
        result = ''.join([chr(((ord(char) - ord('a') - i) % 26) + ord('a'))
                          if ('a' <= char <= 'z') else (char) for char in ciphered])
        count_non_intersect = 0
        non_intersecting_words = [
            word for word in result.split() if word not in dictionary]
        count_non_intersect = len(non_intersecting_words)

        if (count_non_intersect < min_non_intersect_count):
            min_non_intersect_count = count_non_intersect
            dechiper = result
            shift = i
    return (dechiper, shift, min_non_intersect_count)
```

Enhancement (1)

Apply statistical analysis on the cipher and pick the most frequent character

And according to the frequency of English letters we can pick that the most frequent letters are **e-t-a-o**

So I just try these possible letters to be the decipher of the most frequent letter so instead of having 26 trial now we have only 4 😉😉




```
possible_shifts = [(ord(max_char) - ord(ch)) %
                   26 for ch in ['e', 't', 'a', 'o']]
```

```
# for i in range(26): # Brute Force
for i in possible_shifts: # 4 or 1 possible shifts
    result = ''.join([chr(((ord(char) - ord('a') - i) % 26) + ord('a'))
                      if ('a' <= char <= 'z') else (char) for char in ciphered])
    count_non_intersect = 0
    non_intersecting_words = [
        word for word in result.split() if word not in dictionary]
    count_non_intersect = len(non_intersecting_words)

    if (count_non_intersect < min_non_intersect_count):
        min_non_intersect_count = count_non_intersect
        dechiper = result
        shift = i
return (dechiper, shift, min_non_intersect_count)
```

Enhancement (2)

This works well till the last test case in which the cipher text is too long so having 4 trails gives us timeout, So here is the trick we are sad that the cipher text is too long 😢 No, let' make use of this in English language the only word of 1 letter is **a** so as the text is longer of course it is more probable to have the word a so once I find a word of 1 character in the cipher text of course it is a so I have 1 solution here which is to calculate the shift a from this letter to a then use this shift(key) to decrypt the rest of the cipher text by only 1 possible char

```
if (a_candidate != -1):
    # Only 1 possible shift which is mapping thus alone letter to a :D
    possible_shifts = [(ord(a_candidate) - ord(ch)) %
                       26 for ch in ['a']]
```