Scope of Variables

Those variables whose name starts with ‘g_’ are global variables. As for the parameters in function’s parenthesis, they are local variables. Besides, those variables defined in functions are also local variables.

Here I make a list:
global variables:
g_word_list, g_sentence_list, g_length, g_length, g_punct, g_longest, g_shortest
local variables: for variables not in global variables. e.g. p_x…

Goal

Read the information in the “sentence.txt” file without whitespace. Use an algorithm to restore the lost whitespaces and write the ouput into a new text.

Idea of solving this problem

Fisrt, I read the information in word.txt and sentence.txt and restore them into two lists. Meanwhile, I convert the format of numbers in word_list. After that, I want to store the word or the punctuation along with its position index in a list. Then I sort the list by the position index. After sorting, I combine the word or punctuations together. At this time, I divide the situation into several probable conditions and deal with them with individual way, especially processing the combination of punctuations. After combining, the last step is to write every single sentence into a ouput text file.

Structure

In my assignment 2, I define 3 functions as below:

```
def search(p_x):…
def combine(p_x):…
def main():…
```

search(p_x)
=> find every word or punctuation along with their position in one line, return two list of tuples. The format of tuples is like (position index in the seached line, word/punctuation).

```
p_list_sub = list()
for i in g_punct:
    n = 0
    while i in p_x[n:]:
        if i == "'":
            if p_x[p_x.index(i)-1] == 'e':
                pass
            else:
                pos = p_x.index(i,n)
        else:
            pos = p_x.index(i,n)
            if pos < len(p_x)-2:
                if p_x[pos] != ',' or p_x[pos+2] != '0':
                    p_list_sub.append([pos, p_x[pos]])
            else:
                p_list_sub.append([pos, p_x[pos]])
        n = pos + 1
```

```
p_list_sub2 = list()
m = 0
while m < len(p_x):
    in_list = False
    if len(p_x) - m -1 < g_longest:
        L = len(p_x) - m
    else:
        L = g_longest
    for i in range(g_shortest,L+1):
        if p_x[m:m+i].lower() in g_word_list:
            in_list = True
            w = p_x[m:m+i]
    if in_list == True:
        p_list_sub2.append([m, w])
```

The left picture shows the generation of the punctuation list containing tuples of the punctuation and its postition.
The right picture shows the generation of the word list containing tuples of the word and its positions.

combine(p_x)
=> since the word/punctuation can connected with its position, then I combine the two lists(one is for word and the other is punctuation) together into a new list. Then sort this new list. After that, I add proper whitespaces while joining all the words or punctuations one by one. Finally return the line with proper whitespaces.

```
line = ''
l1, l2  = search(p_x)
l3 = l1 + l2
l = sorted(l3)
s = ''
for i in l:
    s += i[1]
if s in g_sentence_list:
    for i in range(len(l)-1):
        if l[i][1] in ['-']:
            line += l[i][1]
        elif l[i][1] in ['(']:
            line += ' ' + l[i][1]
        elif l[i][1] in ['"',"'"] or l[i+1][1] in g_punct:
            line += l[i][1]
        else:
            line += l[i][1] + ' '
    line += l[-1][1]
else:
    line = s
return line
```

The above is dealing with the whitespaces of punctuations and add all items up into a line restored with proper whitespaces.

main()
=> Here is the main body of the code. It reads the information in word.txt and sentence.txt and store the information in g_word_list and g_sentence_list individually. At last, for each item(that is to say: line) in g_sentence_list, transfer the previous functions to get a proper line and write it into “output.txt”.

```
with open('word.txt','r') as f:
    for word in f:
        word = word.strip('\n')
        if word[-1] == '0':
            if len(word) == 4:
                word = word[0]+','+word[1]+word[2]+word[3]
            elif len(word) == 5:
                word = word[0]+word[1]+','+word[2]+word[3]+word[4]
        g_word_list.append(word)
```

The explanation of above picture: Read the information in word.txt and convert the number’s format(e.g. 5000 => 5,000). Generate a g_word_list containing the word information, every single word or number is considered as an item in this list.

```
with open('sentence.txt','r') as fr:
    for l in fr:
        l = l.strip('\n')
        g_sentence_list.append(l)
```

Left: Likewise, generate a g_sentence_list.

```
output = open('output.txt','w')
for i in g_sentence_list:
    output.write(combine(i)+'\n')
output.close()
```

The above: write every new line into the output.txt. Close the output file.