Webscraping the script of Neon Genesis Evangelion

In [1]:
```
import pandas as pd
import bs4 as bs
import requests
import re
import time
```

In this notebook, I'll be pulling scripts from each episode of Neon Genesis Evangelion using the BeautifulSoup and Requests packages and parsing through for each separate line of dialogue using regular expressions. I created two separate dictionaries to track the number of times each character speaks and their word count. The first dictionary covers character lines and word count for the entire series, and the second is a dictionary of dictionaries which contains the same data, but for a given episode and the characters in that episode.

To get the scripts I'll be using for this analysis, I'll be pulling from scripts available online at https://www.animanga.com/scripts/anime_scripts_english.html. The urls follow a pattern in naming conventions, so I can pull each episode script sequentially with a loop.

In [2]:
```
character_lines = {}
lines_by_episode = {}


for i in range(1,27):
    current_ep = {}

    webpage = requests.get(f'https://www.animanga.com/scripts/textesgb/eva{i}.html')
    soup = bs.BeautifulSoup(webpage.text, 'lxml')
    page_text = soup.get_text()

    #-------------------------------------------------------------------------

    regex = re.compile(r"(?m).+:[^*#:]+\n") # regular expression to match each separate
    result = re.findall(regex,page_text)

    #-------------------------------------------------------------------------

    # iterating through each line of text, creating a key and word count
    # if character not already in character_lines, otherwise adding to the existing entr
    for text in result:
        character_name = re.findall('[^:#]*',text)[0] # regex to pull the name of the ch
        character_name = character_name.strip('!"#$%&(*+, ''-./:;<=>?@[\]^_`{|}~') # str
        word_count_line = len(text.split())-1
        if character_name in character_lines:
            character_lines[character_name][0] += 1
            character_lines[character_name][1] += word_count_line
        else:
            character_lines[character_name] = [1,word_count_line]

    for text in result:
        character_name = re.findall('[^:#]*',text)[0] # regex to pull the name of the ch
        character_name = character_name.strip('!"#$%&(*+, ''-./:;<=>?@[\]^_`{|}~') # str
        word_count_line = len(text.split())-1
        if character_name in current_ep:
            current_ep[character_name][0] += 1
            current_ep[character_name][1] += word_count_line
```

```
        else:
            current_ep[character_name] = [1,word_count_line]

    #-------------------------------------------------------------------------

    lines_by_episode[i] = current_ep

    # there shouldn't be a risk of overloading the server, but added a short wait in bet
    time.sleep(1)
```

In [3]:
```
list(character_lines.keys())[0:25]
```

Out[3]:
```
['Neon Genesis Evangelion (Japanese title',
 'Nadia',
 'Email',
 'http',
 'Radio',
 'Misato',
 'Phone',
 'Shinji',
 '',
 'Female Voice',
 'Male Voice',
 'Fuyutsuki',
 'Gendo',
 'EPISODE',
 'Aircraft',
 'Sub Commander A',
 'Sub Commander B',
 'One of Cmdr',
 'Sub Cmdr A',
 'Sub Cmdr B',
 'Cmdr-in-Chief',
 'Commanders',
 'Misato (thinking)',
 'Cmdr',
 'Announce']
```

Looking through the compiled list of characters and lines, there are clearly a lot of typos, ranging from misspelled names, to translator notes being included, to whitespace or punctuation causing errors in dialogue attribution.

The order these issues will be tackled in:

1. Done in the previous step: leading punctuation and whitespace was stripped so that names with errors (ex: 'Shinji' and ' Shinji ') are counted for the same key in each dictionary.

2. Splitting up lines where multiple characters are speaking at the same time and attributing the line to each character individually. There are occasionally other ways the translators denote multiple characters speaking but splitting with '&' was the most common. Looped through each episode and also separately for the entire series.

3. Removing "names" which only appear due to being in the translator notes on each page.

4. Spelling errors (in excel, not in this notebook)

1. Splitting names and adding values

In [4]:
```
# Before:
print(f'Before splitting and adding for Shinji:', character_lines['Shinji'], '- (format:
```

```
# for each episode
for i in range(1,27):
    shared_lines = [[key,val] for key, val in lines_by_episode[i].items() if re.search('

    for item in shared_lines:
        item[0] = item[0].split('&')
        word_count_line = item[1][1]
        for names in item[0]:
            if names.strip() in lines_by_episode[i]:
                lines_by_episode[i][names.strip()][0] += 1
                lines_by_episode[i][names.strip()][1] += word_count_line
            else:
                lines_by_episode[i][names.strip()] = [1,word_count_line]

# for entire series
shared_lines = [[key,val] for key, val in character_lines.items() if re.search('&', key)

for item in shared_lines:
    item[0] = item[0].split('&')
    item_shared_lines = item[1][0]
    item_shared_words = item[1][1]
    for names in item[0]:
        if names.strip() in character_lines:
            character_lines[names.strip()][0] += item_shared_lines
            character_lines[names.strip()][1] += item_shared_words
        else:
            character_lines[names.strip()] = [item_shared_lines,item_shared_words]

print(f'After:', character_lines['Shinji'])
```

```
Before splitting and adding for Shinji: [915, 6854] - (format: [linecount, wordcount])
After: [933, 6917]
```

Now each time a line was spoken by multiple characters (ex: "Shinji & Asuka"), each character receives credit for the spoken line in their original dictionary key.

2. Sorting dictionaries and removing extraneous keys

In [5]:
```
# entire series: dictionary - {character name:[line count, word count]}
sorted_lines = dict(sorted(character_lines.items()))

# individual episodes: dictionary of dictionaries - {episode number: {character name:[li
sorted_eps = {}
for i in range(1,27):
    sorted_eps[i] = dict(sorted(lines_by_episode[i].items()))

    #-------------------------------------------------------------------------

# filtering out common translator notes not in the script
filter = ['Neon','EVA','Email','E-mail','http','title','episode','Episode','EPISODE','Na

for i in range(1,27):
    sorted_eps[i] = {k:v for k, v in sorted_eps[i].items() if not any(x in k for x in fi

sorted_lines = {k:v for k, v in sorted_lines.items() if not any(x in k for x in filter)}

    #-------------------------------------------------------------------------

# converting to dataframe and exporting to excel sheet
df = pd.DataFrame(data=sorted_lines).T
df.columns = ['Linecount','Wordcount']
df.to_excel('src/NGE_entire_series_lines.xlsx')
```

```
df1 = pd.DataFrame(data=sorted_eps)
df1 = df1
df1.to_excel('src/NGE_lines_by_episode.xlsx')
```

The dictionaries are now sorted and converted into pandas DataFrames, and exported to Excel files for further analysis.