Homework 4. Frequent Words and Web scraping

Double Click here to edit this cell

- Name: 김동규
- Student ID: 201800615
- Submission date: 2023/5/18

Problem 1 (15 pts)

- Project Gutenberg is a volunteer effort to digitize and archive cultural works.
- Moby-Dick is an 1851 novel by American writer Herman Melville.
- You can find Moby-Dick in an ordinary text format at https://www.gutenberg.org/files/2701/old/moby10b.txt (https://www.gutenberg.org/files/2701/old/moby10b.txt)
- Use **requests** module to get the text.

- We want to compute word frequency of words appearing in mobydick and generate WordCloud
  - First, you must split the text into words.
  - **Any symbols(!, ., ?, ,, +, -, *, ...)** are delimeters
  - Numbers should not be words.
  - Null string is not a word.
  - Any delimiters should not be words.
  - To split into words, use **re** (regular expression module)
  - (Upper or lower) Cases does not matter in words

1.1 Print top 50 most common words (5 pts)

입力 [1]:

```
# YOUR CODE MUST BE HERE
import requests
import re, sys
from collections import Counter
import matplotlib.pyplot as plt

url = 'https://www.gutenberg.org/files/2701/old/moby10b.txt'
response = requests.get(url)
text = response.text

word_pattern = re.compile(r'₩b([a-zA-Z]+)₩b')

words = re.findall(word_pattern, text.lower())

word_freq = {}

for word in words:
    if word in word_freq:
        word_freq[word] += 1
    else:
        word_freq[word] = 1

top_words = sorted(word_freq.items(), key=lambda x: x[1], reverse=True)[:50]

word_freq_arr = [(word, freq) for word, freq in top_words]

print(word_freq_arr)
```

```
[('the', 14512), ('of', 6676), ('and', 6471), ('a', 4774), ('to', 4690), ('in', 4
190), ('that', 3095), ('it', 2542), ('his', 2530), ('i', 2128), ('he', 1896), ('b
ut', 1823), ('s', 1811), ('as', 1750), ('is', 1748), ('with', 1729), ('was', 164
7), ('for', 1643), ('all', 1537), ('this', 1437), ('at', 1332), ('by', 1232), ('w
hale', 1228), ('not', 1162), ('from', 1103), ('on', 1077), ('so', 1073), ('him',
1067), ('be', 1058), ('you', 949), ('one', 934), ('there', 870), ('now', 787),
('had', 779), ('have', 773), ('or', 761), ('were', 685), ('they', 669), ('which',
650), ('like', 648), ('me', 634), ('then', 632), ('some', 621), ('what', 620),
('their', 620), ('are', 611), ('when', 608), ('an', 600), ('no', 592), ('my', 58
9)]
```

Your output should be like the following:

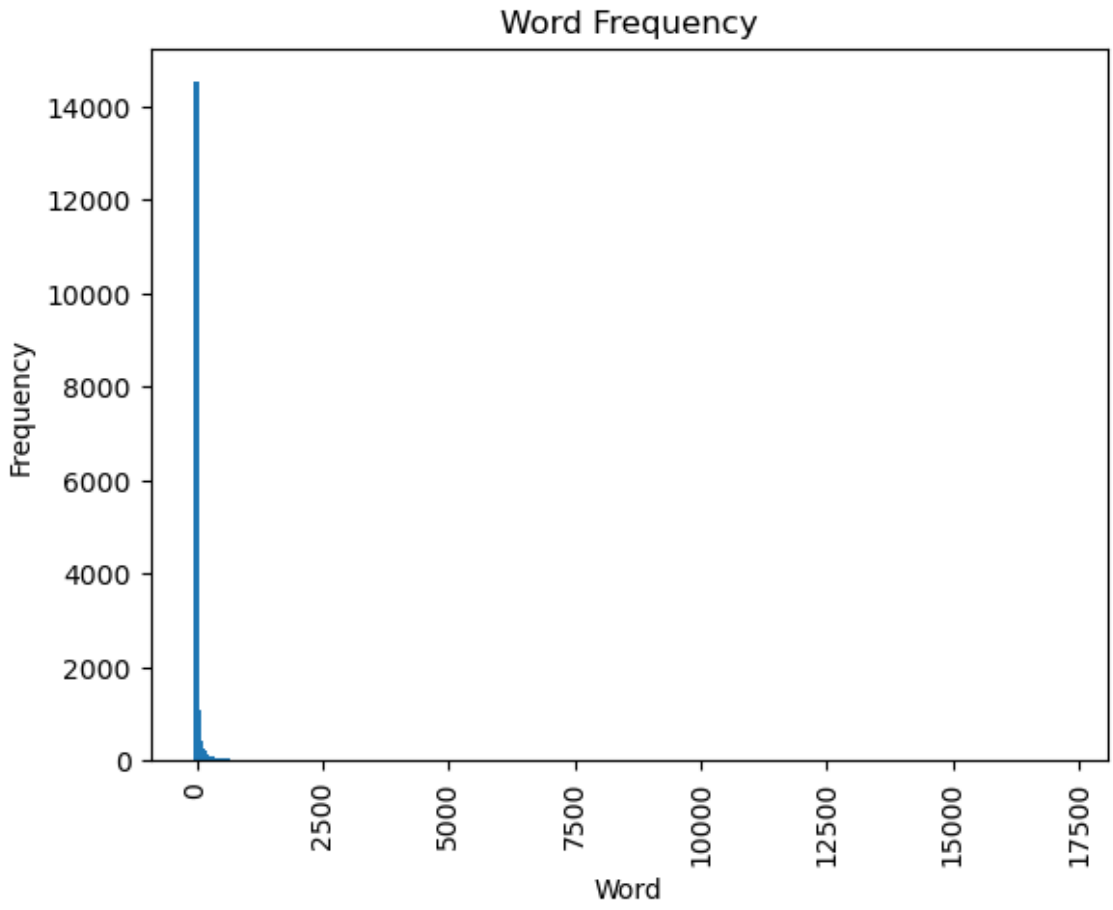
```
[('the', 14512), ('of', 6676), ('and', 6471), ('a', 4774), ('to', 4690), ('in', 4190),
('that', 3095), ('it', 2542), ('his', 2530), ('i', 2128), ('he', 1896), ('but', 1823),
('s', 1811), ('as', 1750), ('is', 1748), ('with', 1729), ('was', 1647), ('for', 1643),
('all', 1537), ('this', 1437), ('at', 1332), ('by', 1232), ('whale', 1228), ('not', 116
2), ('from', 1103), ('on', 1077), ('so', 1073), ('him', 1067), ('be', 1058), ('you', 94
9), ('one', 934), ('there', 870), ('now', 787), ('had', 779), ('have', 773), ('or', 76
1), ('were', 685), ('they', 669), ('which', 650), ('like', 648), ('me', 634), ('then',
632), ('some', 621), ('what', 620), ('their', 620), ('are', 611), ('when', 608), ('an',
600), ('no', 592), ('my', 589)]
```

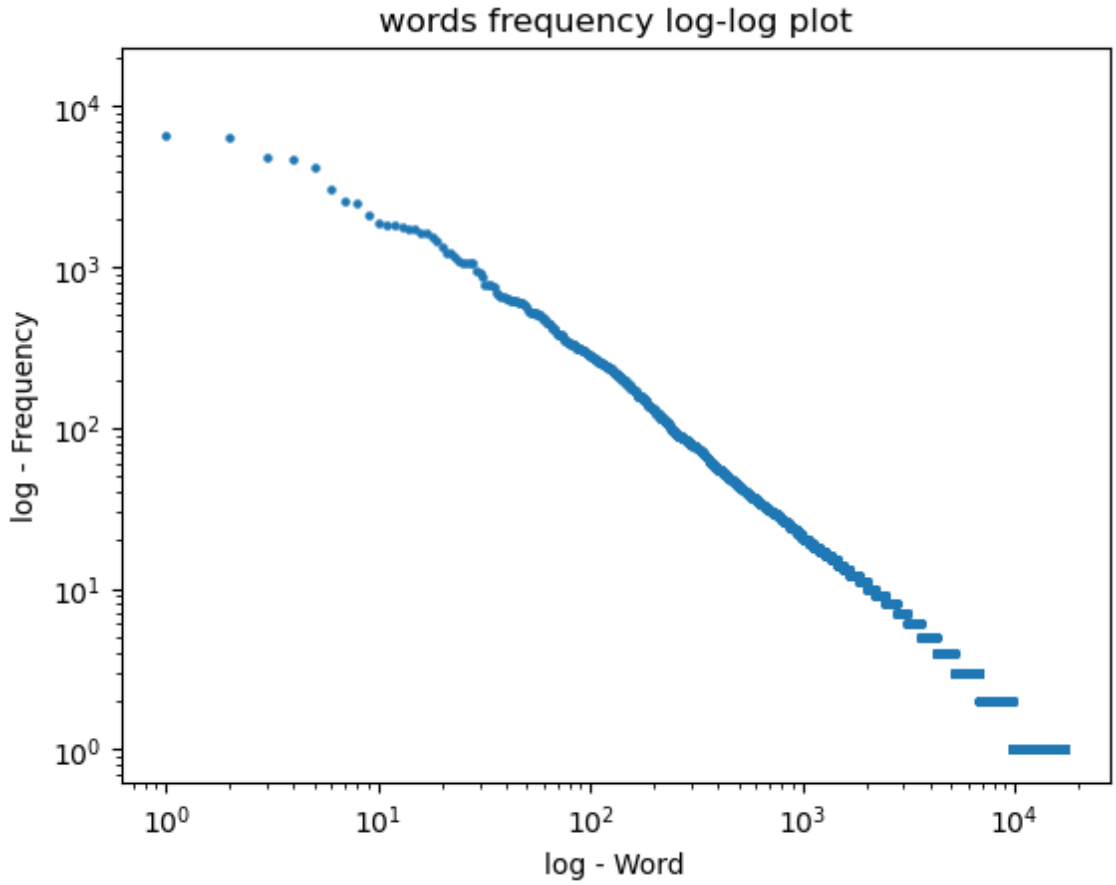
1.2 Plot word frequency (5 pts)

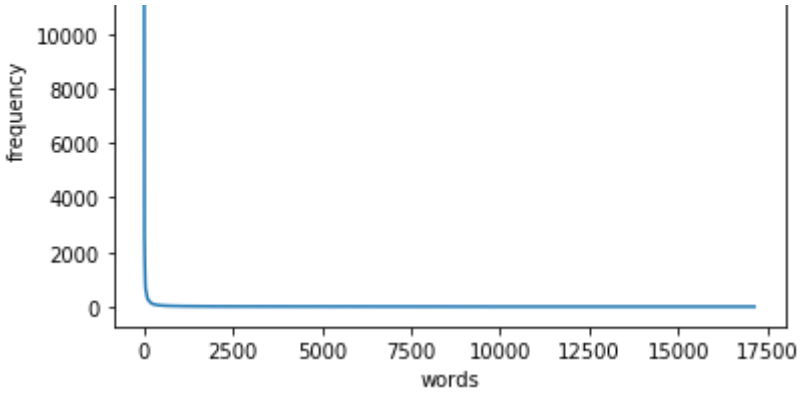
- Sort the word frequency in descending order
- Plot the word frequency
- Plot the word frequency in log-log plot.

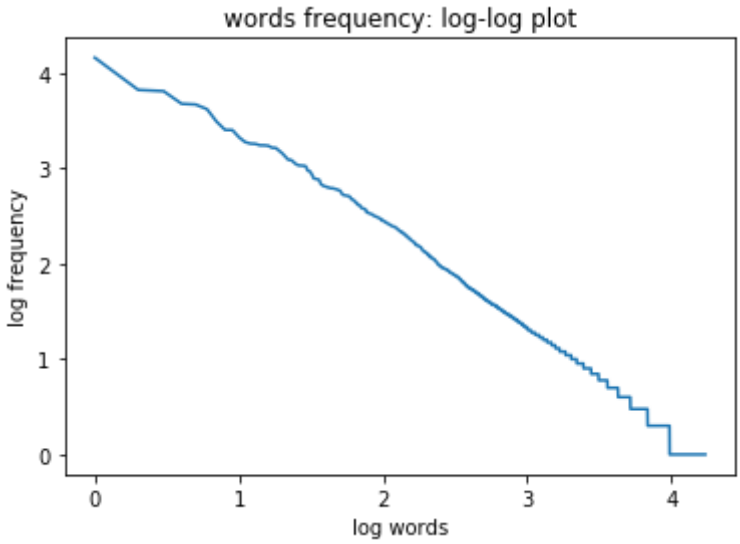
입力 [2]:

```
import re
import matplotlib.pyplot as plt

top_words = sorted(word_freq.items(), key=lambda x: x[1], reverse=True)

word_freq_arr = [(word, freq) for word, freq in top_words]

words, frequencies = zip(*word_freq_arr)
plt.bar(range(len(frequencies)), frequencies,width=100)
plt.xticks(rotation=90)
plt.xlabel('Word')
plt.ylabel('Frequency')
plt.title('Word Frequency')
plt.show()

plt.scatter(range(len(frequencies)), frequencies,s=5)
plt.xlabel('log - Word')
plt.ylabel('log - Frequency')
plt.title('words frequency log-log plot')
plt.xscale('log')
plt.yscale('log')
plt.show()
```




words frequency log-log plot
log - Frequency
log - Word
10^0
10^1
10^2
10^3
10^4


frequency
words
0
2000
4000
6000
8000
10000
0
2500
5000
7500
10000
12500
15000
17500


words frequency: log-log plot
log frequency
log words
0
1
2
3
4

Discussion

- Read this wikipedia article : https://ko.wikipedia.org/wiki/%EC%A7%80%ED%94%84%EC%9D%98_%EB%B2%95%EC%B9%99 (https://ko.wikipedia.org/wiki/%EC%A7%80%ED%94%84%EC%9D%98_%EB%B2%95%EC%B9%99)
- Discuss what you learned from the distribution.

많이 사용되는 단어가 많은 의미를 내포하고 있기에 지프의 법칙이 적용되는게 아닌가라는 생각을 하게되었습니다.

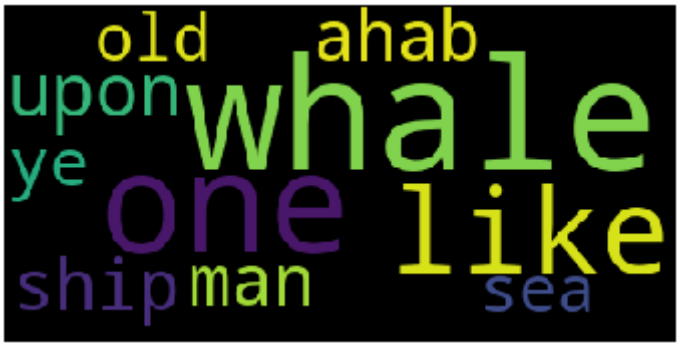
1.3 Word Cloud (5 pts)

- Print top 10 most words except stop words
- Draw word cloud of top 10 most common words
- Googling for how to draw word clouds

Your output should be like:

```
[('whale', 1228), ('one', 934), ('like', 648), ('upon', 566), ('man', 527), ('ship', 518), ('ahab', 511), ('ye', 472), ('sea', 455), ('old', 450)]
```

Your output should be like this (but NOT exactly the the same):



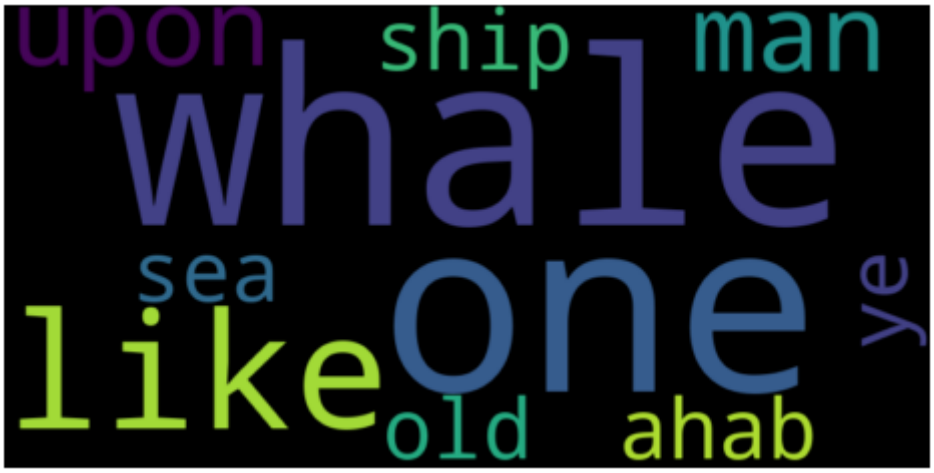
- The following is English stop words list

입力 [3]:

```
stopwords = {'it', 'than', 'out', 'an', 'at', 'until', 'wouldn', 'too', 'each', 'off', 'whom', '
```

입力 [4]:

```
# YOUR CODE MUST BE HERE
import re
import matplotlib.pyplot as plt
from wordcloud import WordCloud

words = [word for word in words if word not in stopwords]
word_freq = {}

for word in words:
    if word in word_freq:
        word_freq[word] += 1
    else:
        word_freq[word] = 1
top_words = sorted(word_freq.items(), key=lambda x: x[1], reverse=True)[:10]

word_freq_arr = [(word, freq) for word, freq in top_words]

wordcloud = WordCloud(width=800, height=400, background_color='black').generate_from_frequencies

plt.figure(figsize=(7, 3))
plt.imshow(wordcloud, interpolation='bilinear')
plt.axis('off')
plt.show()
```



Problem 2 (20 pts)

- We want to find how many CS faculty members at CS department of Stanford Univ work on CS research areas.
- First, visit https://cs.stanford.edu/research (https://cs.stanford.edu/research)
- Take a look at the source html of the web page.
- We want to scrape data on all the faculty members
- Run the following two cells and see what happens
- If necessary, install html5lib

입力 [17]:

```
from bs4 import BeautifulSoup
import requests

url = "https://cs.stanford.edu/research?items_per_page=All&field_faculty_status_value=active"
soup = BeautifulSoup(requests.get(url).text, 'html5lib')

#f = open("stanford_cs.txt", "r",encoding='UTF8')
#text = f.read()
#soup = BeautifulSoup(text, 'html5lib')
```

Remark

- Stanford Univ에 너무 많이 접속해서 (DDOS처럼 여겨져서) 접속이 막힐 수도 있음
- 해당 웹페이지를 처음 접속해서 파일로 저장한 다음,
- 파일로 부터 읽어서 숙제를 테스트하는 게 필요함.

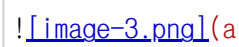
숙제 제출시 아래 cell은 절대 실행하지(출력에 포함하지) 말 것!!!

입力 [ ]:

```
print(soup.tbody.prettify())
```

Draw bar charts on research area contributions of Stanford CS faculty

- For each research area, we want to compute how many professors works on that area.
- If one professor works on n research fields, the contribution to one research field is 1/n.
- The colors for professor ranks (assistant, associate, full professors) may be your own choice.
- Your output should be like:



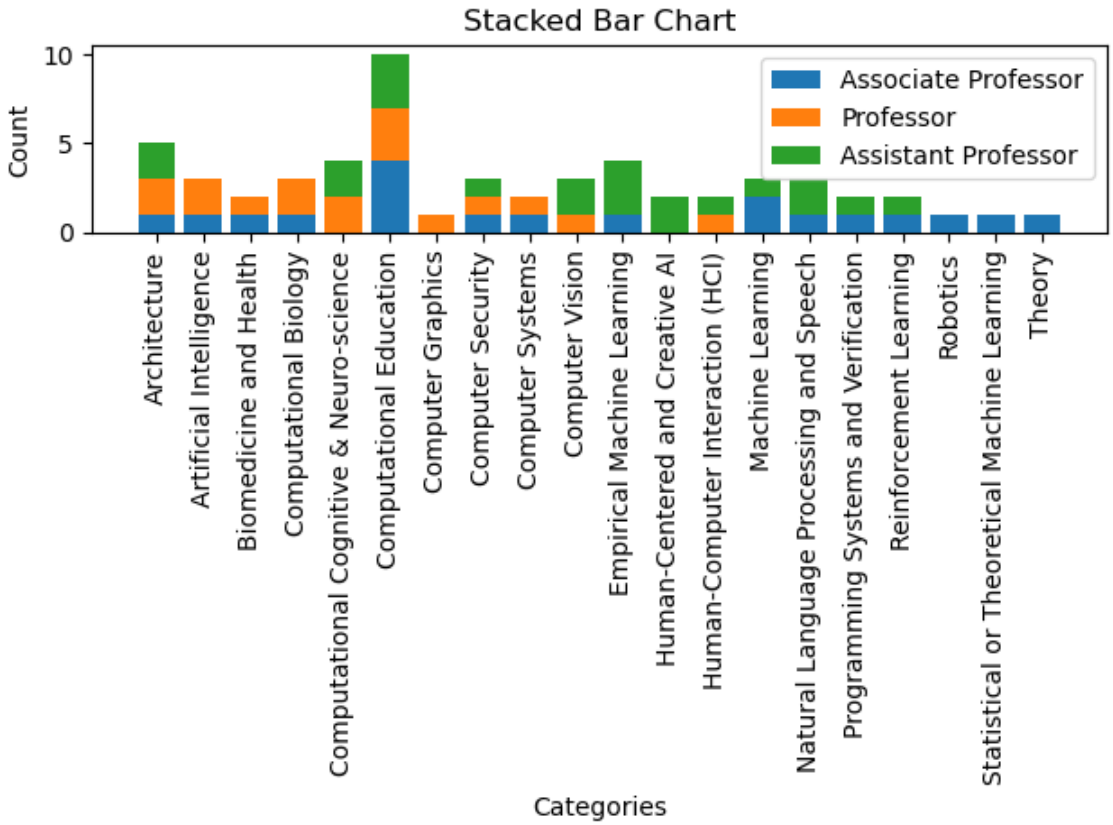
Ethics:

If you cheat, you will get negatgive of the total points.
If the homework total is 22 and you cheat, you get -22.

입力 [18]:

```
import requests
import numpy as np
import re, sys
import matplotlib.pyplot as plt


stopwords = {'td','class','views','field','research','focus','br',' ','₩n',' <br/>',' <br/><br/>

words = [word for word in words if word not in stopwords]
result_prof=[]
subject_list=[]
c_assprof=[0]*23
c_prof=[0]*23
c_asoprof=[0]*23
length=[0]*23
for a in range(2):
    for b in range(23):
        prof=soup('td',{'class':'views-field views-field-field-faculty-title fac_prof_view_title
        text = soup('td',{'class':'views-field views-field-field-research-focus fac_prof_view_fo
        if(prof.strip()=='Assistant Professor' and a==0):
            result_prof.append(prof.strip())
            subjects=[text for text in text if word not in stopwords]
            subject=[]
            for i in range(len(subjects)):
                if i%3==0:
                    subject.append(subjects[i].strip())
            for i in range(len(subject)):
                if subject[i] not in subject_list:
                    subject_list.append(subject[i])
            length[b]=len(subject)

        if(prof.strip()=='Assistant Professor'and a==1):
            index=[0]*length[b]
            subjects=[text for text in text if word not in stopwords]
            subject=[]
            for i in range(len(subjects)):
                if i%3==0:
                    subject.append(subjects[i].strip())
            for i in range(len(index)):
                index[i]=subject_list.index(subject[i])
            for i in range(len(index)):
                c_assprof[index[i]]+=1


        elif(prof.strip()=='Professor' and a==0):
            result_prof.append(prof.strip())
            subjects=[text for text in text if word not in stopwords]
            subject=[]
            for i in range(len(subjects)):
                if i%3==0:
                    subject.append(subjects[i].strip())
            for i in range(len(subject)):
                if subject[i] not in subject_list:
                    subject_list.append(subject[i])
            length[b]=len(subject)

        if(prof.strip()=='Professor'and a==1):
            index=[0]*length[b]
            subjects=[text for text in text if word not in stopwords]
```

```
            subject=[]
            for i in range(len(subjects)):
                if i%3==0:
                    subject.append(subjects[i].strip())
            for i in range(len(index)):
                index[i]=subject_list.index(subject[i])
                c_prof[index[i]]+=1

        elif(prof.strip()=='Associate Professor'and a==0):
            result_prof.append(prof.strip())
            subjects=[text for text in text if word not in stopwords]
            subject=[]
            for i in range(len(subjects)):
                if i%3==0:
                    subject.append(subjects[i].strip())
            for i in range(len(subject)):
                if subject[i] not in subject_list:
                    subject_list.append(subject[i])
            length[b]=len(subject)

        if(prof.strip()=='Associate Professor'and a==1):
            index=[0]*length[b]
            subjects=[text for text in text if word not in stopwords]
            subject=[]
            for i in range(len(subjects)):
                if i%3==0:
                    subject.append(subjects[i].strip())
            for i in range(len(index)):
                index[i]=subject_list.index(subject[i])
            for i in range(len(index)):
                c_asoprof[index[i]]+=1
        elif(prof.strip()=='Courtesy Professor'and a==0):
            result_prof.append(prof.strip())
subject_list=sorted(subject_list)
assistant_professor=np.array(c_assprof)[:20]
professor=np.array(c_prof)[:20]
associate_professor=np.array(c_asoprof)[:20]

categories=subject_list
plt.bar(categories, associate_professor, label='Associate Professor')
plt.bar(categories, professor, bottom=associate_professor, label='Professor')
plt.bar(categories, assistant_professor, bottom=associate_professor+professor, label='Assistant

plt.xlabel('Categories')
plt.ylabel('Count')
plt.title('Stacked Bar Chart')
plt.legend()

plt.xticks(rotation=90)
plt.tight_layout()
plt.show()
```



What to submit

- Run **all cells** after restarting the kernel
- Goto "File -> Print Preview"
- Print the page as pdf
- Pdf file name must be in a form of: homework_4_홍길동_202300001.pdf
- Submit the pdf file in google classroom
- No late homeworks will be accepted
- Your homework will be graded on the basis of correctness, performance, and programming skills
- Your homework will be graded on the basis of correctness and programming skills

入力 [ ]: