Programming 1 Report

과목명	Programming 1
학과	산업공학과
학번	201611145
이름	박민성
제출 일자	2019.5.27
지도 교수님	윤장혁 교수님

1. 프로젝트 진행 과정

2019년 4월 30일 경부터 개인과제를 시작하였습니다. 대부분을 구글링을 통해, 모듈 사용법을 익혔고, 마음에 들지 않아서 2번 정도 코드를 새로 짰습니다. 그 이유는 처음에는 xml에 대한 이해가 부족해서, infoarray에 있는 agent name과 applicant name에 들어 있는 첫번째 자료밖에 찾지 못하는 코드를 짰었기 때문입니다. 현재는 자료의 저장을 이차원 리스트에 저장을 해서, 배열에 자료가 두개 이상 있어도, 정상적으로 다 자료가 저장이 될 수 있게 코드를 짰습니다.

```
for i in range(len(f)-1):
    soup = bs(f[i],'xml')
    agentInfoArray_tag = soup.find('applicantInfoArray')
    applicant_name_split = agentInfoArray_tag.find('name')
    all_applicant_name.append(applicant_name_split.get_text())
    titlesplit = soup.find('inventionTitle')
    alltitle.append(titlesplit.get_text())
    agentInfoArray_tag = soup.find('agentInfoArray')
    inventor_name_split = agentInfoArray_tag.find('name')
    if inventor_name_split !=None:
        all_inventor_name.append(inventor_name_split.get_text())
```

이 것은 제가 초기에 짠 코드입니다. 제일 처음 스플릿 되는 f[0]의 자료만 보고 무조건 xml 한덩이에 하나의 info에는 하나의 name 들어 있을 것이라고 생각했습니다.

이러한 부분은 수업과 실습수업을 통해서 점점 배워가면서 보완해 나갔습니다. 수업시간에 xml에 대한 구조 이해를 더 심화시킬 수 있었고, 예외 처리 기법 등을 현재 코드에 적용했습니다.

2. 중점 사항

어느 컴퓨터에서도 돌아가면 좋겠다 라는 생각으로 코드를 짰습니다.

windows에서 코드를 실행하던, apple 사의 컴퓨터에서 코드를 실행하던 잘 돌아갔으면 좋겠다 라는 마음으로 코드를 짰습니다.

파일 저장 관련해서, 처음에는 그냥 c드라이브에 파일을 생성해서 저장했습니다.

그러다가 문득 다른 컴퓨터에서 돌렸을 때는 저장이 안된다는 것을 깨닫고, 어느 컴퓨터에서라도 폴더가 생성 될 수 있게 했습니다.

사용자의 마음에서 계속 지금 검색할 때 불편한게 무엇일까? 라는 의문을 가지고 프로젝트를 진행 했습니다.

3. 코드 설명 및 출력 화면 설명

메인 기능 설명

```
————————————————————————
원하시는 검색기능을 선택해주세요
1) 특정 출원인->출원한 특허목록출력
2) 특정 키워드포함 특허목록출력
3) 특정 발명자가 출원한 특허목록 출력
4) 통계자료 보기
5) 특정 단어가 들어있는 정보를 모두 출력(아주 자세한 정보를 원하시는분)
6) 특허 검색기능 종료
1,2,3,4,5,6 숫자만 입력해주세요.
————————————————————————
```

1. 특정 출원인을 검색했을 때 해당 출원인이 출원한 특허명이 모두 나옵니다.

```
찾고 싶은 검색어를 입력해주세요
종근
찾으시는 단어 ' 종근 ' 가 없습니다.


['장종근']
다음은 ' 종근 ' 를 포함 하는 단어입니다.
혹시 찾으시는 검색어가 위의 리스트 중에 있나요?

1) 위의 리스트 중에 있다 -> 계속 검색
2) 위의 리스트중에 없다 -> 나가기
```

출원인의 성과 이름 모두 기억나지 않는 경우에도, 리스트에서 입력단어를 포함하는 출원인을 찾아서 출력해줍니다. 그리고 원한다면 리스트에 있는 단어로 계속 검색을 하게 끔 만들었습니다.

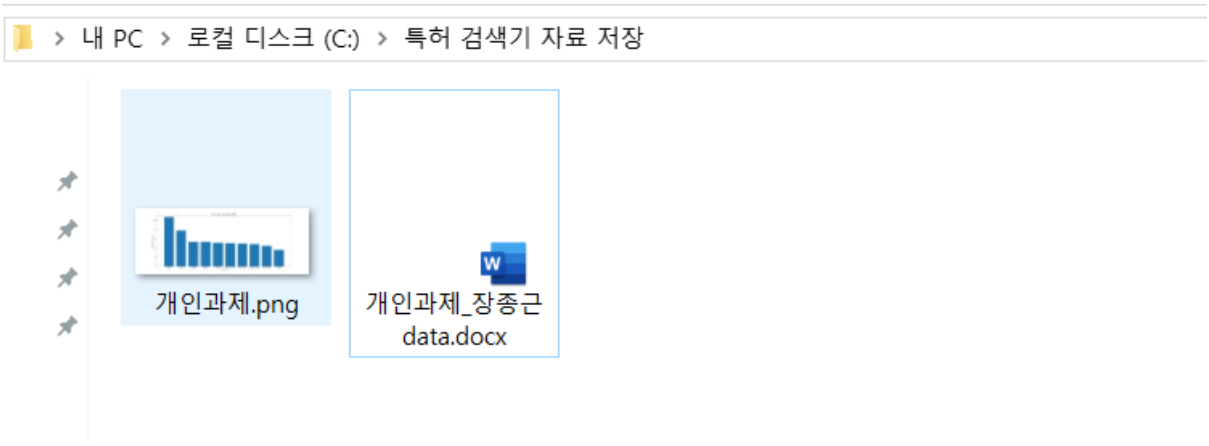
```
케이블 트레이용 방진형 행거장치 및 이를 포함한 케이블 트레이


검색한 내용을 저장하시겠습니까?
번호를 입력해주세요.
1) 저장하겠습니다.
2) 저장하고 싶지 않다.
1
출력하고 싶은 파일의 형태를 정해주세요
1) ##  word 파일로 저장  ##
 장점 : 깔끔해보인다.
2) ##  csv 파일로 저장   ## 로 저장
장점 : 다른 프로그램에서 응용하기 쉽고, dict 타입의 자료 저장하기 용이합니다.
1
저장하고 싶은 파일명을 적어주세요.
추천예시)
ex) 발명자_장수길_invetionTitle_자료
개인과제_장종근_data
C->특허 검색기 자료 저장 폴더에 저장되었습니다.
```

'종근' 검색 후, 추천 받은 단어 '장종근'으로 검색을 다시 했습니다.

'케이블 트레이용 방진형 행거장치 및 이를 포함한 케이블 트레이' 라는 특허명을 출력받았습니다.

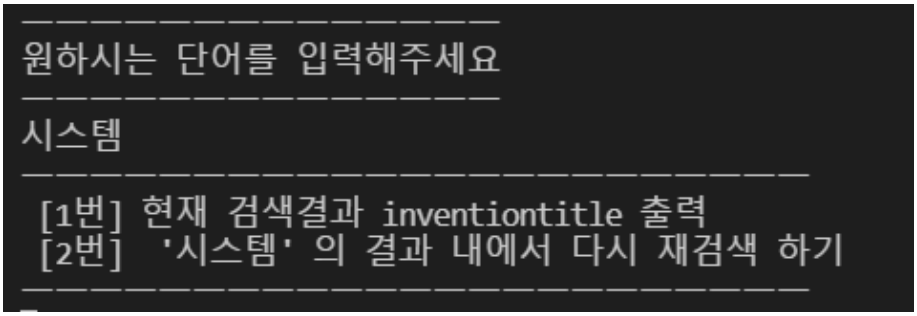
그 후, 저장을 csv or word 로 할 수 있습니다. 원하는 파일명을 설정 할 수 있습니다.



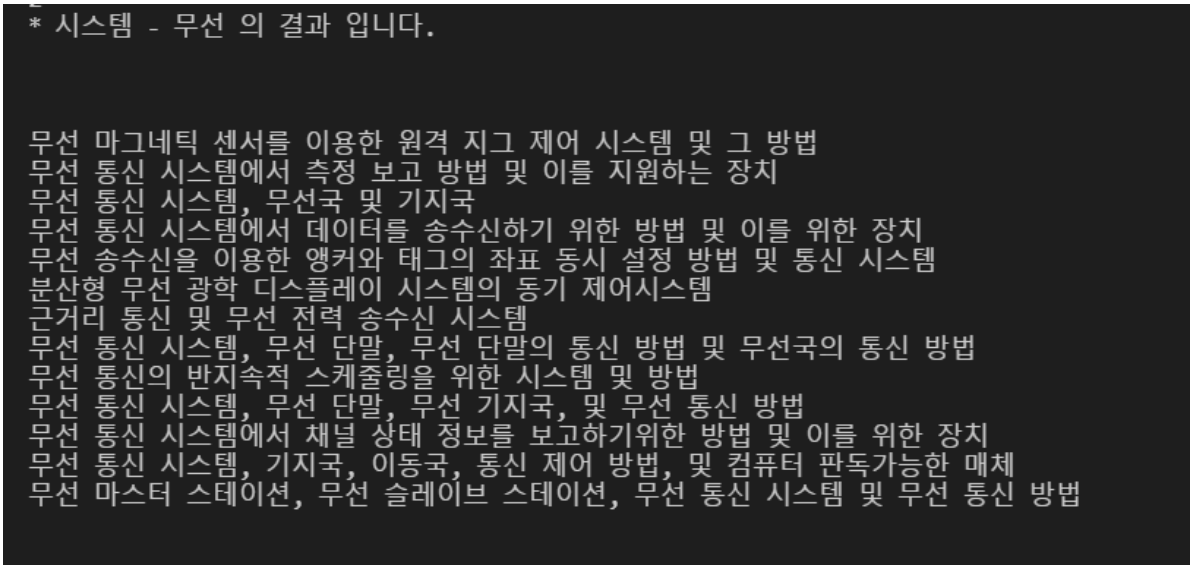
2. 특허명 중에 원하는 단어가 들어 가는 특허명을 모두 표시해줍니다.

   기본적으로 입력 받은 '단어'가 포함 되는 특허명을 모두 출력해줍니다.

   또한, 출력된 정보 중에서 다시 검색이 가능합니다.



Ex) 시스템 검색 후, 무선, 검색

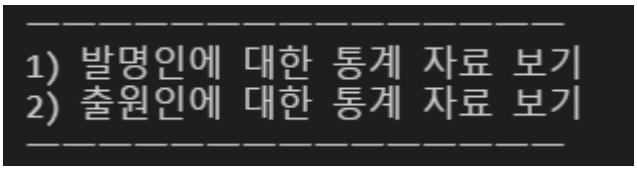


시스템 검색 후 리스트에서 무선이 포함되는 특허명만 출력이 가능합니다.

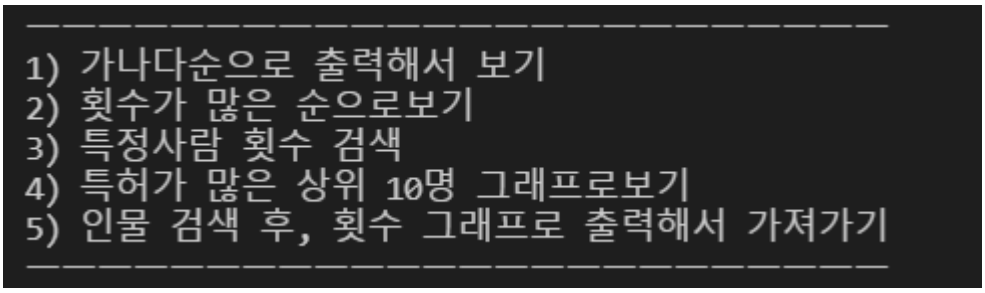
3. 특정 발명자를 검색했을 때 해당하는 특허명이 모두 나옵니다.

   1번과 Logic이 동일합니다. 다만, 출원인, 발명자의 차이입니다. 함수로 코드를 짰기 때문에 list만 바꿔주면, 동일하게 돌아갑니다.

4. 출원인, 발명자에 대한 통계를 보여줍니다

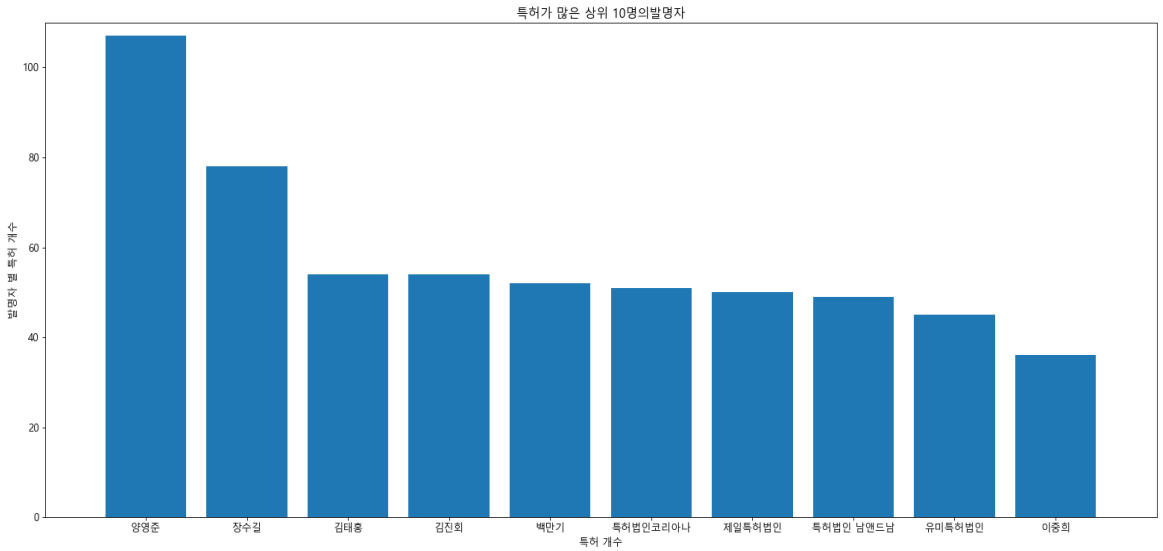


1,2번의 메뉴 구성은 같습니다.



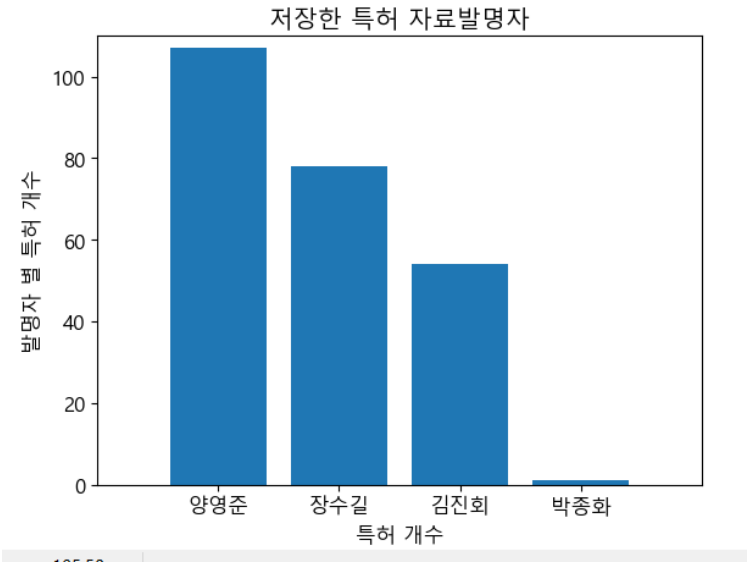
1. 가나다 순으로 dictionary의 key값을 정렬해줍니다.
2. Value 값을 정렬해줍니다.
3. key값을 검색해서, key값과 value 값을 보여줍니다.

4. 상위 10명의 자료를 그래프로 표현해줍니다.



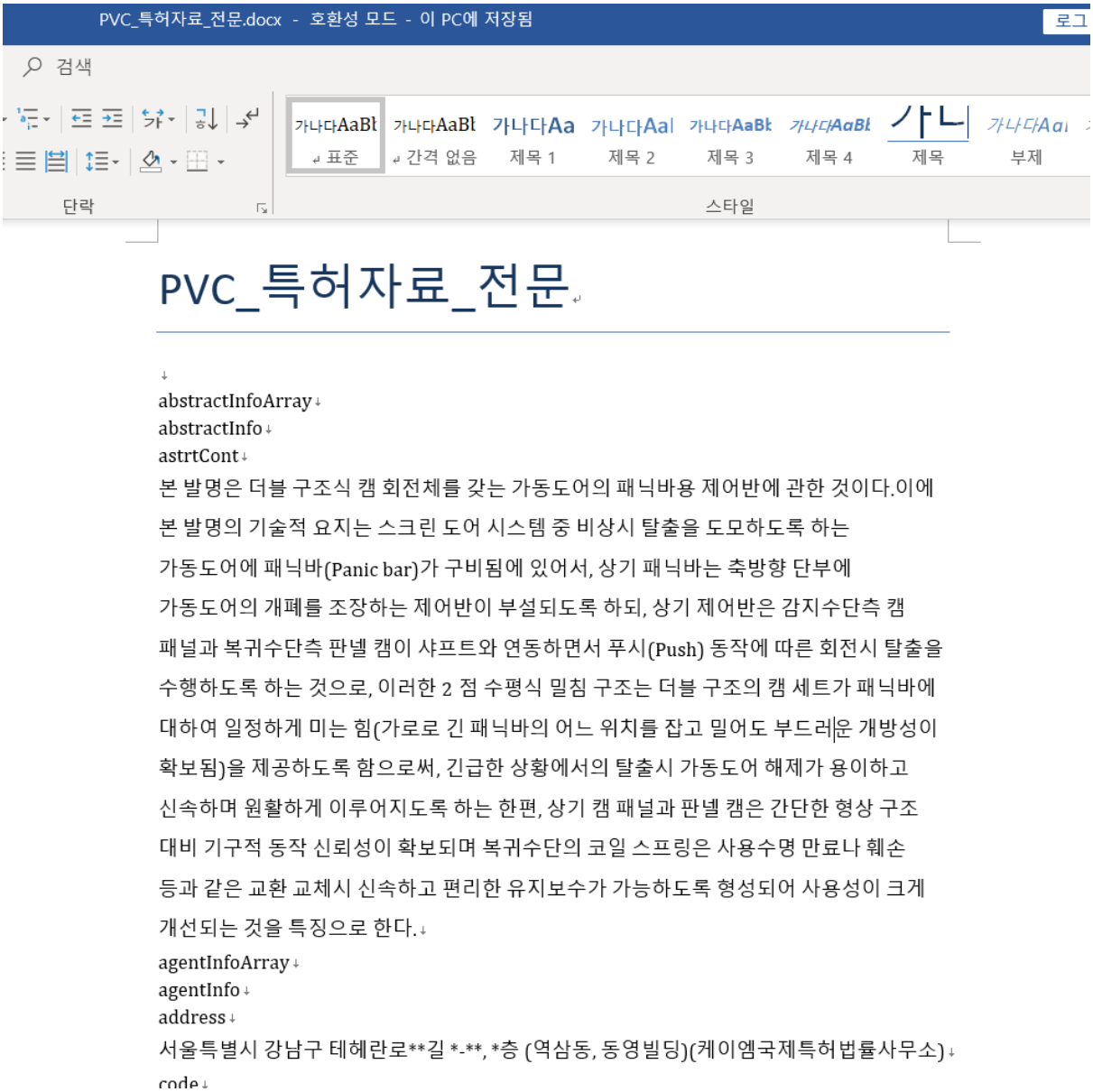
5. 원하는 발명자/출원인을 모두 추가해서 한눈에 볼 수 있는 그래프로 표현해줍니다.

```
원하시는 '발명자'의 이름을 입력해주세요
김진회
{'양영준': 107, '장수길': 78, '박종화': 1, '김진회': 54}
1) 원하는 '발명자' 계속 검색 + 저장
2) 그만 하고 그래프로 보기
```



5. 특정 단어가 포함되어 있는 특허명을 리스트에 보여줍니다. 리스트에서 선택하면, 모든 정보를 보여줍니다.

   모든 자료를 보여줍니다. InfoArray의 이름까지 하나도 지우지 않았기 때문에, 모든 정보를 지저분한 tag는 모두 떼어진 채 볼 수 있습니다.



코드 설명

```
1   import tkinter
2   import re
3   import platform
4   import pandas as pd
5   import operator
6   import os
7   from bs4 import BeautifulSoup as bs
8   import numpy as np
9   import matplotlib.pyplot as plt
10  from matplotlib import font_manager, rc
11  from matplotlib.pyplot import figure
```

1번째줄 import tkinter

화면에 띄우는 GUI를 사용할 때 쓰는 모듈입니다.

2번째줄 import re

re.sub 모듈을 사용했습니다.

3번째줄 import platform

사용자의 운영체제를 파악해주는 모듈입니다.

4번째줄 import pandas as pd

csv 파일을 저장할 때 사용하는 모듈입니다.

5번째줄 import operator

딕셔너리를 정렬할 때 사용했던 모듈입니다.

6번째줄 import os

파일 저장 경로에 관련해서 쓰는 모듈입니다.

7번째줄 from bs4 import BeautifulSoup as bs

Xml 파싱에 사용 되는 모듈입니다.

8번째줄 import numpy as np

그래프를 그릴 때 사용하는 연산 모듈입니다.

9번째줄 import matplotlib.pyplot as plt

그래프를 그릴 때 사용하는 모듈입니다.

10번째줄 from matplotlib import font_manager, rc

그래프의 폰트를 정해주는 모듈입니다.

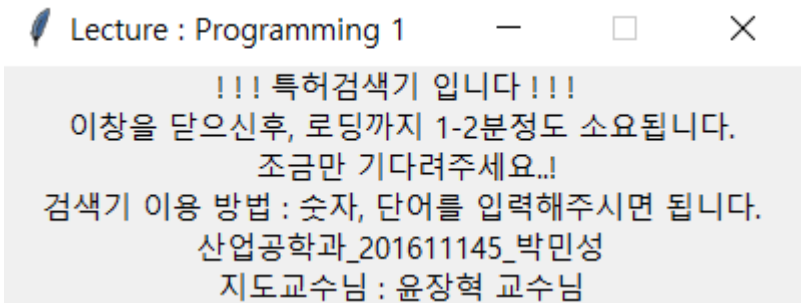
11번째줄 from matplotlib.pyplot import figure

그래프의 크기를 정해주는 모듈입니다.

```
12  figure(num=None, figsize=(27, 10), dpi=80, facecolor='w', edgecolor='k')
13  plt.rcParams.update({'font.size': 16})
14  def notice(word):
15      window=tkinter.Tk()
16      window.title("Lecture : Programming 1 ")
17      window.geometry("400x120+100+100")
18      window.resizable(False, False)
19      label=tkinter.Label(window, text= word )
20      label.pack()
21      window.mainloop()
22  notice("! ! ! 특허검색기 입니다 ! ! ! \n이창을 닫으신후, 로딩까지 1-2분정도 소요됩니다.\n 조금만
23  print('특허검색기능 로딩중입니다, 로딩까지 1-2분정도 소요됩니다.')
24
25  dirname = 'C://특허 검색기 자료 저장'
26  if not os.path.isdir('C://특허 검색기 자료 저장'):
27      os.mkdir('C://특허 검색기 자료 저장')
```

12 ~ 13번째줄: 그래프의 크기를 결정해주고, 폰트의 크기를 결정해줍니다.

14 ~ 21번째줄: GUI 창을 띄워주는 함수를 만들었습니다.



title 은 이름입니다

geometry는 크기를 결정하는 것이고

마지막 window.mainloop()를 통해 창에 계속 띄워주고, 종료하기 전까지 계속 창에 떠있습니다.

notice("하고싶은 말")을 넣으면 창을 띄워줍니다. 처음과 끝에 배치했습니다.

이 모듈 사용법은 구글링을 통해 찾았습니다.

25 ~ 27번째줄: os 모듈을 사용해서, 어떠한 컴퓨터로 가서든 '특허 검색기 자료 저장'이라는 폴더를 생성해줍니다. os.mkdir 이 파일을 생성하는 것인데, 같은 이름의 폴더가 있으면 오류가 나서 if 문으로 만들었습니다.

```
29  if platform.system() == 'Windows':
30      font_name= font_manager.FontProperties(fname="C:/Windows/Fonts/malgun.ttf").get_name()
31      rc('font', family=font_name)
32  else:
33      rc('font', family='AppleGothic')
34
35  f = open("개인과제data.txt",'r', encoding='UTF-8')
36  f = f.read()
37  f = f.replace("xml version=1.0 encoding=utf-8","xml version=\'1.0\' encoding=\'utf-8\'")
38  f = f.split('\n')
```

29 ~ 33번째줄: 제가 사용한 그래프 모듈에서 한글 깨짐현상이 발견되어서 글꼴을 찾아서 입혀줬습니다. Windows와 Apple 운영체제의 글꼴 저장경로가 다르기 때문에, platform 을 판별해서 widows 면 맑은체를 apple이라면 고딕체를 입혀서 한글깨짐현상을 없앴습니다.

35 ~ 38번째줄: 데이터를 오픈해줍니다. 따옴표가 빠져있기 때문에 replace를 통해 xml 구조를 바로 잡아줍니다. 그리고 xml이 여러 개가 있는 txt 파일 이기 때문에, '₩n'으로 스플릿하는 방식으로 코드를 짰습니다.

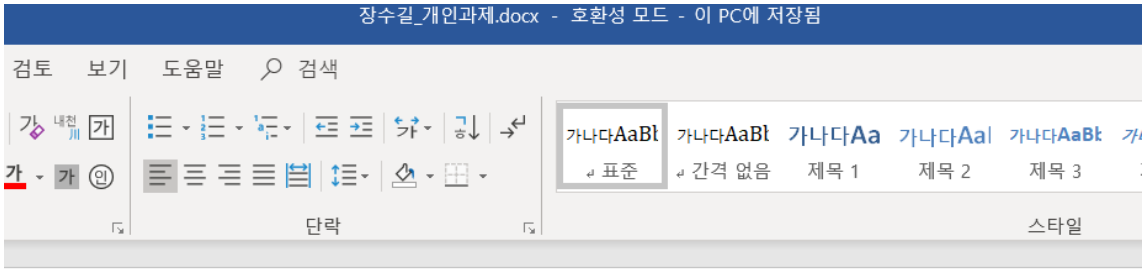
```
40  def makefile(fn):
41      save_or_not=input("검색한 내용을 저장하시겠습니까?\n번호를 입력해주세요.\n1) 저장하겠습니다.\n2)
42      if int(save_or_not)==1:
43          how = input("출력하고 싶은 파일의 형태를 정해주세요\n1) ##  word 파일로 저장  ##\n 장점 : 깔
44  ## 로 저장 \n장점 : 다른 프로그램에서 응용하기 쉽고, dict 타입의 자료 저장하기 용이합니다.\n")
45          want_name= input("저장하고 싶은 파일명을 적어주세요.\n추천예시)\nex) 발명자_장수길_invetionT
46          if int(how)==2:
47              dataframe = pd.DataFrame(fn)
48              a = "C:\\특허 검색기 자료 저장\\"+ want_name+".csv"
49              dataframe.to_csv(a, encoding='cp949',header=False, index=False)
50              print("c드라이브 -> 특허 검색기 자료 저장 폴더에 저장되었습니다.\n")
51          elif int(how)==1:
52              from docx import Document
53              document = Document()
54              document.add_heading(want_name,0)
55              p = document.add_paragraph()
56              p.add_run("\n")
57              for i in fn:
58                  p.add_run(str(i)+'\n')
59              document.save('C:\\특허 검색기 자료 저장\\'+want_name+'.docx')
60              print("C->특허 검색기 자료 저장 폴더에 저장되었습니다.\n")
61      elif int(save_or_not)==2:
62          print("저장하지 않았습니다. 초기화면으로 돌아갑니다.\n")
```

40 ~ 62번째줄: 파일을 저장하는 함수를 만들었습니다.

Csv 저장 방식과 word 파일 저장 두가지를 사용하였습니다.

사용자가 저장하고 싶은 파일명으로 저장을 할 수 있게 구현했습니다.

47 ~ 49번째줄: pd 모듈을 사용해서 csv 파일로 저장하였고, cp949 인코딩을 사용해 깨짐을 방지했습니다.



장수길_개인과제

부호화 장치, 부호화 방법, 복호 장치, 복호 방법, 및 컴퓨터 판독가능한 기억 매체

내부 이형제, 당해 내부 이형제를 포함하는 조성물 및 당해 조성물을 사용한 플라스틱 렌즈의 제조 방법

판 형상체의 적재 장치 및 판 형상체의 적재 방법

화상 처리 장치 및 방법

무선 통신의 제어 채널

52 ~ 59번째줄 : document 모듈을 사용했습니다. Paragraph가 단락 추가입니다. 그리고 제목 부분에 파일명이 표시 되도록했습니다. 자료를 한줄한줄 받아서 p.add_run 해주면 word 파일에 한줄씩 표시가 됩니다. 사용법은 구글링을 한 후, 블로그에서 한줄 한줄 따라 적어봤습니다.

```
64  def list_print_save(list_type):
65      number = int(input("————————————————————\n1)리스트 type 으로 화면에서 보기\n2)한줄씩 프
66      if number == 1:
67          try:
68              if len(word_word_save_list) != 0:
69                  print("*",wantword,"-",wantword2,"의 결과 입니다.\n")
70          except:
71              pass
72          finally:
73              print(list_type)
74
75      if number == 2:
76          try:
77              if len(word_word_save_list) != 0:
78                  print("*",wantword,"-",wantword2,"의 결과 입니다.\n")
79          except:
80              pass
81          finally:
82              print("\n")
83              for i in range(0,len(list_type)):
84                  print(list_type[i])
85              print("\n")
86      makefile(list_type)
```

64 ~ 86번째줄: 검색한 자료를 list로 출력할 것인지 한 줄씩 print 된 형식으로 볼 것인지 하나의 함수로 만들었습니다. 코드가 길어 진 이유는 바로 try 문 때문인데요, 원래 이 함수는 출원인 검색, 발명자 검색에서 쓰였는데, 2번 검색기능인 단어 검색 기능에서도 쓰고 싶어서 추가했습니다. word_word_list가 정의되지 않는 상태에서도 함수가 돌아가게 하기 위해여 finally 문을 사용했습니다. 그리고 마지막에 makefile 함수를 사용해서, 저장 할 것인지 물어봅니다. List type과 dict type 상관없이 돌아갑니다. 처음부터, 함수를 많이 활용하는 식으로 코드를 짰으면 코드 전체의 구성이 간결하고 더 짧아졌겠지만, 코드의 오류를 수정하고, 덧붙이는 식으로 해서 코드가 지저분해 졌습니다. 수업과 실습시간에 예외처리를 배우지 않았다면, 오류가 자꾸 생겨서 함수로 만드는 것을 포기 할 뻔했습니다.

```
88  def fix(all_list):
89      want=input("찾고 싶은 검색어를 입력해주세요\n")
90      find_list=[]
91      for i in range(0,len(all_list)):
92          for j in range(0,len(all_list[i])):
93              if want == all_list[i][j]:
94                  soup = bs(f[i],'xml')
95                  find_list.append(soup.find('inventionTitle').get_text())
96
97      if len(find_list) == 0:
98          print("찾으시는 단어 ' "+want+" ' 가 없습니다.")
99          new=[]
100         for i in all_list:
101             for j in i:
102                 if want in j:
103                     new.append(j)
104         new=list(set(new))
105
106         if len(new)==0:
107             hoxy = 2
```

```
109         elif len(new)!=0:
110             print('\n')
111             print(new)
112             hoxy = input("다음은 ' "+want+" ' 를 포함 하는 단어입니다.\n혹시 찾으시는 검색어가 위
113 
114         if int(hoxy) == 1:
115             want_agent2= input("위 리스트 중 한 검색어를 다시한번 정확히 입력해주세요\n")
116             save_list_2=[]
117 
118             for i in range(0,len(all_list)):
119                 for j in range(0,len(all_list[i])):
120                     if want_agent2 == all_list[i][j]:
121                         soup = bs(f[i],'xml')
122                         save_list_2.append(soup.find('inventionTitle').get_text())
123             save_list_2=list(set(save_list_2))
124             list_print_save(save_list_2)
125 
126         elif int(hoxy)==2:
127             print("————————————————\n초기화면으로 돌아갑니다.\n")
128 
129     elif len(find_list) != 0:
130         find_list=list(set(find_list))
131         list_print_save(find_list)
```

88 ~ 131번째줄: 특정 발명인, 출원인에 대한 특허명 검색기능 함수입니다.

저는 이 기능에 보정 기능을 넣었습니다. 예시로 "장수길"이라는 사람이 있다고 하면, '수길' 또는 '장수' 만 입력해도 [장수길] 이라는 사람을 추천해줍니다. '장'을 입력했을 때 new라는 함수에 아래 보이는 장훈, 장덕순 등등의 자료가 저장됩니다.

```
찾으시는 단어 ' 장 ' 가 없습니다.


['장훈', '장덕순', '장원철', '장한특허법인', '장
', '특허법인와이에스장', '장태화', '장순부', '장
다음은 ' 장 ' 를 포함 하는 단어입니다.
혹시 찾으시는 검색어가 위의 리스트 중에 있나요?

1) 위의 리스트 중에 있다 -> 계속 검색
2) 위의 리스트중에 없다 -> 나가기
```

저는 이 기능이 한글 뿐만 아니라 영어권에서도 쓸모가 많다고 생각합니다. 이름만 아는 경우나, 이름이 가물가물한 경우, 제가 기존에 짰던 코드에서 사용할 수가 없어서 이러한 기능을 추가해 보았습니다. 91-95번째 줄에서 이중 포문을 돌리는 이유는 한 xml 덩어리당 agent name이나 applicant name이 두개 이상 있을 수도 있기 때문입니다.

100 ~ 103번째줄에서도 위의 방식과 마찬가지 입니다.

hoxy라는 스위치 변수를 통해서 검색을 더 이상 원치 않거나, 찾는 인물이 없을 때 초기화면으로 돌려줍니다. Set->list 를 사용해서 리스트에 중복으로 표현하는 것을 막음으로써 보기에 더 좋게 해줍니다.

```python
def toggae(choose_num,main_list):
    if main_list == agent_toggae_list:
        name = "발명자"
    elif main_list == applicant_toggae_list:
        name = "출원인"
    if choose_num ==1:
        result = dict()
        title = "특허가 많은 상위 10명의"
        for i in main_list:
            result[i] = main_list.count(i)
        people_num = 10
        final_result= result
    elif choose_num ==2:
        savesave=dict()
        title = "저장한 특허 자료"
        while True:

            numnumnum = input("1) 원하는 '"+name+ "' 계속 검색 + 저장\n2) 그만 하고 그래프로 보기\n")
            if int(numnumnum) == 1:
                namename = input("원하시는 '"+name+"'의 이름을 입력해주세요\n")
                result2=dict()
                for i in main_list:
                    result2[i] = main_list.count(i)
                for key,value in result2.items():
                    if key == namename:
                        savesave[key] = value
                print(savesave)
            elif int(numnumnum) == 2:
                break
```

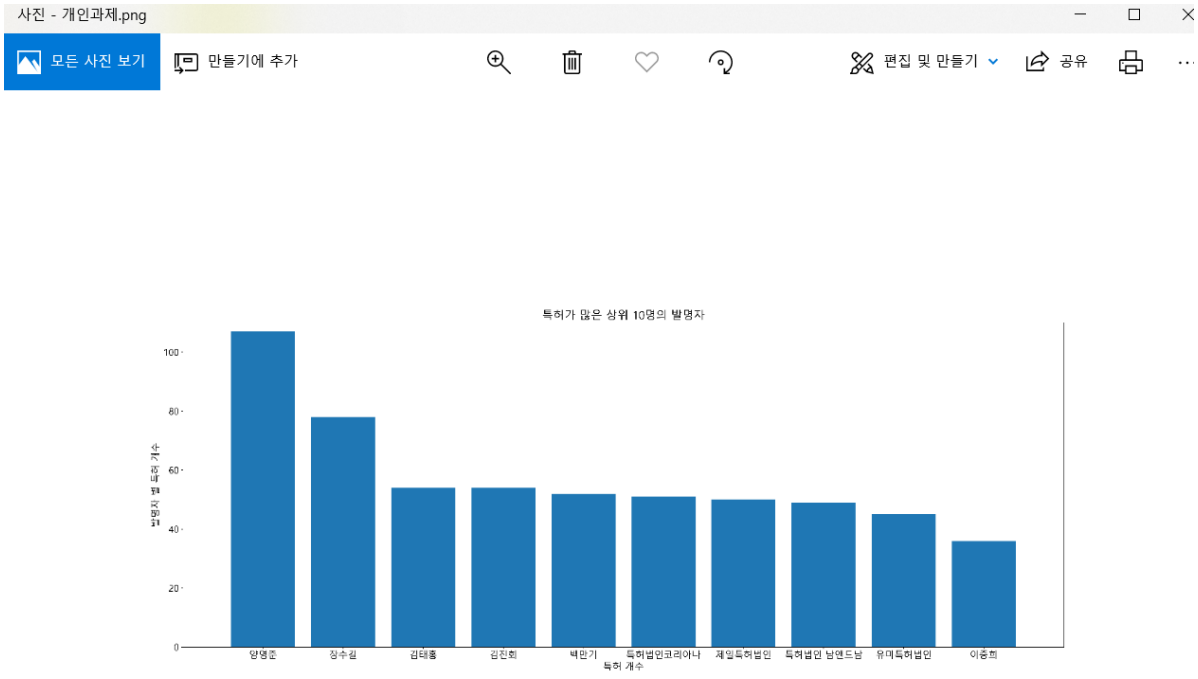
```
163         people_num = len(savesave)
164         final_result = savesave
165 
166     y1_value = []
167     x_name =[]
168     numnum = sorted(final_result.items(), key = operator.itemgetter(1,0), reverse = True)
169     for key,value in numnum:
170         y1_value.append(value)
171         x_name.append(key)
172     a=[]
173     b=[]
174     for i in range(people_num):
175         a.append(y1_value[i])
176         b.append(x_name[i])
177     x_name = b
178     y1_value = a
179     n_groups = len(x_name)
180     index = np.arange(n_groups)
181     plt.bar(index, y1_value, tick_label=x_name, align='center')
182     plt.xlabel('특허 개수')
183     plt.ylabel( name+' 별 특허 개수')
184     plt.title(title+name)
185     plt.xlim( -1, n_groups)
186     if main_list == agent_toggae_list:
187         plt.ylim( 0, 110)
188     elif main_list == applicant_toggae_list:
189         plt.ylim( 0, 25)
190     print(plt.show())
191     savenum= input("위 그래프를 저장하시겠습니까?\n1) 저장하겠다.\n2) 저장하지 않겠습니다.\n")
192 
193     if savenum==1:
194         want_name = input("저장하고 싶은 이름을 입력해주세요.\n")
195         plt.savefig("c:\\특허 검색기 자료 저장\\"+ want_name+".png",dpi=300)
196         print("c드라이브 -> 특허 검색기 자료 저장 폴더에 저장되었습니다.\n")
197     elif savenum==2:
198         print("초기화면으로 돌아갑니다.\n")
```

133 ~ 198번째줄: 통계 기능을 넣었습니다. 통계치를 출력,저장 해주는 함수입니다.

dictionary를 이용해서 실습수업 시간에 했던 것처럼 개수를 세어줍니다.

일단, main_list에 출원인 or 발명자의 통계치를 넣느냐에 따라 그래프가 다르게 출력됩니다. Choose_num 1은 특허가 많은 상위 10명의 통계치를 그래프로 표현해주고, 2는 자신이 원하는 사람을 차례차례 저장해서 통계자료로 보여줍니다. 우선 모든 결과값을 result에 dict 형태로 저장해줍니다. 그리고 sorted 함수를 통해서, 특허개수가 많은 순으로 뽑은 다음 상위 10개만 그래프로 뽑아줍니다. 2번 기능은 1번 기능과 방식은 크게 다르지 않습니다. 원하는 사람만 값을 추가해줘서 그래프로 뽑아줍니다. 내가 원하고 싶은 자료만 넣어서 , 새로운 result2라는 dict를 만들어서 1번과 동일한 방식을 사용해서 그래프로 표현해줍니다.

x축은 사람을, y축은 특허의 개수를 표현해줍니다. 구글링을 통해서 그래프 그리는 방법을 배웠습니다. plt.ylim은 y축의 범위입니다. 출원인의 최대값은 25 수준이고, 발명자의 최대값은 110수준이기 때문에 바꿔줬습니다.



그리고 마지막에 그래프를 '저장하고 싶은 파일명'.png로 저장할 수 있게 했습니다.

```
200   all_agentname=[]
201   all_applicantname=[]
202   all_title=[]
203   agent_toggae_list=[]
204   applicant_toggae_list=[]
205   all_content_save=[]
206   temporary_list=[]
207
208   alltitle=[]
209   view_list=[]
210   for i in range(len(f)-1):
211       soup = bs(f[i],'xml')
212       agentInfoArray = soup.findAll('agentInfoArray')
213       applicantInfoArray = soup.findAll('applicantInfoArray')
214       applicantfider=[]
215       agentfinder=[]
216       agent_to2list=[]
217       applicant_to2list=[]
218       all_content = soup.prettify()
219       all_content_save.append(all_content)
220       titlesplit = soup.find('inventionTitle')
221       alltitle.append(titlesplit.get_text())
222       for i in range (0,len(agentInfoArray)):
223           agentfinder=agentInfoArray[i].findAll('name')
224       for i in range(0,len(agentfinder)):
225           agent_to2list.append(agentfinder[i].get_text())
226           agent_toggae_list.append(agentfinder[i].get_text())
227       all_agentname.append(agent_to2list)
228       for i in range (0,len(applicantInfoArray)):
229           applicantfinder=applicantInfoArray[i].findAll('name')
230       for i in range(0,len(applicantfinder)):
231           applicant_to2list.append(applicantfinder[i].get_text())
232           applicant_toggae_list.append(applicantfinder[i].get_text())
233       all_applicantname.append(applicant_to2list)
234       titlesplit = soup.find('inventionTitle')
235       all_title.append(titlesplit.get_text())
```

200 ~ 235번째줄: xml자료 활용

38번째줄에 f에 담긴 자료를 활용합니다. bs4를 활용합니다. For문 하나로 돌려야 더 시간 단축이 되어서 이렇게 복잡한 for 문이 완성되었습니다. 원래 코드에는 3개정도 따로따로 돌렸었는데, 이런 식으로 하면 로딩시간이 너무 길게 되어서 바꾸었습니다.

지금 f에 담긴 자료는 리스트 형태입니다. xml 자료 하나 당 하나의 원소를 이루고 있습니다.

For 문을 돌려서 하나 하나씩 soup에 파싱해서 findAll을 이용해서 정보들을 찾아줍니다.

get_text는 태그를 떼어줍니다.

all_ agentname 에는 모든 발명자 이름이 들어갑니다.

먼저 agentInfoArray라는 모든 agent name을 findall합니다. Agentfinder에 태그를 붙인채 name을 다 찾습니다. Name이 1개일 수도있고 여러 개 있수도 있기 때문에 for문으로 get_text를 해줍니다. get_text라는 식은 list 형태에서는 되지 않기 때문에 하나하나 for문으로 돌려줍니다. 모든 정보는 바로 agent_toggae_list에 저장하고, 리스트 형태로 만든후에 all_agentname에 저장해줍니다.

agent_toggae_list는 횟수를 찾는 것에 집중 해야 되기 때문에 단순 리스트의 형태로 저장됩니다., 쉽기 생각하시면 all_agentname에 있는 자료가 이중이 아니라 일렬로 나열되어 있는 형태라고 생각하시면 됩니다. **applicant**도 agent와 같은 방식을 사용하고 있습니다.

all_title 리스트에는 모든 특허명이 들어가 있습니다.

all_content에는 검색기능 5번에 쓰일 자료들이 들어갑니다. 218번째줄에 soup.prettify()라는 식을 사용하게 되는데 이 식은 태그와 태그안에 있는 정보들을 일렬로 정렬해서 출력해줍니다. 근데 split 된 형태가 아니기 때문에 사용하기가 쉽습니다. 이렇게 저장한 자료는 5번 검색기능에서 쓰입니다.

temporary_list는 5번 검색에서 쓰입니다.

```
  </ax2102:claim>
 </ax2102:claimInfo>
</ax2102:claimInfoArray>
<ax2102:designatedStateInfoArray type="kr.or.kipris.plus.webservice.services.patentbean.DesignatedStateInfoArray"/>
<ax2102:familyInfoArray type="kr.or.kipris.plus.webservice.services.patentbean.FamilyInfoArray">
 <ax2102:familyInfo type="kr.or.kipris.plus.webservice.services.patentbean.FamilyInfo">
  <ax2102:familyApplicationNumber>
  </ax2102:familyApplicationNumber>
 </ax2102:familyInfo>
</ax2102:familyInfoArray>
<ax2102:imagePathInfo type="kr.or.kipris.plus.webservice.services.patentbean.ImagePathInfo">
 <ax2102:docName/>
 <ax2102:largePath/>
 <ax2102:path/>
</ax2102:imagePathInfo>
<ax2102:internationalInfoArray type="kr.or.kipris.plus.webservice.services.patentbean.InternationalInfoArray">
 <ax2102:internationalInfo type="kr.or.kipris.plus.webservice.services.patentbean.InternationalInfo">
  <ax2102:internationOpenDate>
   2016.12.29
  </ax2102:internationOpenDate>
  <ax2102:internationOpenNumber>
   WO2016208225
  </ax2102:internationOpenNumber>
  <ax2102:internationalApplicationDate>
   2016.03.03
```

<prettify 예시>

```
while True:
    try:
        searchnum = int(input("————————————————————————\n원하시는 검색기능을 선택해주세요\n
키워드포함 특허목록출력\n3) 특정 발명자가 출원한 특허목록 출력\n4) 통계자료 보기\n5) 특정 단어가 들어있는 정
        if searchnum == 1:
            fix(all_applicantname)
        elif searchnum == 2:
            yudonum = 0
            word_save_list = []
            wantword=(input("————————————————\n원하시는 단어를 입력해주세요\n——————————
            for word in all_title:
                if wantword in word:
                    word_save_list.append(word)
            if len(word_save_list) == 0:
                print("* '"+wantword+"' 의 검색 결과가 없습니다. *\n")
                yudonum = 1
            if yudonum == 0:
                researchnum = int(input("————————————————\n [1번] 현재 검색결과 inventiontit
                if researchnum == 1:
                    word_save_list=list(set(word_save_list))
                    list_print_save(word_save_list)
                elif researchnum ==2:
                    print(word_save_list)
                    word_word_save_list=[]
                    wantword2 = input("————————————————————————————\n'위의 리스트는 "+
                    for word in word_save_list:
                        if wantword2 in word:
                            word_word_save_list.append(word)
                    if len(word_word_save_list) != 0:

                        word_word_save_list=list(set(word_word_save_list))
```

```
                    list_print_save(word_word_save_list)
                elif len(word_word_save_list) ==0:
                    print("*",wantword,"-",wantword2,"의 결과가 없습니다.\n")
        elif searchnum == 3:
            fix(all_agentname)

        elif searchnum==4:
            num = int(input("————————————\n1) 발명인에 대한 통계 자료 보기\n2) 출원인에 대한 통계 자료
            if num == 1:
                result = dict()
                for i in agent_toggae_list:
                    result[i] = agent_toggae_list.count(i)
                num2 = int(input("————————————————\n1) 가나다순으로 출력해서 보기\n2) 횟수가 많은 순으
                if num2 == 1:
                    ganada = sorted(result.items())
                    print(ganada)
                    makefile(ganada)
                elif num2 ==2:
                    numnum = sorted(result.items(), key = operator.itemgetter(1,0), reverse = True)
                    print(numnum)
                    makefile(numnum)
                elif num2 ==3:
                    want_word2 = input("————————————————\n원하시는 단어를 입력해주세요\n——————————
                    for key,value in result.items():
                        if key == want_word2:
                            print(key,':',value)
                elif num2==4:
                    print(toggae(1,agent_toggae_list))
                elif num2==5:
                    notice("원하시는 발명자를 모두 차례대로 입력해주세요\n원하시는 모든 발명자의\n특허의 개수를 그
                    print(toggae(2,agent_toggae_list))
```

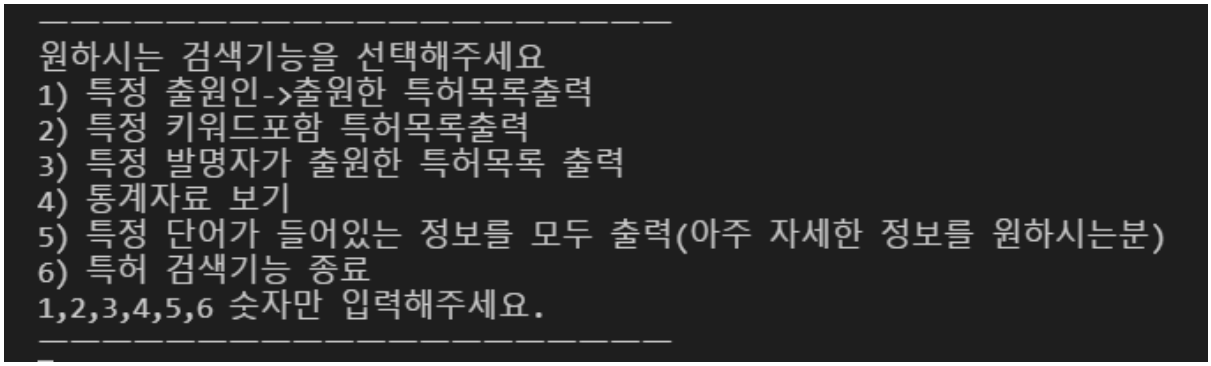
```
299             elif num==2:
300                 result = dict()
301                 for i in applicant_toggae_list:
302                     result[i] = applicant_toggae_list.count(i)
303                 num2 = int(input("————————————————————\n1) 가나다순으로 출력해서 보기\n2) 횟수가 많은 순으
304                 if num2 == 1:
305                     ganada = sorted(result.items())
306                     print(ganada)
307                     makefile(ganada)
308                 elif num2 ==2:
309                     numnum = sorted(result.items(), key = operator.itemgetter(1,0), reverse = True)
310                     print(numnum)
311                     makefile(numnum)
312                 elif num2 ==3:
313                     want_word2 = input("————————————————————\n원하시는 단어를 입력해주세요\n————————
314                     for key,value in result.items():
315                         if key == want_word2:
316                             print(key,':',value)
317                 elif num2==4:
318                     print(toggae(1,applicant_toggae_list))
319                 elif num2==5:
320                     notice("원하시는 출원인을 모두 차례대로 입력해주세요\n원하시는 모든 출원인의\n특허의 개수를 그
321                     print(toggae(2,applicant_toggae_list))
322 
323         elif searchnum==5:
324             word = input("찾고 싶은 단어가 무엇인가요??\n")
325             for i in range(0,len(all_content_save)):
```

```
326                 if word in all_content_save[i]:
327                     a=all_content_save[i]
328                     text = re.sub('</.+?>', '',a, 0, re.I|re.S)
329                     text2 = re.sub('<.+?:', '',text, 0, re.I|re.S)
330                     text3 = re.sub('type="kr.or.kipris.plus.webservice.services.patentbean.+?>', '',text2, 0, re.I|re.S)
331                     textxx = text3
332                     text3 = re.sub('>','',text3)
333                     temporary_list.append(text3)
334                     view_list.append(alltitle[i])
335             view_list=list(set(view_list))
336             print(view_list)
337 
338             accurate_num=input("' "+word+" '가 들어 있는 inventiontitle의 모든 목록 입니다.\n 어떤 발명에 대한 정보를 출력할까요?\n
339             text3=temporary_list[view_list.index(accurate_num)]
340             text3=text3.split('\n')
341             mid=[]
342             last=[]
343             for i in range(len(text3)):
344                 if text3[i].strip() != '':
345                     mid.append(text3[i].strip())
346             for i in range(4,len(mid)-1):
347                     print(mid[i])
348                     last.append(mid[i])
349 
350             print(makefile(last))
351         elif searchnum==6:
352             notice("검색기를 이용해주셔서 대단히 감사합니다.\n건국대학교\n산업공학과_201611145_박민성\nLECTURE : Programming 1\n지도
353             break
354     except  ValueError:
355         print("————————————————————\n 숫자만 입력해주세요.\n 정확한 숫자를 입력해주세요\n 초기화면으로 돌아갑니다.\n————————
```

237 ~ 355번째줄: 검색 기능입니다.

전체를 while 문으로 감쌌습니다. 6번을 누르게 되면 검색기 종료인데, break 문을 통해서 빠져 나왔습니다.

숫자가 들어갈 자리에 문자를 입력 했을 때, valueError로 잡아줍니다.



searchnum을 입력 받아서 각 기능으로 넘어갑니다.

1번] **241 ~ 242번째줄**: 앞에 fix함수에 all_applicantname 리스트를 넣어줍니다. 이 리스트를 기반으로 단어를 input 받아서, 원하는 출원인의 inventiontitle을 출력받습니다.

2번] **243 ~ 270번째줄**: all_title에는 모든 inventiontitle이 들어 있는데, wantword 에 원하시는 단어를 입력 받은 후, all title에서 찾아줍니다.

250 ~ 253번째줄: all_title에 wantword와 일치하는 정보가 없으면 검색 결과가 없다고 프린트해줍니다. yudonum을 통해, 일치하는 경우가 있다면 다음 if문으로 넘어가게 합니다.

254 ~ 270번째줄: wantword + wantword2로 검색이 가능합니다.

Ex) 시스템 - > 전기 , 시스템을 포함하는 모든 특허명을 리스트에 저장 후, 리스트를 출력해줍니다. 이 리스트를 본후, '전기'라는 wantword2를 입력하면, 저장 된 리스트에서 '전기'를 포함하는 특허명을 찾아줍니다.

원리는 앞 코드와 같습니다.

3번] **271 ~ 272번째줄**: 1번] 과 같습니다.

4번] **274 ~ 321번째줄**: num =1 을 입력하게 되면, 발명자에 대한 통계자료를 볼 수 있고

Num=2 를 입력하게 되면, 출원인의 대한 통계자료를 볼 수 있습니다.

276 ~ 298번째줄: 발명자에 대한 통계 자료

result라는 딕셔너리에 agent_toggae_list의 발명자들의 이름과 횟수를 넣어줍니다.

새로운 num2를 받아서 1. 가나다 순 정렬, 2. 특허가 많은 순으로 정렬, 3. 특정 인물의 특허 개수 출력, 4. 특허 개수 상위 10명의 그래프 보기, 5.원하는 발명인만 모아서 특허 개수 그래프로 보기를 선택해줍니다

282 ~ 284번째줄: sorted를 써서 간단하게 key값을 가나다 순으로 정렬해줍니다.

286 ~ 288번째줄: import operator를 이용해서, 딕셔너리 value 값에 대한 정렬을 해줍니다.

290 ~ 293번째줄: 딕셔너리 key,value를 이용해서 해당 key값의 value를 찾아서 출력해줍니다.

295번째줄: 함수 toggae를 이용해서, 특허 개수 상위 10명을 그래프를 표현해줍니다.

297 ~ 298번째줄: 원하는 발명자만 입력해서 특허개수를 그래프로 표현해줍니다.

299 ~ 321번째줄: 출원인에 대한 정보입니다. 앞의 276~298번째와 동일한 방식입니다.

5번] 자료에 '특정 단어' 가 하나라도 들어간다면, 그 자료에 특허명을 리스트로 보여주고, 리스트에서 자료 하나를 선택하면 자료의 전문을 보여주는 검색 기능입니다.

아까 저장했던 **218번째줄** all_content_save 리스트를 활용합니다.

```
applicationNumber
10-2017-7006212
claimCount
18
examinerName
진민숙
finalDisposal
등록결정(일반)
inventionTitle
미디어 컨텐트 송수신 방법 및 그를 이용한 송수신 장치
inventionTitleEng
MEDIA CONTENT TRANSCEIVING METHOD AND TRANSCEIVING  APPARATUS USING SAME
openDate
2017.03.17
openNumber
10-2017-0030651
originalApplicationDate
2012.06.20
originalApplicationKind
국제출원
originalApplicationNumber
10-2013-7032157
originalExaminationRequestDate
2017.03.06
```

5번 예시 (내용이 엄청 길어서 단편적인 사진입니다.)

325 ~ 348번째줄: prettify 한 자료를 더 예쁘게 다듬어 줍니다.

327 ~ 334번째줄: re.sub을 활용하여 태그를 다 제거해주고, 지저해 보이는 링크도 없애줍니다

Word가 들어가는 모든 자료를 view_list에 저장 한후,

리스트에서 한 특허명을 골라서 accurate_num에 입력을 받습니다.

그리고 그 특허명의 인덱스를 찾아서, 그 자료만 찾아서, 출력해줍니다.

344번째줄: 양 옆, 빈 공간을 다 없에줍니다. 그리고 빈 자료는 저장하지 않습니다.

346 ~ 348번째줄: 0,1,2,3 그리고 마지막 자료가 쓸모 없는 자료이기 때문에 날려줍니다.

6] break 문을 통해 검색기를 종료해줍니다.

22페이지나 되는 긴 글을 읽어 주셔서 감사합니다. 프로그래밍이 처음이고, 실력이 많이 좋지 않아서 하드코딩을 하게 되었는데, 보시기가 너무 불편 할 것 같아서 많이 걱정이 됩니다.

처음에는 어떻게 해야 될지 감도 안오고, FOR문 사용법도 몰랐는데, 개인과제를 통해서 실력이 많이 좋아 진 것 같고, 프로그래밍 언어와도 친해진 것 같아서 뿌듯합니다.

도움을 주신 **윤장혁 교수님**, **오승현 조교님**, **이지호 조교님** 정말 **감사드립니다**.