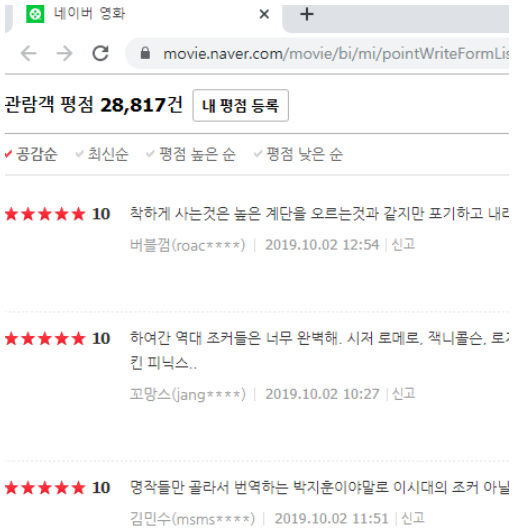
빅데이터 마이닝

14주 (목) 형태소 분석과 워드 클라우드

자연어 처리와 워드 클라우드 과정

Web crawling (beautiful soup)



형태소분석 (koNLPy)

```
[('영화', 'Noun'), ('끝나고도', 'Verb'), ('여운',
'Noun'), ('이', 'Josa'), ('가시', 'Noun'), ('지',
'Josa'), ('않아', 'Verb'), ('멍하게', 'Adjective')
, ('있었습니다', 'Adjective'), ('.',
'Punctuation'), ('화려한', 'Adjective'), ('액션',
'Noun'), ('대신', 'Noun'), ('숨막히는',
'Adjective'), ('설계', 'Noun'), ('가', 'Josa'), ('
있습니다', 'Adjective'), ('그리고', 'Conjunction')
```

시각화 (wordcloud)

```
[('영화', 522), ('조커', 423), ('연기', 168), ('사
람', 127), ('이', 118), ('호아킨', 108), ('그',
92), ('것', 86), ('진짜', 83), ('피닉스', 81), ('
수', 79), ('때', 73), ('보고', 65), ('내', 63), ('
생각', 62), ('정말', 58), ('말', 58), ('사회',
53), ('최고', 50), ('있는', 48), ('인생', 47), ('
그냥', 47), ('왜', 45), ('이해', 45), ('히스',
44), ('장면', 43), ('연기력', 42), ('더', 42), ('
명작', 39), ('미친', 38), ('레저', 37), ('몰입',
```

자연어 처리를 위한 개발 환경 설정

1

웹 크롤링과 시각화를 위해 필요한 모듈

웹 크롤링 모듈

- beautifulsoup4 (웹 크롤링 라이브러리)
- JPype1 : Python-Java bridge, 자바를 파이썬 에서 사용할 수 있는 모듈, 이 모듈을 설치하기 전에 자바개발환경(JDK)이 먼저 설치되어 있어야 함.

한국어 텍스트 분석 모듈

- koNLPy : 한국어 자연어처리(Natural Language Processing)를 위한 패키지

  - Okt(구 twitter) : 형태소 분석 모듈

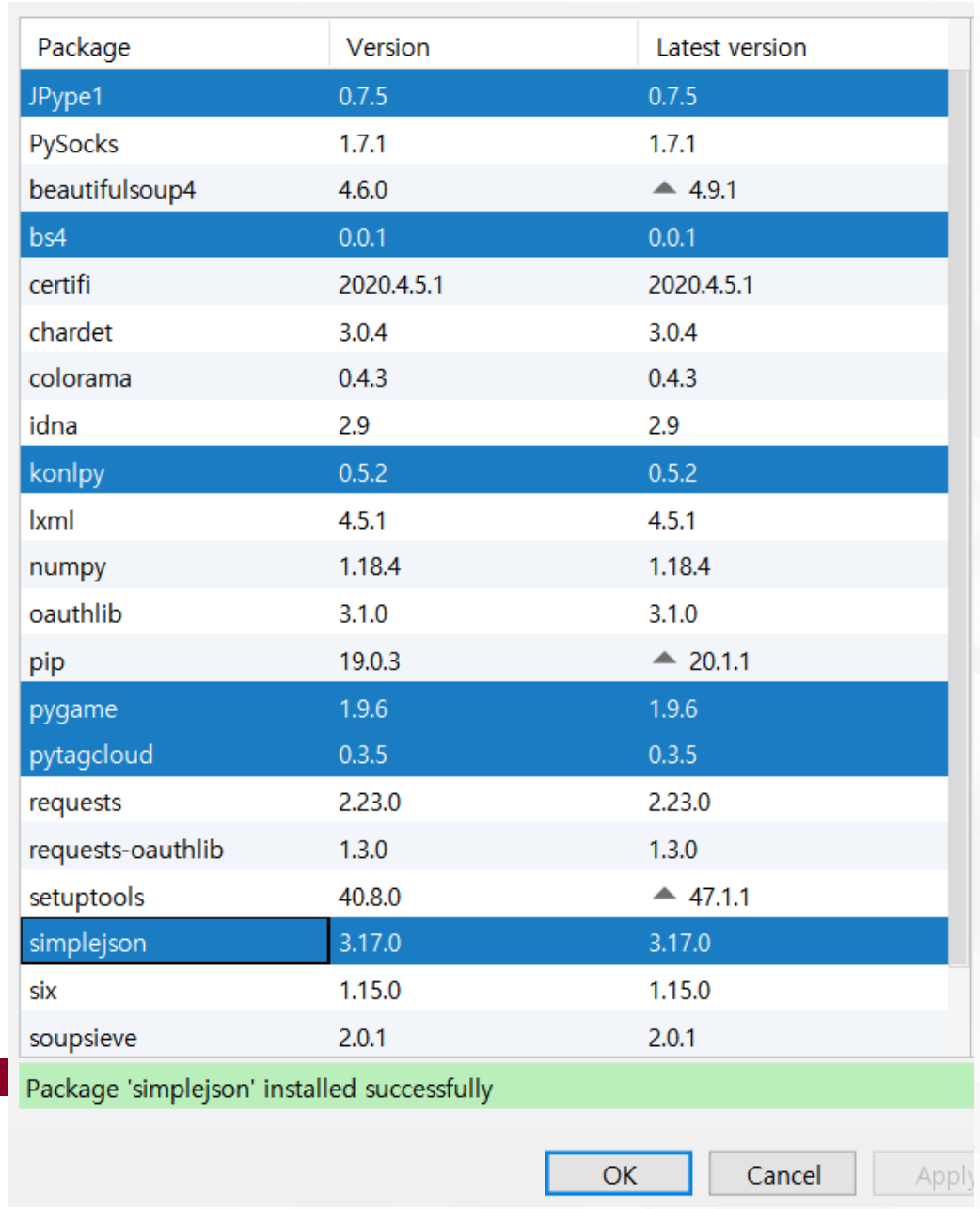
단어 시각화 모듈

- pytagcloud : 이미지나 HTML형태의 단어구름으로 만들어 주는 모듈
- pygame : Python Game 개발 모듈로 워드클라우드 구현에 필요
- simplejson : 폰트 파일 등록을 위한 자료구조 파일 처리

웹 크롤링과 시각화를 위해 필요한 모듈

모듈 설치

- JDK (Java SE Downloads)
- JAVA_HOME 설정
- JPype1설치(파이참)
- bs4 설치(파이참)
- KoNLPy 설치 (파이참)
- pygame 설치 (파이참)
- pytagcloud 설치 (파이참)
- simplejson설치 (파이참)



설치 - JDK (Java SE Downloads)

- http://goo.gl/vPPjjM [접속]하고, JDK 다운로드

https://www.oracle.com/java/technologies/javase-jdk15-downloads.html

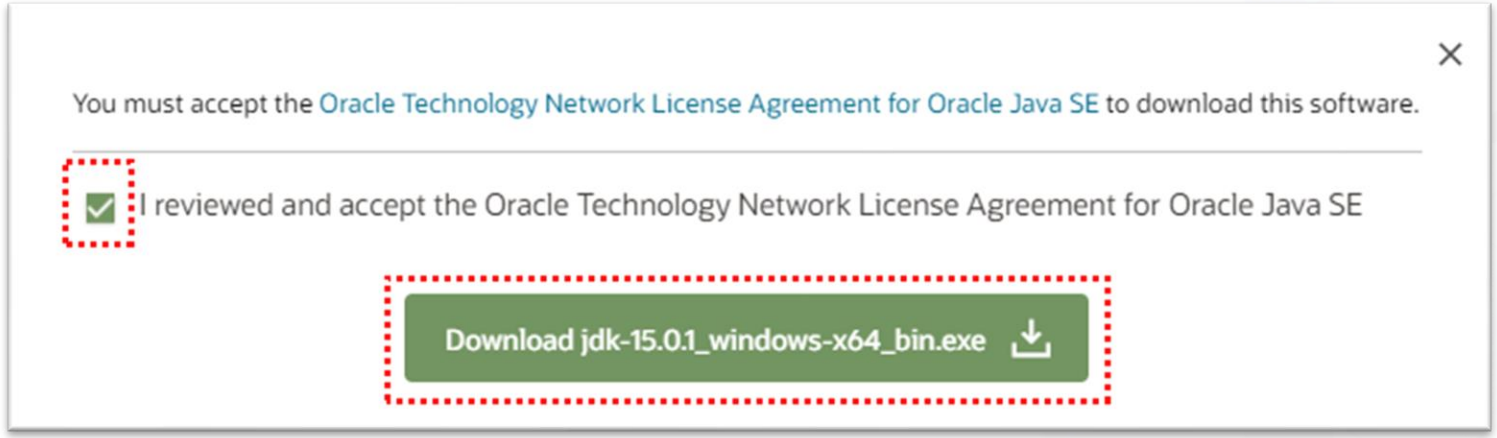
Java SE Development Kit 15.0.1

This software is licensed under the Oracle Technology Network License Agreement for Oracle Java SE

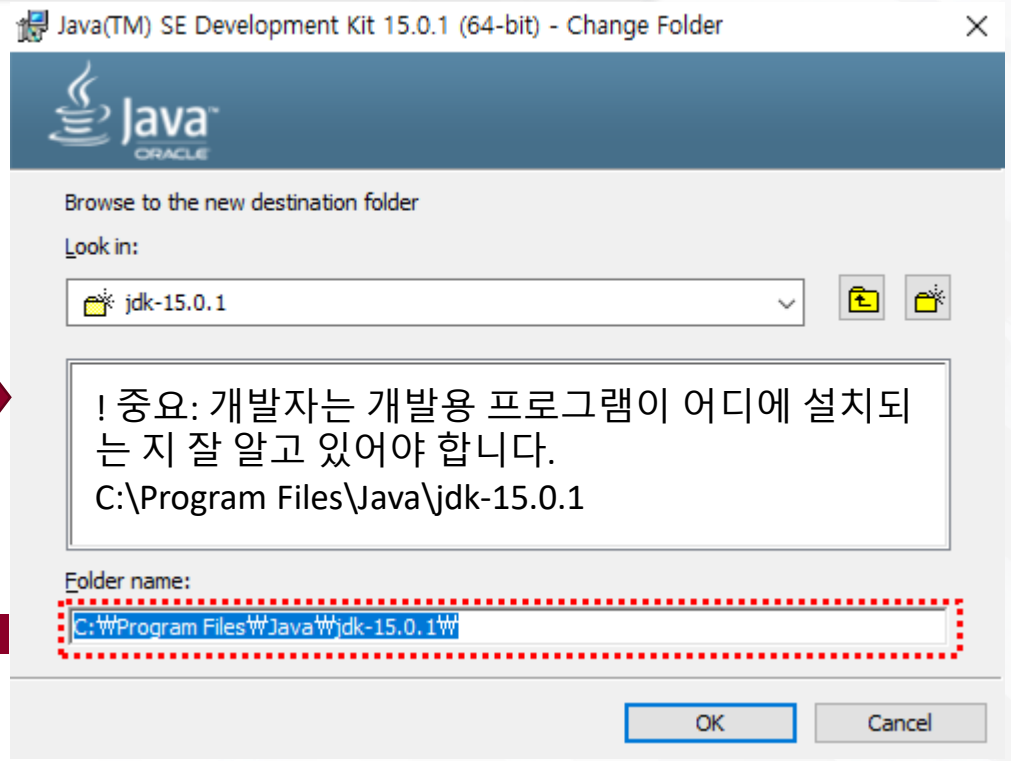
Product / File Description	File Size	Download
Linux ARM 64 RPM Package	141.81 MB	jdk-15.0.1_linux-aarch64_bin.rpm
Linux ARM 64 Compressed Archive	157.01 MB	jdk-15.0.1_linux-aarch64_bin.tar.gz
Linux x64 Debian Package	154.79 MB	jdk-15.0.1_linux-x64_bin.deb
Linux x64 RPM Package	162.02 MB	jdk-15.0.1_linux-x64_bin.rpm
Linux x64 Compressed Archive	179.33 MB	jdk-15.0.1_linux-x64_bin.tar.gz
macOS Installer	175.94 MB	jdk-15.0.1_osx-x64_bin.dmg
macOS Compressed Archive	176.53 MB	jdk-15.0.1_osx-x64_bin.tar.gz
Windows x64 Installer	159.69 MB	jdk-15.0.1_windows-x64_bin.exe
Windows x64 Compressed Archive	179.27 MB	jdk-15.0.1_windows-x64_bin.zip

설치 - JDK (Java SE Downloads)

- 라이선스 동의 후 파일다운로드

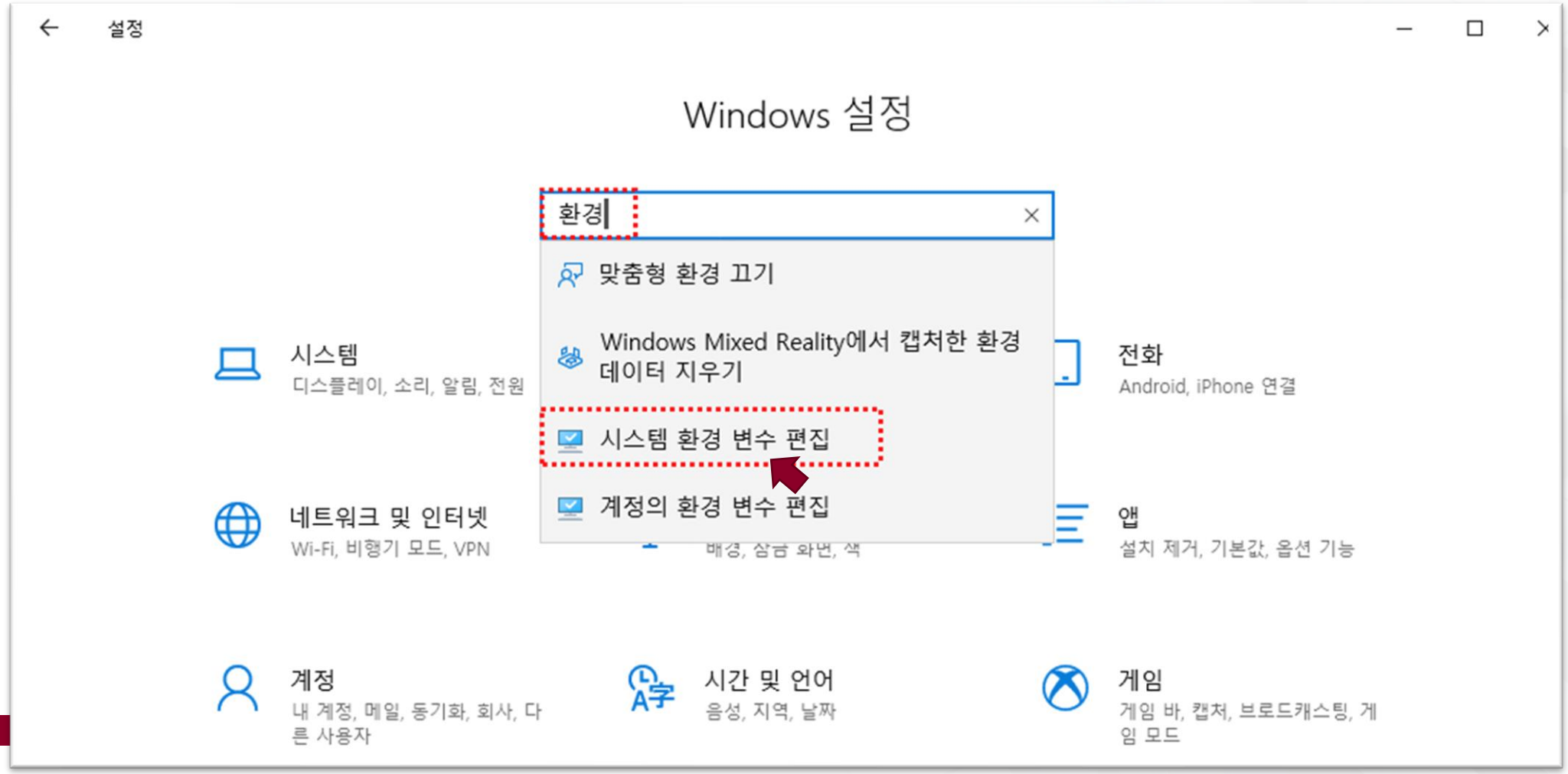


- JDK 설치



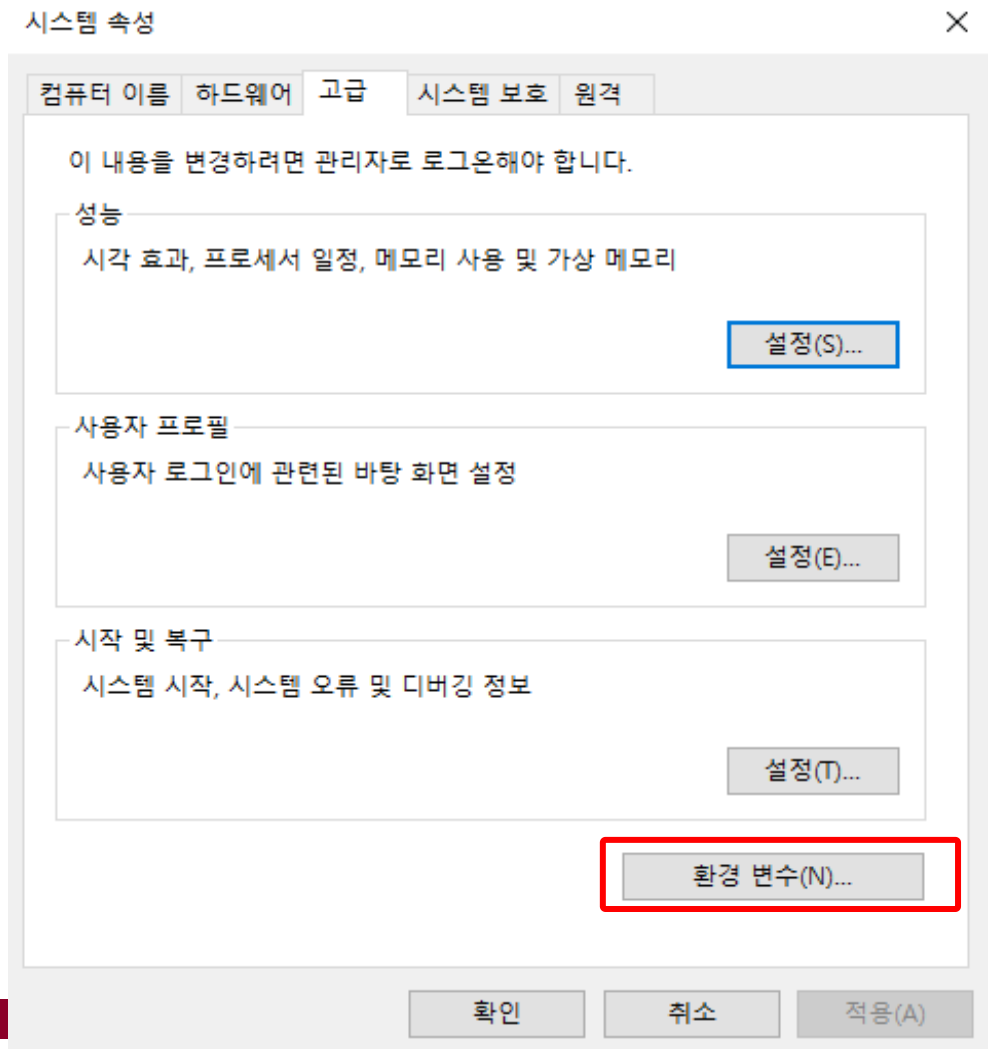
JAVA_HOME 경로 설정

- 시스템 - 고급 시스템 설정



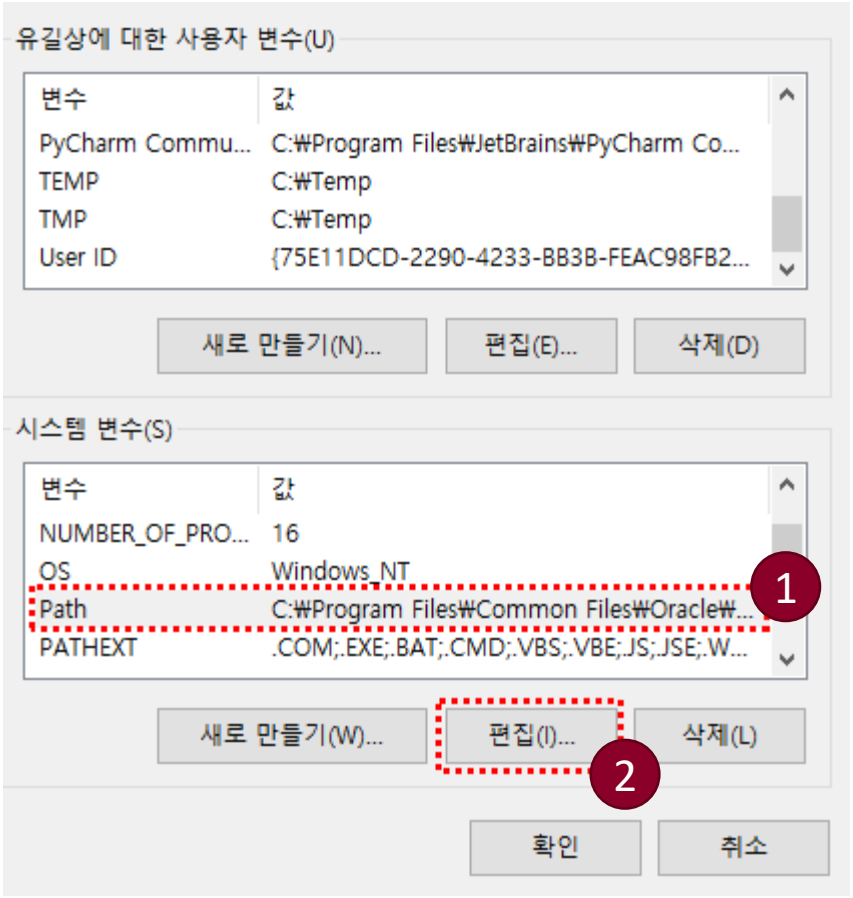
JAVA_HOME 경로 설정

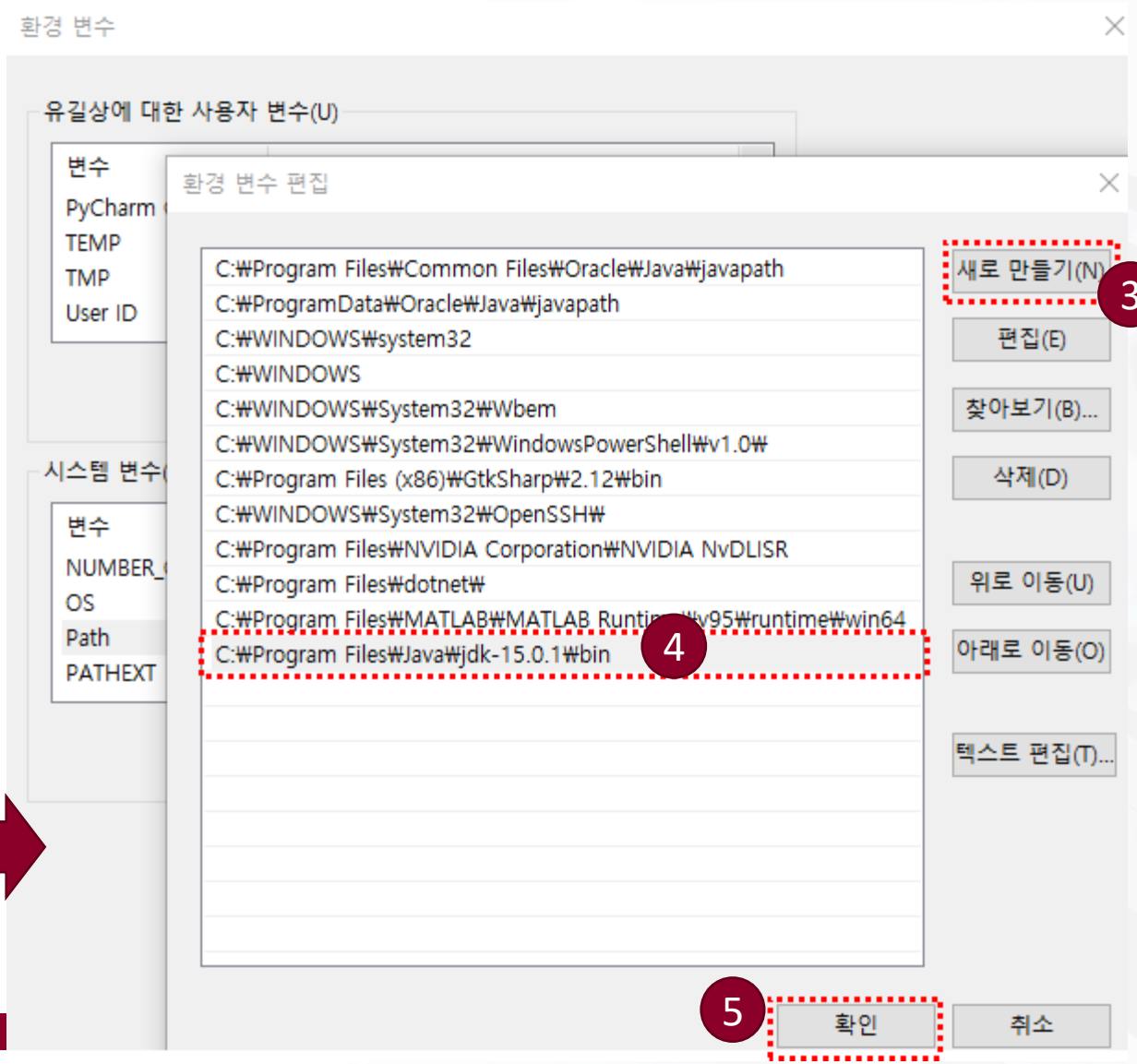
- 환경 변수 선택



JAVA 경로 설정(윈도우 10기준)

- 경로 지정

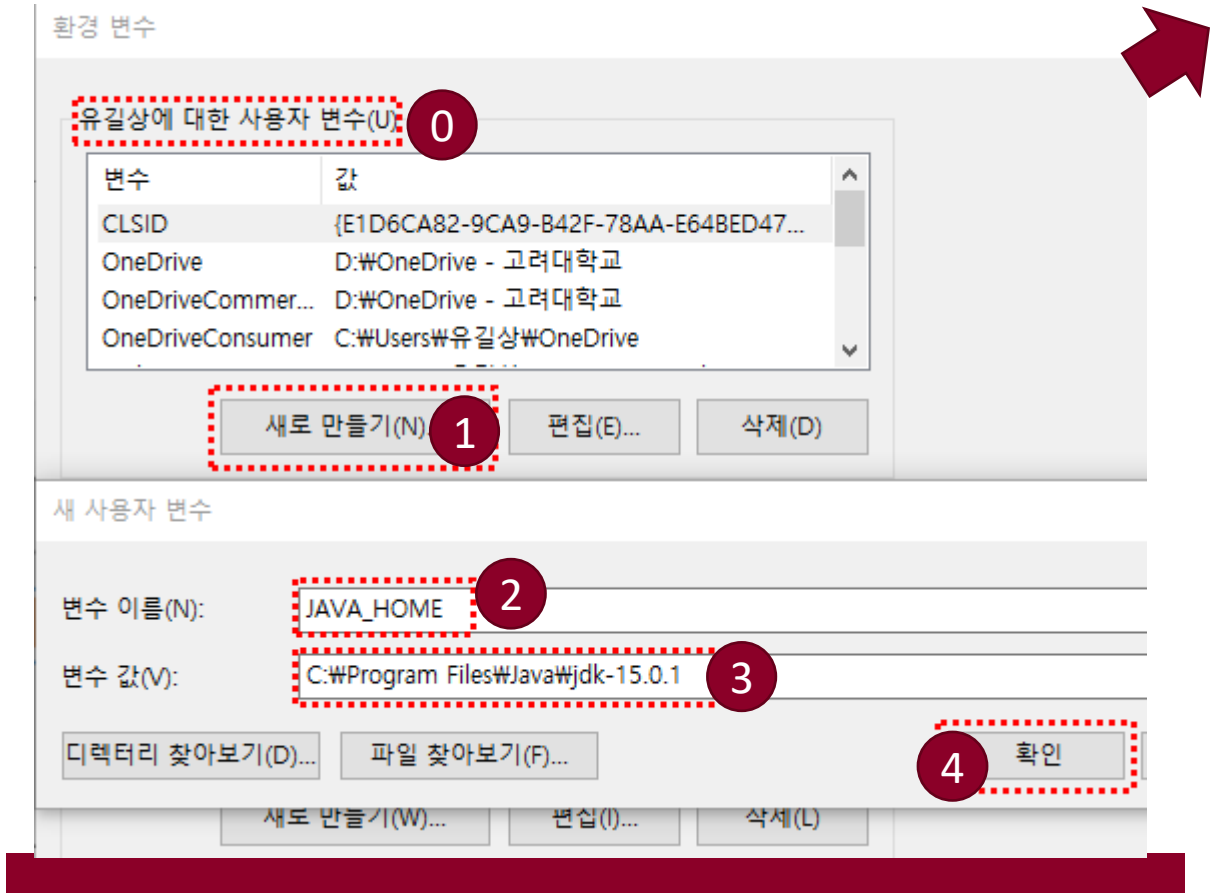


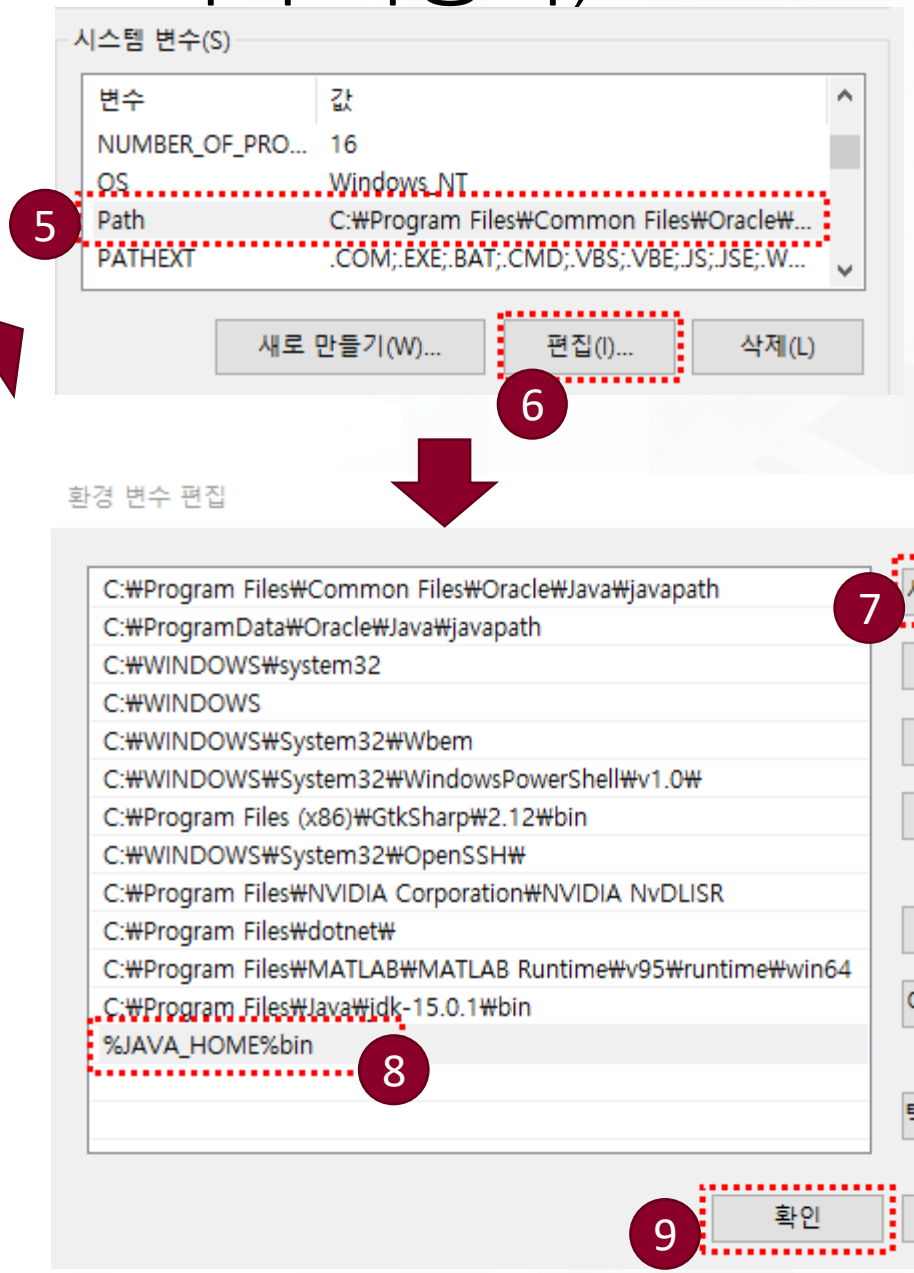


JAVA_HOME 경로 설정(윈도우 10 이하 사용자)

- JAVA_HOME 직접 입력

  C:\Program Files\Java\jdk-15.0.1





JAVA설치 확인

- 경로 지정 후 컴퓨터를 재 부팅
- 재 부팅 후 파이참 터미널 창에서(또는 cmd창) 아래와 같이 입력 후 설치 여부 확인

```
(venv) C:\python\ch13_crawing>java -version
java version "15.0.1" 2020-10-20
Java(TM) SE Runtime Environment (build 15.0.1+9-18)
Java HotSpot(TM) 64-Bit Server VM (build 15.0.1+9-18, mixed mode, sharing)
```

형태소 분석

2

https://konlpy.org/en/latest/

한글 자연어 처리 방법1 - 꼬꼬마 모듈

다른 분석기에 비해 분석 시간이 가장 오래 소요됨

```
from konlpy.tag import Kkma # 꼬꼬마 모듈 사용
kkma = Kkma()
my_text = '한국어 분석을 시작합니다 재미있어요~~'
print(kkma.sentences(my_text)) # 문장 분석
print(kkma.nouns(my_text))  # 명사 분석
print(kkma.pos(my_text))    # 형태소 분석
```

['한국어 분석을 시작합니다', '재미있어요~~']

['한국어', '분석']

[('한국어', 'NNG'), ('분석', 'NNG'), ('을', 'JKO'), ('시작하', 'VV'), ('ㅂ니다', 'EFN'), ('재미있', 'VA'), ('어요', 'EFN'), ('~~', 'SW')]

한글 자연어 처리 방법2 - 한나눔 모듈

결과에 대한 품질이 문장에 따라 차이가 큼(너무 잘게 분석하거나 아예 분석을 못함)

```
from konlpy.tag import Hannanum # 한나눔 모듈 사용
hnn = Hannanum()
my_text = '한국어 분석을 시작합니다 재미있어요~~'
print(hnn.nouns(my_text))  # 명사 분석
print(hnn.morphs(my_text)) # 다른 모듈과 차이점 확인
print(hnn.pos(my_text))    # 형태소 분석
```

['한국어', '분석', '시작']

['한국어', '분석', '을', '시작', '하', 'ㅂ니다', '재미있', '어요', '~~']

[('한국어', 'N'), ('분석', 'N'), ('을', 'J'), ('시작', 'N'), ('하', 'X'), ('ㅂ니다', 'E'), ('재미있', 'P'), ('어요', 'E'), ('~~', 'S')]

한글 자연어 처리 방법3 - Open Korean Text 모듈

단어 클라우드 분석에 활용하는 경우 가장 적합

```
from konlpy.tag import Okt   # okt 모듈
t = Okt()
my_text = '한국어 분석을 시작합니다 재미있어요~~'
print(t.nouns(my_text))   # 명사 분석 for word cloud
print(t.morphs(my_text))
print(t.pos(my_text))     # 형태소 분석
```

다른 모듈과 차이 확인

['한국어', '분석', '시작']

['한국어', '분석', '을', '시작', '합니다', '재미있어요', '~~']

[('한국어', 'Noun'), ('분석', 'Noun'), ('을', 'Josa'), ('시작', 'Noun'), ('합니다', 'Verb'), ('재미있어요', 'Adjective'), ('~~', 'Punctuation')]

한글 자연어 처리 방법4 - Komoran 모듈

자소 분리나 오탈자에 대해서도 어느 정도 분석 품질이 좋음

```
from konlpy.tag import Komoran  # Komor
k = Komoran()
my_text = '한국어 분석을 시작합니다 재미있어요~~'
print(k.nouns(my_text))  # 명사 분석
print(k.morphs(my_text))
print(k.pos(my_text))    # 형태소 분석
```

['한국어', '분석', '시작']

['한국어', '분석', '을', '시작', '하', 'ㅂ니다', '재미있', '어요', '~', '~']

[('한국어', 'NNP'), ('분석', 'NNG'), ('을', 'JKO'), ('시작', 'NNG'), ('하', 'XSV'), ('ㅂ니다', 'EC'), ('재미있', 'VA'), ('어요', 'EC'), ('~', 'SO'), ('~', 'SO')]

형태소분석(koNLPy)

처리 과정

- ✓ 크롤링한 댓글파일을 불러와서 리스트 변수에 저장
- ✓ konlpy 모듈호출 및 Okt 객체생성
- ✓ 반복문을 사용하여 문장별 형태소구분 및 품사매칭 (koLNPy함수)
- ✓ [응용실습] 필요한 품사만 추출해보기
- ✓ [응용실습] 선별된 품사별 빈도수 계산하고 상위 빈도 10위 까지 출력해 보기

형태소분석(koNLPy)

- 크롤링한 댓글파일 가져와서 리스트 변수에 저장

```
file = open('naver_opinion.txt', 'r', encoding='utf-8')
total_lines = file.readlines() # 외부 텍스트 파일 한 번에 모두 읽기
file.close()
print(total_lines)    # 결과 확인 → 개행문자 포함되어 읽혀 짐

reply_text = []
for line in total_lines:
    reply_text.append(line[:-1])

print(reply_text)     # 결과 확인 -→ 개행문자가 제거
```

- ✓ **Read() # 문자열로 가져오기**
- ✓ **Readline() # 한줄 문자열로 가져오기**
- ✓ **Readlines() # 여러줄을 리스트로 가져오기**

형태소분석(koNLPy)

- **Okt 객체 생성**

```
from konlpy.tag import Okt

ok_twitter = Okt()
```

형태소분석(koNLPy)

■ 형태소 분류하고 확인해 보기

```
sentences_tag = []
for sentence in reply_text:
    morph = ok_twitter.pos(sentence)
    sentences_tag.append(morph)
    print(morph)              # 분석된 1건 결과 확인
    print('-' * 30)

print(sentences_tag)  # 형태소 처리된 결과 확인
print(len(sentences_tag))    # 태그 개수 확인
```

[('휴잭맨', 'Noun'), ('의', 'Josa'), ('.', 'Punctuation'), ('매력', 'Noun'), ('이', 'Josa'), ('한껏', 'Adverb'), ('돋보인', 'Verb'), ('영화', 'Noun'), ('입니다', 'Adjective'), ('.', 'Punctuation'), ('다른', 'Noun'), ('출연자', 'Noun'), ('들', 'Suffix'), ('도', 'Josa'), ('너무', 'Adverb'), ('멋지구요', 'Adjective'), ('.', 'Punctuation'), ('사람', 'Noun'), ('은', 'Josa'), ('누구', 'Noun'), ('나', 'Josa'), ('소중하며', 'Adjective'), (',', 'Punctuation'), ('현재', 'Noun'), ('를', 'Josa'), ('멋지게', 'Adjective'), ('사는게', 'Verb'), ('중요하다는', 'Adjective'), ('메세지', 'Noun'), ('도', 'Josa'), ('있구요', 'Adjective'), ('.', 'Punctuation'), ('좋았습니다', 'Adjective')]

형태소분석(koNLPy)

■ 필요한 품사만 추출해보기

✓ **명사만 출력해 보기**

```
for my_sentence in sentences_tag:
    for word, tag in my_sentence:
        if tag in ['Noun']:
            print(word)
```

```
휴잭맨
매력
영화
다른
출연자
사람
누구
현재
메세지
정말
인생
최고
```

형태소분석(koNLPy) 과정

■ 필요한 품사만 추출해보기

✓ **명사를 버켓리스트에 담기**

```
bucket_list = []
for my_sentence in sentences_tag:
    for word, tag in my_sentence:
        if tag in ['Noun']:
            bucket_list.append(word)
            print(word)

print(bucket_list)
```

['휴잭맨', '매력', '영화', '다른', '출연자', '사람', '누구', '현재', '메세지', '정말', '인생', '최고', '영화', '남', '기립박수', '뻔', '강추', '노래', '진짜', '쩐다', '내내', '호강', '배우', '연기', '올해', '마무리', '영화로', '시작', '얼마', '꿈', '얘기', '눈물', '그', '꿈', '허황', '때문', '수', '그', '꿈', '때문', '생각', '음악', '시작', '처음', '잭', '에프', '론', '목', '좀', '전개', '스토리', '아주', '감동', '실화', '영화인', '것', '생각', '더욱', '가족', '아주', '사람', '아주', '영화', '명대사', '기대', '이상', '습', '라라', '랜드', '뮤지컬', '쇼맨', '수', '위', '한해', '마무리', '세상', '역시', '용기', '생각', '다시', '해', '주어', '연말', '가족', '생각', '거', '기대', '이상', '노래', '때', '온몸', '소름', '진짜', '라라', '랜드', '잼남', '인생', '영화', '하나', '또', '연말', '가족', '끼리', '한번', '더', '듯', ' ' 한 . . .

형태소분석(koNLPy) 과정

■ 단어 빈도수 구하기

✓ **각 원소의 출현 횟수를 계산하는 Counter 모듈을 활용한다.**

```
from collections import Counter
counts = Counter(bucket_list)
print(counts)
```

✓ **명사 빈도순서데로 상위 30개 출력해보기**

```
print(counts.most_common(20))
```

Counter({'영화': 1106, '노래': 349, '최고': 320, '음악': 216, '정말': 192, '감동': 191, '인생': 183, '진짜': 179, '뮤지컬': 168, '번': 151, '스토리': 131, '꼭': 106, '더': 102, '또': 100, '눈': 94, '휴잭맨': 93, '보고': 91, '영화관': 81, '말': 75, '다시': 74, '하나': 72, '추천': 70, '시간': 69, '것': 68, '연기': 67, '수': 64, '한번': 62, '생각': 54, '이': 54, '내내': 53, '모두': 53, '처음': 52, '배우': 51, '가족': 50, '계속': 50, '라라': 49, '랜드': 49, '뮤지컬영화': 49, '영상': 49, '볼': 49, '귀가': 46, '춤': 46, '사람': 45, '내용': 45, '올해': 44, '소름': 44, '그': 41, '보기': 41, '강추': 40, '역시': 39, '. . .

[('영화', 1106), ('노래', 349), ('최고', 320), ('음악', 216), ('정말', 192), ('감동', 191), ('인생', 183), ('진짜', 179), ('뮤지컬', 168), ('번', 151), ('스토리', 131), ('꼭', 106), ('더', 102), ('또', 100), ('눈', 94), ('휴잭맨', 93), ('보고', 91), ('영화관', 81), ('말', 75), ('다시', 74)]

✓ **명사와 형용사를 모두 추출하고 상위 50개를 출력해 봅시다**

워드 클라우드 3

워드 클라우드

- 워드 클라우드(word cloud)는 자주 나타나는 단어들의 크기를 상대적인 크기로 시각화하여 보여줌으로써 직관적으로 많이 사용되는 단어의 영향력을 알 수 있음
- 필요한 라이브러리

  konlpy.tag(형태소 분석을 통해 명사 추출)

  pytagclud(워드클라우드 그림 그리기)

  random 모듈(단어색 랜덤 지정)

  Counter(단어빈도 집계)
- 필요한 폰트 파일 복사 및 json파일에 등록

C:\Windows\Fonts

C:₩python₩wordclud₩Lib₩site-packages₩pytagcloud₩fonts₩fonts.json

fonts.json

```
1 [
2     {
3         "name": "NanumBarunGothic",
4         "ttf": "NanumBarunGothicBold.ttf",
5         "web": ""
6     },
```

필요한 라이브러리 임포트

```
import pytagcloud
import random
from konlpy.tag import Okt
from collections import Counter
```

저장된 문서 불러오기

- 크롤링한 댓글파일 가져와서 리스트 변수에 저장

```
file = open('Naver_wordcloud.txt', 'r', encoding='utf-8')
reply_text = file.readlines()
file.close()

print(reply_text)
```

문장에서 명사 추출하기

- 댓글에서 명사를 추출하고 nouns변수에 저장

```
ok_twitter = Okt()
nouns = []
tags = []      # 단어의 {색상, 명사, 크기}의 속성을 저장하는 리스트
for sentence in reply_text:
    for noun in ok_twitter.nouns(sentence):
        nouns.append(noun)

count = Counter(nouns)  # 각 명사별로 빈도계산
print(nouns)
print(count)
```

```
pygame 2.0.0 (SDL 2.0.12, python 3.8.6)
Hello from the pygame community. https://www.pygame.org/contribute.html
['올라프', '편', '요약', '기', '맥힙', '크리스토퍼', '뮤비', '좀', '흠칫', '함', '미래', '보
Counter({'편': 437, '스토리': 394, '엘사': 351, '노래': 333, '영화': 319, '겨울왕국': 26
```

그림 출력 처리

■ 단어를 그림으로 출력하려면 다음과 같은 형식으로 저장되어야 함

{'color': (245, 140, 176), 'tag': ' 엘사', 'size': 278}

{ RGB컬러값, 표시할 단어명, 글자크기값}

그림 출력 처리

■ 단어빈도 집계 및 단어 색상을 랜덤으로 지정

{랜덤 color, tag, size} 형식으로 빈도순서로 100개 저장

```
for n, c in count.most_common(100):  # n : 명사, c : 빈도
    tags.append({'color': (random.randint(0, 255),
                           random.randint(0, 255),
                           random.randint(0, 255)),
                 'tag': n,
                 'size': c * 2 })
print(tags)
```

[{'color': (4, 127, 230), 'tag': '영화', 'size': 2212}, {'color': (106, 10, 135), 'tag': '노래', 'size': 698}, {'color': (23, 249, 34), 'tag': '최고', 'size': 640}, {'color': (145, 218, 229), 'tag': '음악', 'size': 432}, {'color': (169, 74, 137), 'tag': '정말', 'size': 384}, {'color': (8, 14, 51), 'tag': '감동', 'size': 382}, {'color': (194, 96, 170), 'tag': '인생', 'size': 366}, {'color': (3, 203, 103), 'tag': '진짜', 'size': 358}, {'color': (230, 193, 204), 'tag': '뮤지컬', 'size': 336}, {'color': (62, 111, 65), 'tag': '번', 'size': 302}, {'color': (100, 238, 113), 'tag': '스토리', 'size': 262}, {'color': (62, 164, 152), 'tag': '꼭', 'size': 212}, {'color': (152, 72, 3), 'tag': '더', 'size': 204}, {'color': (56, 242, 61), 'tag': '또', 'size': 200}, {'color': (74, 40, 45), 'tag': '눈', 'size': 188}, . . .

단어구름 이미지 생성

- **관심명사 워드클라우드 이미지 파일 (jpg, png)생성**

```
pytagcloud.create_tag_image(tags, # 명사모음 리스트
                            'wordcloud.png', # 저장할 파일명
                            fontname='NanumBarunGothicBold’,
                            # 출력을 위해 등록된 폰트이름
                            size=(1280, 720))
                            # 파일 크기
```

워드클라우드 파일 확인

네이버 영화 '겨울왕국2' 관람평 분석결과

의미없는 단어 제거

- **단어구름에서 의미없는 글자 제외 시키기**

```
ok_twitter = Okt()
nouns = []
tags = []
for sentence in reply_text:
    for noun in ok_twitter.nouns(sentence):
        if noun in ['영화', '도', '번', '이번', '저', '이', '것', '때', '말',
'볼', '그', '또', '애', '편이', '그냥', '부분', '좀' '더', '편']:
            pass
        else:
            nouns.append(noun)
count = Counter(nouns)  # 각 명사별로 빈도계산
```

워드클라우드 파일 확인

다음 영화 '겨울왕국2' 댓글 분석결과 (폰트 :H2GTRE.ttf , 단어 100개 출력)

워드클라우드 파일 확인

다음 영화 ‘토이스토리3’ 댓글 분석결과 (폰트 :H2GTRE.ttf , 단어 200개 출력)

워드클라우드 파일 확인

다음 영화 '겨울왕국2' 댓글 분석결과(폰트 :H2MKPB.ttf , 단어 100개 출력)

워드클라우드 파일 확인

네이버 영화 '위대한 쇼맨' 댓글 분석 결과 (폰트 :H2MKPB.ttf , 단어 150개 출력)

워드 클라우드 응용

위대한 쇼맨

토이 스토리3

겨울왕국2

Thank you