2조 김정훈,한정민,모유찬

1.Grading

(1) pagination(db,pageNum) :

(a) 매개변수 : if __name__=’__main__’에서 grade에 연결한 db instance 및 출력하려는 page 에 대해서 넘김

(b) 내부 변수로 numberoflineperpage =10 를 선언함으로써, line skip 및 한 페이지에서 보여주는 line수를 한정 함

(c) filter 조건 없이, projection 조건으로, grades와 sid 열만 출력되게 설정, skip(), limit()을 이용하여 페이지에 해당되는 내용 출력함

(d) 출력은 base code 를 이용

```
database@database-vm:~/practice/mongopractice$ python3 grading.py 1 1
{ grades:[{'type': 'homework', 'score': 69}, {'type': 'exam', 'score': 86}, {'type': 'quiz', 'score': 29}], sid:0 }
{ grades:[{'type': 'homework', 'score': 2}, {'type': 'quiz', 'score': 6}, {'type': 'exam', 'score': 67}], sid:1 }
{ grades:[{'type': 'quiz', 'score': 91}, {'type': 'homework', 'score': 13}, {'type': 'exam', 'score': 76}], sid:2 }
{ grades:[{'type': 'homework', 'score': 72}, {'type': 'quiz', 'score': 55}, {'type': 'exam', 'score': 93}], sid:3 }
{ grades:[{'type': 'homework', 'score': 17}, {'type': 'exam', 'score': 98}, {'type': 'quiz', 'score': 90}], sid:4 }
{ grades:[{'type': 'homework', 'score': 83}, {'type': 'quiz', 'score': 68}, {'type': 'exam', 'score': 98}], sid:5 }
{ grades:[{'type': 'quiz', 'score': 88}, {'type': 'exam', 'score': 90}, {'type': 'homework', 'score': 30}], sid:6 }
{ grades:[{'type': 'quiz', 'score': 6}, {'type': 'homework', 'score': 55}, {'type': 'exam', 'score': 75}], sid:7 }
{ grades:[{'type': 'quiz', 'score': 99}, {'type': 'homework', 'score': 33}, {'type': 'exam', 'score': 99}], sid:8 }
{ grades:[{'type': 'homework', 'score': 33}, {'type': 'quiz', 'score': 79}, {'type': 'exam', 'score': 46}], sid:9 }
```

```
database@database-vm:~/practice/mongopractice$ python3 grading.py 1 10
{ grades:[{'type': 'homework', 'score': 99}, {'type': 'quiz', 'score': 89}, {'type': 'exam', 'score': 27}], sid:90 }
{ grades:[{'type': 'homework', 'score': 23}, {'type': 'quiz', 'score': 74}, {'type': 'exam', 'score': 92}], sid:91 }
{ grades:[{'type': 'homework', 'score': 47}, {'type': 'quiz', 'score': 75}, {'type': 'exam', 'score': 91}], sid:92 }
{ grades:[{'type': 'homework', 'score': 84}, {'type': 'quiz', 'score': 96}, {'type': 'exam', 'score': 82}], sid:93 }
{ grades:[{'type': 'homework', 'score': 84}, {'type': 'quiz', 'score': 58}, {'type': 'exam', 'score': 75}], sid:94 }
{ grades:[{'type': 'homework', 'score': 70}, {'type': 'exam', 'score': 95}, {'type': 'quiz', 'score': 66}], sid:95 }
{ grades:[{'type': 'homework', 'score': 94}, {'type': 'quiz', 'score': 49}, {'type': 'exam', 'score': 21}], sid:96 }
{ grades:[{'type': 'homework', 'score': 41}, {'type': 'quiz', 'score': 78}, {'type': 'exam', 'score': 80}], sid:97 }
{ grades:[{'type': 'homework', 'score': 26}, {'type': 'quiz', 'score': 41}, {'type': 'exam', 'score': 83}], sid:98 }
{ grades:[{'type': 'homework', 'score': 85}, {'type': 'quiz', 'score': 78}, {'type': 'exam', 'score': 92}], sid:99 }
```

(2) letter(db):

(a) python 내 bisect package 이용하여, 점수별 letter 지정

(b) 매개변수 : if __name__=’__main__’에서 grade에 연결한 db instance 넘김

(c) 문제에서 제시해준 수식을 이용하여, total 점수 계산, (**(3) pefect에서도 total 점수를 동일하게 사용하여, 별도의 함수 gettotal로 분리 후 해당 함수를 호출하여 total 점수를 계산)

(d) 기존 정보 sid 에 total 점수와 letter정보를 저장하기 위해서 OrderedDict()를 사용하여 저장(기존에 출력하는 예시가, 열의 순서가 있어서, OrderedDict()를 사용하여서 key/value를 저장하였었음)

(e) (d)과정을 모든 line에 대해서 처리하여, 임시 list type 변수 result에 append 후 최종적으로 base코드에서 제공해준 출력 코드를 이용하여 결과 출력

(f) 핵심은 OrderedDict()를 이용하여, 결과를 저장하고, letter를 결정할때는 bisect package를 이용하여 처리하였다는 점

```
database@database-vm:~/practice/mongopractice$ python3 grading.py 2
{ letter:A, sid:54, total:91.2 }
{ letter:B, sid:16, total:88.8 }
{ letter:B, sid:5, total:87.5 }
{ letter:B, sid:99, total:87.1 }
{ letter:B, sid:72, total:86.6 }
{ letter:B, sid:93, total:85.4 }
{ letter:B, sid:63, total:84.9 }
{ letter:B, sid:23, total:83.2 }
{ letter:B, sid:44, total:82.9 }
{ letter:B, sid:52, total:82.6 }
```

(3) perfect(db):

(a) 매개변수 : if __name__='__main__'에서 grade에 연결한 db instance 넘김

(b) 위의 (1),(2)와 다르게 최종 결과를 collection 을 생성하여 저장 후 출력해야하기 때문에, 처음에 결과를 저장하는 collection 'relative'를 db내에 있는지 확인 후 만약 있다면, drop_collection명령어를 통해, 해당 relative collection을 지워줌

(c) total 점수가 필요하므로 (2) letter 때와 동일하게 미리 선언해놓은 gettotal 함수 이용하여 점수 계산, 이때 별도의 옵션이 필요한데, 100점 과목이 있거나, document내에 note 가 있으면 점수에 10점을 추가로 주는 부분이 있음, gettoal 함수에 두번째 parameter를 True로 함으로써 처리

(d) (2)과 다르게 letter를 계산하는 방식이, 전체 학생에 대해서 total 성적의 min/max를 구해야하기 때문에, 모든 학생들에 대한 total 산출이 완료된 후 letter를 계산함

(e) (2)과 마찬가지로 결과를 처음에 원하는대로 출력하기위해 OrderedDict를 이용하여서 결과를 저장하였음

(f) (c)-(e)과정에서 나온 최종 결과들은 'relative'라는 collection생성 후 insert해 줌

(g) (f)에서 저장한 결과를 find 함수를 통해서 query한 후 출력

```
database@database-vm:~/practice/mongopractice$ python3 grading.py 3
{ letter:B, sid:0 }
{ letter:D, sid:1 }
{ letter:C, sid:2 }
{ letter:B, sid:3 }
{ letter:B, sid:4 }
{ letter:A, sid:5 }
{ letter:B, sid:6 }
{ letter:C, sid:7 }
{ letter:B, sid:8 }
{ letter:C, sid:9 }
```

2.Pokemon Go

(1) problem_1

(a) 바람이가 갖고 있는 몬스터의 상세정보를 pokemon collection에서 가져옴

(b) (a)에서 불러온 정보 내에서 'weaknesses'에 있는 약점이 되는 몬스터의 목록을 local list 변수인 wind_weak 에 추가

(c) (b) 에서 불러온 몬스터의 정보를 key(몬스터 타입) / value(약점 monster로 언급된 횟수)의 dictionary를 만들기 위해, python 내 collections.Counter 를 사용.

(d) (c)에서 만든 몬스터: 언급된 수 pair 의 Counter (dict) 를 이용하여, 바람이가 갖고 있는 몬스터들 중 공통으로 약점이 되는 몬스터의 목록을 추출하여 intersect_pokemon (list) 변수에 저장함

(e) (d)에서 추출한 몬스터의 타입 목록과 해당 몬스터들이 출현하는 시간을 Regular Expression (단순하게 [20:00~23:59 내 출현하는 몬스터를 찾기 위해서 ^2[0-3]: 으로 처리

(f) (e)에서 나온 최종 결과를 base code를 이용하여 출력

```
database@database-vm:~/practice/mongopractice$ python3 pokemon.py 1
{ id:142, name:Aerodactyl, spawn_time:23:40, type:['Rock', 'Flying'] }
{ id:149, name:Dragonite, spawn_time:23:38, type:['Dragon', 'Flying'] }
{ id:126, name:Magmar, spawn_time:20:36, type:['Fire'] }
```

(2) problem_2

(a) 최종 진화한 몬스터들의 목록을 추출하기 위해, next_evolution은 없고, pre_evolution은 있는(진화를 한번 이상한 몬스터)를 추출,이때 추출한 몬스터의 목록을 loop돌면서 바로 출력할 것이기 때문에 출력 조건인 id 열로 오름차순 정렬하여 추출

(b) (a)에서 추출한 몬스터들을 한번씩 loop돌면서 진화 이전의 몬스터들의 목록을 추출

(c) (b)에서 추출된 이전 몬스터들이 진화하는데 필요한 캔디 종류와 갯수를 임시 변수 count 누적 합함.

(d) (b)-(c)과정을 완료후 해당 최종 몬스터의 이름 / 진화에 필요한 캔디 수 / 캔디 이름에 대해서 출력

```
database@database-vm:~/practice/mongopractice$ python3 pokemon.py 2
Venusaur => Bulbasaur Candy: 125
Charizard => Charmander Candy: 125
Blastoise => Squirtle Candy: 125
Butterfree => Caterpie Candy: 62
Beedrill => Weedle Candy: 62
Pidgeot => Pidgey Candy: 62
Raticate => Rattata Candy: 25
Fearow => Spearow Candy: 50
Arbok => Ekans Candy: 50
Raichu => Pikachu Candy: 50
Sandslash => Sandshrew Candy: 50
Nidoqueen => Nidoran  (Female) Candy: 125
Nidoking => Nidoran  (Male) Candy: 125
Clefable => Clefairy Candy: 50
Ninetales => Vulpix Candy: 50
Wigglytuff => Jigglypuff Candy: 50
Golbat => Zubat Candy: 50
```

3.Email Search Engine

- 최초 script 실행 시. enron(email collection)내에 subject/text 를 이용하여 만든 text index가 있는지 확인, 만약 없다면 생성, 있다면, 생성하는 구문 없이 진행
- 이하 3개의 함수를 선언하여 검색어 처리 및 출력

(1) searchfromemail : argv로 부터 입력받은 필터 조건을 토대로 email collection 을 검색하는 wrapper function

(a) 매개변수 : console 에서 실행 시 추가 매개변수로 입력한 필터조건

(b) (2)에서 별도 선언한 cratequeryandsortcondtion함수를 통해서 매개변수로 넘겨받은 조건에 대해서 query 조건 및 sort조건을 dictionary 로 형태로 생성

(c) 만약 script실행 시에 sort 조건이 있었다면, find를 통해서 얻은 결과를 재정렬

(d) 주어진 base code를 이용하여 결과 출력

(2) createqueryandsortcondition : argv에서 입력받은 필터조건을 query 조건 string 으로 변환해주는 function

(a) query문을 생성할 때 고려해야할 부분은 '/' , ',' , ' : (keyword 포함 여부 from,to,sort,not)

(b) 매개변수로 넘겨받은 필터조건을 split 함수를 통해 1차적으로 '/'로 분리

(c) 필터조건에 ':' 있는지 체크

(i) 없다면 : 그냥 단순히 본문 및 제목에서 검색하는 것이므로, $text,$search 를 이용한 dictionary query string 생성

(ii) 있다면 :

1) from : key 를 'sender'로 생성하여 (3) 함수를 이용하여 복수의 발신자를 검색하는지 단수의 사람이 검색하는지에 맞춰 value 저장

2) to : key를 'to'로 생성하여 (3)함수 이용하여 value 값 생성

3) not : 기존에 $text,$search 를 key가 최종 query (dictionary) 에 있는지 확인 후 만약 없다면 새로 생성해주고, 있다면 '$search' 부분에 내용만 추가

(3) convertedkeywordvlaue : multiple value 로 검색할 시 $in : [values] 형태로 string이 필요하므로, 해당 처리를 위한 function

```
database@database-vm:~/practice/mongopractice$ python3 engine.py 'Social / not: Network / sort: date'
sender                  subject                 text                                    date
councilofthemaer        REMINDER: Franci        REMINDER: FRANCISCO BARRIO TERRAZAS,    2001-06-01 14:03
councilofthemaer        Francisco Gros -        FRANCISCO GROSPresident of the Natio    2001-06-01 13:20
rob.bradley@enro        Philanthropy Ide        Ken:The attached one pager describes    2000-10-05 01:51
jngoodman@ncpa.o        i2Technologies/S        Ken:Yesterday, we received a copy of    2000-09-12 01:55
```

```
database@database-vm:~/practice/mongopractice$ python3 engine.py 'from: robyn@layfam.com'
sender                  subject                 text                                    date
robyn@layfam.com          Nicholas B-day        Hello Mamie and Papi, Invitations ar    2002-01-08 23:48
```

```
database@database-vm:~/practice/mongopractice$ python3 engine.py 'to: cindy.olson@enron.com, greg.whalley@enron.com / Please / not: At
tached, previously'
sender                  subject                 text                                    date
dorothy.barnes@e        Revised Speech L        Please call if you note any changes     1999-10-14 06:33
joannie.williams        Management Commi        Please call if you have any question    2001-10-15 06:19
ryan.seleznov@en        Employee Concern        I wanted to assure all of you that t    2001-10-19 12:55
ken.rice@enron.c        Urgently Need Yo        I am compiling two "Top Ten Lists" t    2001-05-30 09:52
mary.clark@enron        All-employee mee        Ken, Greg and Mark:Our next all-empl    2001-09-26 16:59
jeffrey.mcmahon@        RE: E-mail for a        Fine with me.   Let's get this out A    2001-12-07 11:48
p..dupre@enron.c        Need a Bright Te        Ken/Greg:I wish you both the best as    2001-12-03 10:24
```