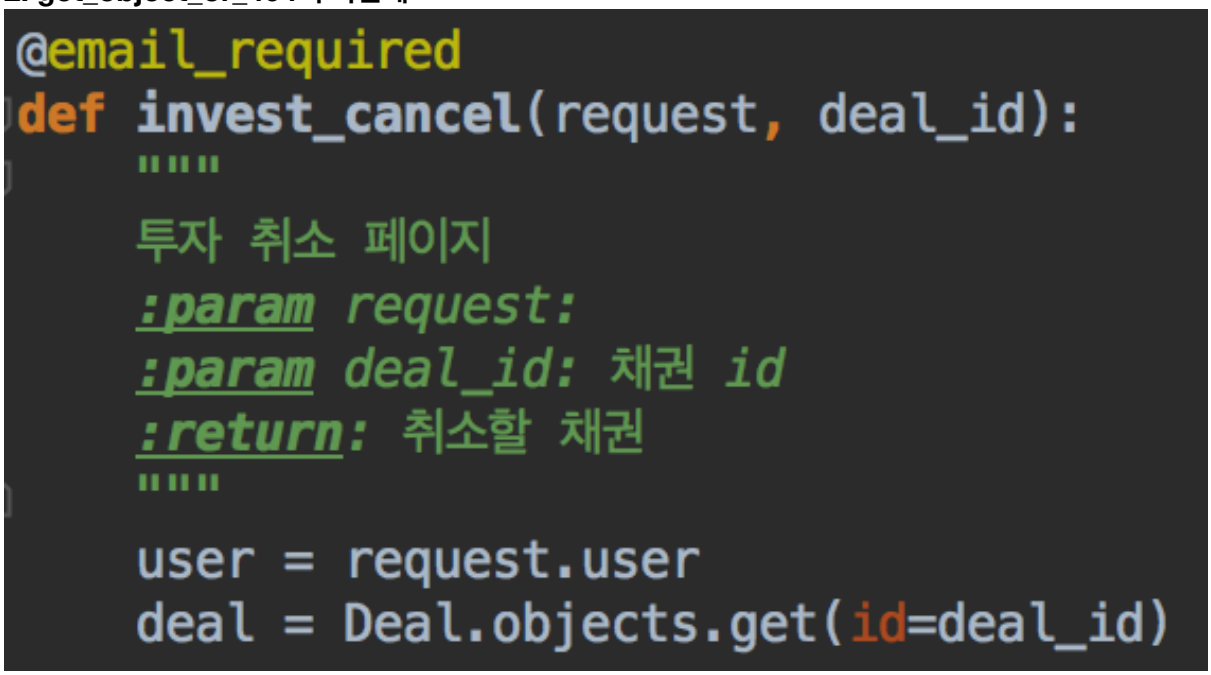
TSD 7장 심화주제

1. ObjectDoesNotExist 를 쓸때와 Object.DoesNotExist 쓸때의 차이는 무엇일까?

```
from django.core.exceptions import ObjectDoesNotExist
try:
    e = Entry.objects.get(id=3)
    b = Blog.objects.get(id=1)
except ObjectDoesNotExist:
    print("Either the entry or blog doesn't exist.")
```

2. get_object_or_404의 나쁜예



3. Lazy evaluation 이라는것이 무엇일까?
현주님이 설명해 주시겠지

4. pdb
PyCharm 쓰자

5. Iterator()와 all()의 차이점

In [4]:
```
@print_query_count
def count_user():
    count = 0
    for user in User.objects.iterator():
        user.name
        count += 1
    return count
count_user()
```

```
1개의 raw 쿼리가 실행 되었습니다.
[{'sql': 'DECLARE "_django_curs_140735981622208_1" NO SCROLL CURSOR WITH HOLD FOR SELECT "eight_user"."id", "eight_us
er"."password", "eight_user"."last_login", "eight_user"."is_superuser", "eight_user"."last_activity", "eight_user"."i
```

In [5]:
```
@print_query_count
def count_user():
    count = 0
    for user in User.objects.all():
        count += 1
    return count
count_user()
```

```
1개의 raw 쿼리가 실행 되었습니다.
[{'sql': 'SELECT "eight_user"."id", "eight_user"."password", "eight_user"."last_login", "eight_user"."is_superuser",
 "eight_user"."last_activity", "eight_user"."is_staff", "eight_user"."is_beta_tester", "eight_user"."ada_account_link
ing_token", "eight_user"."invest_amount", "eight_user"."email", "eight_user"."is_email_confirm", "eight_user"."userna
```

- Caching 을 하지 않는다.
- 단방향만 고려하면 된다.
- Client side cursor, server side cursor

6. SQL Explain과 Index

- Index 라는것은 무엇인가?
  - Index는 어떻게 동작할까?
  - Binary tree
  - B Tree
    - https://www.youtube.com/watch?v=coRJrcIYbF4
  - Bitmap Index
- Explain
  - **EXPLAIN ANALYSE select * from eight_user;**
  -
```
QUERY PLAN
Seq Scan on eight_user  (cost=0.00..3269.68 rows=54968 width=600) (actual time=0.011..36.381 rows=54974 loops=1)
Planning time: 0.082 ms
Execution time: 45.006 ms
```
  - **EXPLAIN ANALYSE select * from** eight_user **where** id=1;
  -
```
QUERY PLAN
Index Scan using eight_user_pkey on eight_user  (cost=0.29..4.31 rows=1 width=600) (actual time=0.014..0.014 rows=1 loops=1)
  Index Cond: (id = 1)
Planning time: 0.092 ms
Execution time: 0.045 ms
```
  - Debug toolbar 에서
  -

SQL explained

Executed SQL

SELECT ··· FROM "eight_deal" WHERE "eight_deal"."index" = 3539

시각

3.0319690704345703 ms

Database

default

QUERY PLAN
Index Scan using eight_deal_index_e6b7cfa08ef2aa6_uniq on eight_deal (cost=0.28..8.30 rows=1 width=1029) (actual time=0.014..0.015 rows=1 loops=1)
Index Cond: (index = 3539)
Planning time: 0.966 ms
Execution time: 0.077 ms

7. Transaction

- ACID
  - Atomic(원자성)은 트랜잭션과 관련된 작업들이 모두 수행되었는지 아니면 모두 실행이 안되었는지를 보장하는 능력이다. 자금 이체는 성공할 수도 실패할 수도 있지만 원자성은 중간 단계까지 실행되고 실패하는 일은 없도록 하는 것이다.
  - Consistent(일관성)은 트랜잭션이 실행을 성공적으로 완료하면 언제나 일관성 있는 데이터베이스 상태로 유지하는 것을 의미한다. 무결성 제약이 모든 계좌는 잔고가 있어야 한다면 이를 위반하는 트랜잭션은 중단된다.
  - Isolated(고립성)은 트랜잭션을 수행 시 다른 트랜잭션의 연산 작업이 끼어들지 못하도록 보장하는 것을 의미한다. 이것은 트랜잭션 밖에 있는 어떤 연산도 중간 단계의 데이터를 볼 수 없음을 의미한다. 은행 관리자는 이체 작업을 하는 도중에 쿼리를 실행하더라도 특정 계좌간 이체하는 양 쪽을 볼 수 없다. 공식적으로 고립성은 트랜잭션 실행내역은 연속적이어야 함을 의미한다.
  - Durable(영속성)은 성공적으로 수행된 트랜잭션은 영원히 반영되어야 함을 의미한다. 시스템 문제, DB 일관성 체크 등을 하더라도 유지되어야 함을 의미한다. 전형적으로 모드 트랜잭션은 로그로 남고 시스템 장애 발생 전 상태로 되돌릴 수 있다. 트랜잭션은 로그에 모든 것이 저장된 후에만 commit 상태로 간주될 수 있다.

8. Reference

- https://wiki.postgresql.org/images/4/45/Explaining_EXPLAIN.pdf
- https://www.postgresql.org/docs/current/static/using-explain.html
- https://msdn.microsoft.com/en-us/library/aa266531(v=vs.60).aspx