Queries: retrieve related objects

The models we declared contain relationships. Readers write multiple reviews and have multiple annotations. Similarly, books have multiple reviews and multiple annotations. In this exercise we will see how we can fetch all the reviews made by a reader, and to fetch the author of a review.

Fetching many objects

We fetch related objects of some object by accessing its attribute defined with `.relationship()`. For example, to fetch all reviews of a reader with `id = 123` we do the following:

```
reader = Reader.query.get(123)
reviews_123 = reader.reviews.all()
```

Note that `reviews` attribute was defined as a column in the model for `Reader` using the `.relationship()` method (see **app.py** to remind yourself).

Fetching one object

For `Review` object we can fetch its authoring `Reader` through the `backref` field specified in `Reader`'s `.relationship()` attribute. For example, to fetch the author of review `id = 111` we do the following:

```
review = Review.query.get(111)
reviewer_111 = review.reviewer
```

You can modify the examples above so not to have temporary variables (`reader` and `review_1`) by chaining the operations:

```
reviews_123 = Reader.query.get(123).reviews.all()
```

and

```
reviewer_111 = Review.query.get(111).reviewer
```

Notice the subtle difference between the examples above. The first needs `.all()` because one reader can have many reviews. In the second example, we do not use `.all()` since each review is associated with only one reader. That is our one-to-many relationship.

```
from app import db, Book, Reader, Review, Annotation

#Fetching 'many' objects
reader = Reader.query.get(123) #fetch a reader with id = 123
reviews_123 = reader.reviews.all() #fetch all reviews made by reader wiith
id = 123
[print(review.id) for review in reviews_123]
#check the image on the right - Ann Adams (id = 123, wrote reviews with ids
111 and 113)
```

```
#fetching 'one' object
review = Review.query.get(111) #fetch a review with id = 111
reviewer_111 = review.reviewer #get the reviewer for review with id = 111.
There's only one!
print("The author of [", review, "] is", reviewer_111)

#By chaining we avoid using temporary variables
reviews_123 = Reader.query.get(123).reviews.all() #same result as line 5
reviewer_111 = Review.query.get(111).reviewer #same result as line 10

print("\nCheckpoint 1: fetch all the reviews made for a book with id = 13.")
book_13 = Book.query.get(13).reviews.all()
[print(book.id) for book in book_13]

print("\nCheckpoint 2: fetch all the annotations made for a book with id =
19.")
book_19_an = Book.query.get(19).annotations.all()
[print(annotation.id) for annotation in book_19_an]

print("\nCheckpoint 3: fetch the reader who owns the annotation with `id =
331.`")
author_331 = Annotation.query.get(331).reviewer_id
print(author_331)
```