* 8:26 AM

“latest\_reviews”: Review.objects.order\_by(“-id”)[0:3]

s\_jancy\_joseph8:27 AM

i did this for latest row

* {% for post in posts.all reversed %}
* in jinja
* to get latest
* and kept getting errors, so i had to do it in views instead

s\_jancy\_joseph8:30 AM

mmm….

* i heard in stack overlflow taht set in jinja doesnt work well
* but your seems less hassle

s\_sotheary\_ouk8:30 AM

for dojo reads, you had to say how many reviews a user wrote. So i had to count in the [views.py](http://views.py/) and send the count number through context

this\_user = User.objects.get(id=user\_id)  
count = 0  
for i in this\_[user.book](http://user.book/)\_reviews.all():  
count +=1  
context = {  
“this\_user”: this\_user,  
“count”: count  
}

533E2b75

1. What is the flow of information in a typical request, from when we type an address on our browser, to when we receive the response in our browser?

Client makes a request and the server will return a response(html,css and js).A request is simply is simply client making a function call and the response is simply whatever was returned from the function.Users can type an address in the address bar,click a link ,refresh a page,submit a form etc…how users can interact with the the internet.

www.facebook.com Tells us which server we want to use

[www.facebook.com/me](http://www.facebook.com/me) ---> routing

we can make different routes to do different functions..we write the routes in urls.py but define the fucntions in views.py.

A couple of important things to notice here:

1. *Every function's first argument will be the****request****object.*
2. We don't distinguish in our routes anywhere between GET vs POST requests. This will be done *within* a given function.
3. If we are returning a string, we cannot simply return a string, but must send the string via HttpResponse (which must be imported. We'll be returning rendered templates again soon enough!)

Here's a visual of how routes get resolved in a Django project:

A screenshot of a cell phone

Description automatically generated

1. What are MVC and OOP? Why would we use each?

**MTV (or MVC)**

Developers who design frameworks have to make decisions about organizing code. One of the most popular patterns for organizing code is one known as MVC: Model-View-Controller. Django's pattern, MTV: Model-Template-View, is very similar. Here's a basic breakdown of responsibilities:

| **Django Component** | **MVC Component** | **Purpose** |
| --- | --- | --- |
| **Model** | **Model** | * May build database tables * Handles logic that relies on data * Interfaces with the database |
| **Template** | **View** | * HTML page that gets served to the client * May contain some logic to be handled by a template engine |
| **View** | **Controller** | * Receives incoming requests * Minimal logic * Calls on models to aggregate/process data * Determines appropriate response |

Object-oriented programming (**OOP**) is a way of writing computer programs which is using the idea of "objects" to represent data and methods. Usually, computer programs were just a list of instructions to the computer, telling the computer to do certain things in a certain way, which is called procedural programming.

The four principles of object-oriented programming are encapsulation, **abstraction**,**inheritance**, and polymorphism. These words may sound scary for a junior developer. And the complex, excessively long explanations in Wikipedia sometimes double the confusion.Jun 27, 2018

1. What is jQuery and why do we use it?

## What is jQuery?

jQuery is a JavaScript library. What this means for us is that it allows every browser to read JavaScript code. In the same manner, browser compatibility is a big issue when working on the front-end. This means that the jQuery code you write is suited for use by all browsers without change. JavaScript is interpreted by your browser, and unfortunately, Google Chrome interprets JavaScript differently than Safari, Mozilla and Internet Explorer do.  We don't want to have to write different versions of our code for different browsers, so cross-browser compatibility is important. All of the jQuery's functions work the same way regardless of which browser the user is running.

JQuery also converts what would have been a long block of code into just a few lines. As a developer, you should always practice the DRY method (Don't Repeat Yourself) which a group of developers discovered when they wrote the same lines of code over and over again (and it was a good amount of code too!). So, why not cut down many lines of code by making a library, and also by simplifying it for everyone? It sounds like a great idea!

1. Consider a database with the following tables: users, friendships, where users includes id, name, and email, and friendships includes id, user\_id, friend\_id; If our friendships table includes a single association to confirm a friendship between two users, what is the QUERY to pull all of your non-friends?

Friend.objects.Exclude(friend = User.obejcts.get(id = user\_id).friend)

What are some ways to make your website uniform across multiple browsers?  
jQuery code you write is suited for use by all browsers without change. All of the jQuery's functions work the same way regardless of which browser the user is running.

**Validate your code**

Run your HTML and CSS through validators before uploading your site. Valid code might not make your page magically work in all browsers, but invalid code can produce all sorts of unpredictable and hard-to-debug problems.

include a DOCTYPE to avoid a browser switches to quirks mode

## Use CSS reset rules

you’ve probably lost count of the number of times you’ve had to add margin: 0; padding: 0; to a CSS rule to make it consistent across all browsers.

## Test in as many browsers as possible

1. What are the differences between submitting a form via method="post" vs method="get"?

request.POST to access any form data that is submitted. (If the form is a GET request, that data can be accessed with request.GET.)

1. What are the advantages/disadvantages of sending data to the server in the url vs making a post request?
2. Why should we never render a page on a post request?
3. You notice that when you click submit on a form, your app breaks. Describe how you would approach debugging this problem.
4. What are a couple of security threats and how do we defend against them?
5. What is the difference between session, post, and cookies?
6. Explain why and when we use session.
7. What is an ORM and why do we use it? What are its advantages and disadvantages?
8. In your models, you may make a class User with the following code: class User(models.Model):. Why do we have models.Model inside the parentheses?
9. **Why models.Model?**
10. First, notice we are *inheriting* from the models.Model base class. If you didn't have a chance to practice inheritance back in the OOP chapters, ***inheritance*** is an important OOP principle that allows us to write code in one class (parent) and then allow other classes (children) to *inherit* that same code without having to re-write it in the child classes.
11. Practically speaking, this means that, even though we don't see additional code, Django's Model class provides a lot right out of the box. There's no way the Django developers could have anticipated all the different classes we as developers might create, but what they could anticipate was that we'd need classes, and that these classes would need to be created in our database, and they'd each need a primary key field. With that in mind, they created one generic parent Model class that contains these fields and functionalities.
12. You'll notice, for example, that we do not need to type an id field into any of our classes--Django automatically adds a field called "id" to every class inheriting from models.Model and sets it to be an auto-incremented field. We also don't have to write a separate \_\_init\_\_ method for each class. Very shortly, we'll also see the models come pre-loaded with all the CRUD functionality so we aren't required to write out all the SQL statements.
13. In your models, you may make a class UserManager with the following code: class UserManager(models.Manager):. Why do we have models.Manager inside the parentheses?
14. What is self and why does it appear in methods that we define in a class?
15. What is the \_\_init\_\_ method in a class?
16. Name 2 HTTP verbs and when would we use each.
17. What is RegEx and what do we use it for?
18. On large web applications, what are the benefits of using a framework?
19. What does a templating engine do for us?
20. **Jinja** is a web **template** engine for the Python programming language and is licensed under a BSD License created by Armin Ronacher. It is similar to the **Django template**engine but provides Python-like expressions while ensuring that the **templates** are evaluated in a sandbox.
21. What are 3 different type of database table relationships?
22. Why do we use routes and how do they work?
23. What is a virtual environment and why do we use it?
24. Explain what this code does: from flask import Flask
25. What is the importance of normalizing your database?
26. What is an API?
27. What is AJAX and why do we use it?
28. What is the difference between an HTTP request/response and an AJAX request/response?
29. What is the difference between client-side and server-side validations, and when do we use either or both?
30. What are the major differences between Flask and Django?
31. What are the differences between tuples, lists, and dictionaries?