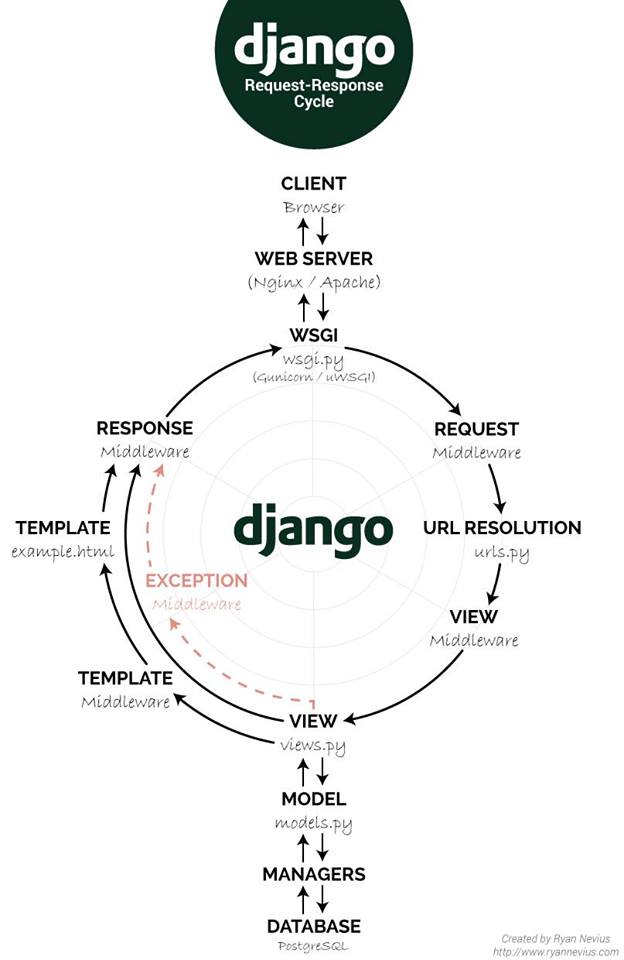
1. How django works (work flow)



1. What is \_\_init\_\_.py inside application folder?

The \_\_init\_\_.py file is actually Python’s convention for determining modules. When you try to run import library or from library import x, Python will search all folders in the Python Path. If a folder has an \_\_init\_\_.py file, Python will also search inside of that folder, otherwise it is ignored. So if we want to be able to import code from our app we need \_\_init\_\_.py.

1. What is the use of manage.py

It puts your project’s package on sys.path.

It sets the DJANGO\_SETTINGS\_MODULE environment variable so that it points to your project’s settings.py file.

1. How you will add extra function/feature in admin part?

The admin part of Django is much like any other, you can make modifications to the admin.py file by overriding existing behaviour.

1. Have you customized admin style?
2. What are all middleware you used and what’s the purpose of it?

Whenever you start a project there are several middleware that are configured, which include

SecurityMW, SessionMW, CommonMW, CSRFMW, AuthMW, MessageMW and XFrameOptionMW.

I have used a customized login required MW and authentication MW to authenticate user with email rather than username.

1. How you used session? How can we handle session expire time?

By default, Django has sessions enabled with SessionMW and it stores the sessions in your database as it is database backed session. You can chose to store session data in cache or file system as well.

You can set expiration time with set\_expiry method in views which takes a number as an argument corresponding to seconds. Also, to get the required behaviour you need to set [**SESSION\_SAVE\_EVERY\_REQUEST**](https://docs.djangoproject.com/en/dev/ref/settings/#std:setting-SESSION_SAVE_EVERY_REQUEST) to true. Else, you will have to vall the set\_expiry method in all the views.

1. How you will write django complex query?

By using from **django.db.models import Q**

1. Django group\_by query

Use annotate, it reduces the number of queries being executed on the database and translates to group by on database. You can use all aggregate functions within the annotate function.

1. What is signals and how to use it? What are two important parameter in signals?

Signals are events sent whenever there is a modification made to data in database. Signals allow certain senders to notify a set of receivers that some action has taken place. Signals are called no below actions:

**model's save() is called, model’s delete()is called, ManyToManyField on a model is changed or Django starts or finishes an HTTP request**

An example would be to make changes to a profile associated with a user whenever a user is saved.

The two parameters are: A signal to listen for, which is a required argument and the optional sender argument to specify which model class to listen into for the signal

1. Have you written template tags? What’s the use of it?

No, the template tags are used to address the presentation logic needs of your application.

1. Form inheritance

It is much similar to class inheritance. It comes handy when you want additional functionality in the sub classes of forms that you do not need in base classes or that you do not have in base classes.

1. Model inheritance

There are several ways of inheriting a model. The most used ones are: abstract base models, multi-table inheritance, proxy models, and multiple inheritance.

When there are multiple models that have same set of attributes, it is clever to group these attributes in an abstract class and inherit it.

When we talk about multi-table inheritance we are referring to having a top level table with all the fields that are core to that context. Then we are going to have other tables with the more specific fields. You select from the table you want, and then do a join on a relationship to get the parent table data

A proxy model is just another class that provides a different interface for the same underlying database model.

The main use case of multiple inheritance is mixins classes.

1. Limitation of multiple inheritance?

When using multiple inheritance, you need to create an Auto Field with primary key=True in the base class, else the subclass will raise a key error.  If multiple parents contain a [Meta](https://docs.djangoproject.com/en/2.1/topics/db/models/#meta-options) class, only the first one is going to be used, and all others will be ignored.

1. How to get all logged-in user list or count?

<https://stackoverflow.com/questions/2723052/how-to-get-the-list-of-the-authenticated-users>

1. What are all reusable applications you used in Django project?

* Django crispy forms
* Django filter
* Django widget tweaks

1. Save dynamic data without using DB/file.
2. How you are requesting Ajax in Django? How it works?
3. How you will access a particular block in child html?
4. How u are debugging application?
5. Logging module have u used?
6. What is message framework and explain it
7. What are modules in Django?
8. What is serialization and why we are using it?
9. How you are testing your application?
10. How you are deploying application with apache?
11. What’s the difference between mod\_python & mod\_wsgi
12. What are all new features in Django latest version
13. What’s major advantage in Django 1.2
14. which function won’t run in Django 1.2 which runs in Django 1.0
15. If Django powered application is high traffic site then how you will reduce the traffic?
16. How to modify model without affecting data?
17. What are things we have to consider while designing models
18. How to make your own middleware,
19. How does signals works,
20. How does template tags, filter work? How do you make your own?
21. How do you monitor and optimize the Django ORM queries?
22. What are the basic steps you will take to secure your default django product upon deployment?
23. Generators and Iterators.
24. OOPS concepts on Python
25. decorators in Python and Django
26. Model Forms and ManyToManyRelations.
27. MVC Architecture and complex queries of Django Querysets.
28. What are middlewares in Django and why are they used?

Django uses middlewares to provide additional functions and security. However, a really simple Django application can be designed without using any middleware.

At its core, Django takes an HTTP request (sent to the Web server) and turns it into an HTTP response (sent from the server back to the browser). This is usually done via Views and Templates. Middleware provides a way to consistently process the requests or the responses on a global level without having to customize every view. It can either work on the requests or on the response objects depending on where in the lifecycle it operates.

1. What are the differences between pre 1.10 and post 1.10 django versions for middleware?

These are the behavioral differences between using MIDDLEWARE and MIDDLEWARE\_CLASSES:

1. Under MIDDLEWARE\_CLASSES, every middleware will always have its process\_response method called, even if an earlier middleware short-circuited by returning a response from its process\_request method. Under MIDDLEWARE, middleware behaves more like an onion: the layers that a response goes through on the way out are the same layers that saw the request on the way in. If a middleware short-circuits, only that middleware and the ones before it in MIDDLEWARE will see the response.
2. Under MIDDLEWARE\_CLASSES, process\_exception is applied to exceptions raised from a middleware process\_request method. Under MIDDLEWARE, process\_exception applies only to exceptions raised from the view (or from the render method of a TemplateResponse). Exceptions raised from a middleware are converted to the appropriate HTTP response and then passed to the next middleware.
3. Under MIDDLEWARE\_CLASSES, if a process\_response method raises an exception, the process\_response methods of all earlier middleware are skipped and a 500 Internal Server Error HTTP response is always returned (even if the exception raised was e.g. an Http404). Under MIDDLEWARE, an exception raised from a middleware will immediately be converted to the appropriate HTTP response, and then the next middleware in line will see that response. Middleware are never skipped due to a middleware raising an exception
4. How to make your own middleware?
5. How do signals work?
6. How does template tags, filter work? How do you make your own?
7. How do you monitor and optimize the Django ORM queries?
8. What are the basic steps you will take to secure your default Django product upon deployment?
9. Explain Model Forms and ManyToManyRelations
10. What is role of ALLOWED\_HOSTs in Django?
11. What is CSRF? How it's Important in Django based application?
12. How the Templating system is working in Django?
13. How to query as GROUP BY in Django?

Using annotate.

1. Difference between aggregate and annotate?

Aggregate works on entire query set while annotate works on mentioned columns. Drawback of using aggregate is that you need to loop through the entire queryset to aggregate on each field which is database heavy.

1. How Django complex query will be return (Django Aggregation)?
2. How to fetch limited data from Django models like SQL Limit?

Entry.objects.all()[:5]

1. What's the difference between select\_related and prefetch\_related?
2. Difference between Function Based views and Class based views? Which one you prefer? Why?
3. Tell about few packages which you used in your Django career?
4. Difference between Form and Model Form? Can we add custom styles in forms fields?
5. User authentication Django? Where you can find the built-in User (importing)?
6. What is Model, views?
7. What does of Django field class types do?
8. Have you used Django a content management system (CMS)?
9. How to Handle Load Management in Django?
10. Is Django supporting Session? How do we set a variable in session?
11. How to improve the performance of the Django application?
12. What is Memcache? What is redis server?
13. Why logging is important in django application? In which area it will important?
14. What is Scheduling? Which packages you are using for this same?
15. What is caching?
16. In which django version you are using for development?
17. Have you heard about the South? What it will do?
18. How do we write the test cases?
19. How can we write the Restful API's?
20. REST API package? (django-tastypy and DRF), please study the diff, advantages, and disadvantages?
21. When we submit a form without CSRF token the server will throw 403 error message, but when we are posting request from (drf or tastypy)
22. It got accepted without any error, why? How?
23. What is Unicode?
24. Internationalization and localization?
25. Which all are the Common Web application tools provided by django?
26. What is Model Manager?
27. How do we add extra model field in django models?
28. What is Meta class method?
29. Some times while running your project you will get error like "the port is already in use" how will you resolve the issue? [Hint: Port Killing]

In Linux,

netstat -ntlp

kill -9 PID

On Mac,

sudo lsof -i tcp:8000

1. List out the inheritance styles in Django?
2. What is diff between Nosql and Sql databases?
3. Which ORM is using for the mongo dB database integration in Django? How? Advantage?
4. Where we are using RDBMS and Nosql databases?
5. Difference between Oauth and Oauth2?
6. How does authentication work in django?
7. How do we write custom authentication backend with Django?
8. What is token authentication?
9. What is authentication and authorization?
10. How to handle the high traffic in django based application?
11. What are built-in management commands? How do we write custom management commands?
12. How you debug the application?
13. Have you tried django-channels, sockets?
14. How to implement the social media authentication in django based application?
15. What is Hashing? Difference between hashing and encryption?
16. How to deploy a django project?
17. How good are you in GAE hosting? How to host a django project in GAE?
18. How good are you in Git commands? Explain merge command and resolve conflicts.
19. What’s use of the apps.py?
20. What’s \_\_pycache\_\_? What is .pyc files will do?

Python Interview Questions

1. What is data type
2. list, tuple, dict diff
3. list sorting without using generic function
4. Can we use list as key in dict?

No, because lists are not hashable

1. Dict sorting both key and value sorting

Sorted(dict) sorts on key, to sort on value, use operator.itemgetter(1)

1. What is Listcomprehension and where you use it?

It is a syntactic construct that helps to create a list from an existing list with some conditions on the go.

1. What is iterator, generator?
2. How to get traceback? Explain
3. how to write class
4. what is \_\_init\_\_
5. what is self
6. what is constructor
7. what is decorator? How it works? How to write custom decorator?
8. What is \_\_init\_\_ and \_\_new
9. What is super function in python and where you will use
10. Have you used set ()? if so where (Note that Set() deprecated in python 2.6)
11. Single linked list?
12. Which version you are using?
13. Deprecation in python latest version or version which you are working
14. Memory management in Python?
15. What all the Advanced features added or removed from the latest version of python?
16. Built in data types in Python? Give some example?
17. Is it Python Call by reference or Call by value?
18. The difference between range and xrange?