PYTHON LIFE

Python | Django Interview Questions and Answers

1. What is Django?
Ans: Django is a high-level Python Web framework that encourages rapid development and clean, pragm atic design. Built by experienced developers, it takes care of much of the hassle of Web development, so you can focus on writing your app without needing to reinvent the wheel. It’s free and open source..

2. What does Django mean?
Ans: Django is named after Django Reinhardt, a gypsy jazz guitarist from the 1930s to early 1950s who is known as one of the best guitarists of all time.

3. Which architectural pattern does Django Follow?
Ans: Django follows Model-View Template (MVT) architectural pattern.

4. Explain the architecture of Django?
Ans: Django is based on MVT architecture. It contains the following layers:

Models: It describes the database schema and data structure.

Views: The view layer is a user interface. It controls what a user sees, the view retrieves data from approp riate models and execute any calculation made to the data and pass it to the template.

Templates: It determines how the user sees it. It describes how the data received from the views should b e changed or formatted for display on the page.

Controller: Controller is the heart of the system. It handles requests and responses, setting up database c onnections and loading add-ons. It specifies the Django framework and URL parsing.

5. Which foundation manages Django web framework?
Ans: Django web framework is managed and maintained by an independent and non-profit organization n amed Django Software Foundation (DSF).

6. Is Django stable?
Ans: Yes, Django is quite stable. Many companies like Disqus, Instagram, Pinterest, and Mozilla have be en using Django for many years.

7. What are the features available in Django web framework?
Ans: Features available in Django web framework are:

Admin Interface (CRUD)
Templating
Form handling
Internationalization
Session, user management, role-based permissions
Object-relational mapping (ORM)
Testing Framework
Fantastic Documentation

8. What are the advantages of using Django for web development?
Ans: It facilitates you to divide code modules into logical groups to make it flexible to change.
It provides auto-generated web admin to make website administration easy.
It provides pre-packaged API for common user tasks.
It provides template system to define HTML template for your web page to avoid code duplication.
It enables you to define what URL is for a given function.

It enables you to separate business logic from the HTML.

9. How to create a project in Django?
Ans: To start a project in Django, use the command $django-admin.py and then use the following comma
nd:

Project

init.py

manage.py

settings.py

urls.py

10. What are the inheritance styles in Django?
Ans: There are three possible inheritance styles in Django:

1. Abstract base classes: This style is used when you only want parent?s class to hold information that yo
u don't want to type out for each child model.

2. Multi-table Inheritance: This style is used if you are sub-classing an existing model and need each mod
el to have its own database table.

3. Proxy models: This style is used, if you only want to modify the Python level behavior of the model, with
out changing the model's fields.

11. How can you set up the database in Djanago?
Ans: To set up a database in Django, you can use the command edit mysite/setting.py , it is a normal pyth
on module with module level representing Django settings.

By default, Django uses SQLite database. It is easy for Django users because it doesn't require any other
type of installation. In the case of other database you have to the following keys in the DATABASE 'defau
lt' item to match your database connection settings.

Engines: you can change database by using 'django.db.backends.sqlite3' , 'django.db.backeneds.mysql',
'django.db.backends.postgresql_psycopg2', 'django.db.backends.oracle' and so on

Name: The name of your database. In the case if you are using SQLite as your database, in that case dat
abase will be a file on your computer, Name should be a full absolute path, including file name of that file.

Note: You have to add setting likes setting like Password, Host, User, etc. in your database, if you are not
choosing SQLite as your database.

12. What does the Django templates contain?
Ans: A template is a simple text file. It can create any text-based format like XML, CSV, HTML, etc. A tem
plate contains variables that get replaced with values when the template is evaluated and tags (%tag%) th
at controls the logic of the template.

13. Is Django a content management system (CMS)?
Ans: No, Django is not a CMS. Instead, it is a Web framework and a programming tool that makes you abl
e to build websites.

14. What is the use of session framework in Django?
Ans: The session framework facilitates you to store and retrieve arbitrary data on a per-site visitor basis. It
stores data on the server side and abstracts the receiving and sending of cookies. Session can be imple
mented through a piece of middleware.

15. How can you set up static files in Django?
Ans: There are three main things required to set up static files in Django:

1. Set STATIC_ROOT in settings.py

2. run manage.py collectsatic

3. set up a Static Files entry on the PythonAnywhere web tab

16. How to use file based sessions?
Ans: You have to set the SESSION_ENGINE settings to "django.contrib.sessions.backends.file" to use fil
e based session.

17. What is some typical usage of middlewares in Django?
Ans: Some usage of middlewares in Django is:

Session management,
Use authentication
Cross-site request forgery protection
Content Gzipping, etc.

18. What does of Django field class types do?
Ans: The Django field class types specify:

The database column type.
The default HTML widget to avail while rendering a form field.
The minimal validation requirements used in Django admin.
Automatic generated forms.

19. What is the command to start Django's built-in development server?
A. manage.py runserver
B. manage.py –start
C. manage.py run
D. manage.py startserver –dev
E. manage.py –run
Ans: A

20. Given a model named 'User' that contains a DateTime field named 'last_login', how do you query for u
sers that have never logged in?
A. User.objects.filter(last_login=Null)
B. User.objects.filter(last_login__null=True)
C. User.objects.filter(last_login__isnull=False)
D. User.objects.filter(last_login__isnull=True)
E. User.objects.filter(last_login=Never)
Ans: D

21. What does the Django command ‘manage.py shell‘ do?
A. Starts a command line in whatever $SHELL your environment uses.
B. Starts a Django command prompt with your Python environment pre-loaded.
C. Starts a Python command prompt with your Django environment pre-loaded.
D. Loads a special Pythonic version of the Bash shell.
E. Loads a Python command prompt you can use to sync your database schema remotely.
Ans: C

22. Assuming you’ve imported the proper Django model file, how do you add a ‘User’ model to the Django admin?
A. admin.register(Users)
B. admin.site(self, User)
C. user.site.register(Admin)
D. users.site.register(Admin)
E. admin.site.register(User)
Ans: E

23. What is the Django command to start a new app named ‘users’ in an existing project?
A. manage.py –newapp users
B. manage.py newapp users
C. manage.py –startapp users
D. manage.py startapp users
E. manage.py start users
Ans: D

24. What does a urls.py file do in Django?
A. This file contains site deployment data such as server names and ports.
B. It contains a site map of Django-approved URLs.
C. It contains URL matching patterns and their corresponding view methods.
D. You run this file when you get obscure 404 Not Found errors in your server logs.
E. This file provides an up to date list of how-to URLs for learning Django more easily.
Ans: C

25. What is the command to run Django’s development server on port 8080 on IP address 12.34.56.78?
A. manage.py –run 12.34.56.78 8080
B. manage.py –dev 12.34.56.78:8080
C. manage.py runserver 12.34.56.78:8000
D. manage.py run 12.34.56.78:8080
E. manage.py runserver 12.34.56.78:8080
Ans: E

26. Django is written using what programming language?
A. PHP
B. Ruby
C. Javascript
D. Java
E. Python
Ans: E

27. After you make a new ‘app’ in your existing Django project, how do you get Django to notice it?
A. No additional action is required, Django notices new apps automatically.
B. Run the ‘manage.py validate‘ command, and then start a new shell.

C. Run the ‘manage.py syncdb‘ command.
D. In settings.py, add the app to the PROJECT_APPS variable.
E. In settings.py, add the new app to the INSTALLED_APPS variable.
Ans: E
28. What is the purpose of settings.py?
A. To configure settings for the Django project
B. To configure settings for an app
C. To set the date and time on the server
D. To sync the database schema
Ans: A

29. How do you define a ‘name’ field in a Django model with a maximum length of 255 characters?
A. name = models.CharField(max_len=255)
B. model.CharField(max_length=255)
C. name = models.CharField(max_length=255)
D. model = CharField(max_length=255)
E. name = model.StringField(max_length=auto)
Ans: C

30. What is the definition of a good Django app?
A. A good Django app provides a small, specific piece of functionality that can be used in any number of Django projects.
B. A good Django app is a fully functioning website that has 100% test coverage.
C. A good Django app is highly customized and cannot be used in multiple projects.
Ans: A

31. What is the most easiest, fastest, and most stable deployment choice in most cases with Django?
A. FastCGI
B. mod_wsgi
C. SCGI
D. AJP
Ans: B

32. How do you exclude a specific field from a ModelForm?
A. Create a new Form, don’t use a ModelForm
B. Use the exclude parameter in the Meta class in your form
C. Set the field to hidden
D. You can not do this
Ans: B

33. Assuming you have a Django model named ‘User’, how do you define a foreign key field for this mode
l in another model?
A. model = new ForeignKey(User)
B. user = models.IntegerKey(User)
C. user = models.ForeignKey(User)
D. models.ForeignKey(self, User)
Ans: C

34. What preferred method do you add to a Django model to get a better string representation of the mod
el in the Django admin?
A. __unicode__
B. to_s(self)
C. __translate__
D. __utf_8__
Ans: A

36. What is Model Form used for?
A. To model an input form for a template
B. To specify rules for correct form when writing Django code
C. To define a form based on an existing model
Ans: C

37. What happens if MyObject.objects.get() is called with parameters that do not match an existing item in the database?
A. The Http404 exception is raised.
B. The DatabaseError exception is raised.
C. The MyObject.DoesNotExist exception is raised.
D. The object is created and returned.
Ans: C

38. A set of helpful applications to use within your Django projects is included in the official distribution. Th is module is called what?
A. django.extras
B. django.helpers
C. django.utilities
D. django.ponies
E. django.contrib
Ans: E

39. What is the correct syntax for including a class based view in a URLconf?
A. (r'^pattern/$', YourView.as_view()),
B. (r'^pattern/$', YourView.init()),
C. (r'^pattern/$', YourView),
D. (r'^pattern/$', YourView()),
Ans: A

40. What is the command to start a new Django project called 'myproject'?
A. django-admin.py startproject myproject
B. django-admin.py –start myproject
C. django.py startproject myproject
D. django.py –new myproject
E. django.py new myproject
Ans: A

41. How to make django timezone-aware?
A. In settings.py: USE_L10N=True
B. in views.py, import timezone
C. in views.py, import tz
D. in urls.py, import timezone
E. In settings.py: USE_TZ=True
Ans: E

42. In Django how would you retrieve all the 'User' records from a given database?
A. User.objects.all()
B. Users.objects.all()
C. User.all_records()
D. User.object.all()
E. User.objects
Ans: A

43. How can you define additional behavior and characteristics of a Django class?
A. def setUp():
B. class Meta:
C. class __init__:
D. def Meta():
E. def __init__():
Ans: B

44. What is the Django shortcut method to more easily render an html response?
A. render_to_html
B. render_to_response
C. response_render
D. render
Ans: B

45. What does the Django command ‘manage.py validate‘ do?
A. Checks for errors in your views.
B. Checks for errors in your templates.
C. Checks for errors in your controllers.
D. Checks for errors in your models.
E. Checks for errors in your settings.py file.
Ans: D

46. What is the correct way to include django’s admin urls? from django.contrib import admin’) from django.conf.urls import patterns, include, url urlpatterns = patterns(”, ______________________)
A. url(r’^admin/’, admin.as_view(), name=’admin),
B. url(r’^admin/’, include(admin)),
C. url(r’^admin/’, include(admin.site.urls)),
D. url(r’^admin/’, admin.urls),
E. admin.autodiscover()
Ans: C

47. Where is pre_save signal in Django
A. from django.db.models import pre_save
B. from django.db.models.signals import pre_save
C. There is no pre_save signal
D. from django.db.models.signal import pre_save
Ans: B

48. Given the Python data: mydata = [[0, ‘Fred’], [1, ‘Wilma’]] How do you access the data in a Django template?
A. {% for d in mydata %}

{% d.1 %}

{% endfor %}
B. {% for d in mydata -%}

{{ d.1 }}

{% end -%}
C. {% for d in mydata %}

{{ d.1 }}

{% endfor %}
D. {{ for d in mydata }}

{{ d[1] }}

{{ endfor }}
E. {% mydata.each |d| %}

{{ d.2 }}

{% end %}
Ans: C

49. What is the purpose of the STATIC_ROOT setting?
A. Defines the URL prefix where static files will be served from .
B. Defines the location where all static files will be copied by the 'collectstatic' management command, to be served by the production webserver.
C. A project's static assets should be stored here to be served by the development server.
D. Defines the location for serving user uploaded files.
Ans: B

50. How to create a DateTimeField named created and filled in only on the creation with the current time?

A. created = models.CreationTimeField()
B. created = models.DateTimeField(default=datetime.datetime.now())
C. created = models.DateTimeField(auto_now_add=True, auto_now=True)
D. created = models.DateTimeField(auto_now=True)
E. created = models.DateTimeField(auto_now_add=True)
Ans: E
51. What is the difference between a project and an app in Django?
Ans: In Django, a project is the entire application and an app is a module inside the project that deals with one specific requirement. E.g., if the entire project is an ecommerce site, then inside the project we will h ave several apps, such as the retail site app, the buyer site app, the shipment site app, etc.

52. What is Django Admin Interface?
Ans: Django comes with a fully customizable in-built admin interface, which lets us see and make change s to all the data in the database of registered apps and models. To use a database table with the admin in terface, we need to register the model in the admin.py file.

53. Explain Django's Request/Response Cycle.
Ans: In the Request/Response Cycle, first, a request is received by the Django server. Then, the server lo oks for a matching URL in the urlpatterns defined for the project. If no matching URL is found, then a resp onse with 404 status code is returned. If a URL matches, then the corresponding code in the view file ass ociated with the URL is executed to build and send a response.

54. What is a model in Django?
Ans: A model is a Python class in Django that is derived from the django.db.models.Model class. A model is used in Django to represent a table in a database. It is used to interact with and get results from the da tabase tables of our application.

55. What are migrations in Django?
Ans: A migration in Django is a Python file that contains changes we make to our models so that they can be converted into a database schema in our DBMS. So, instead of manually making changes to our data

base schema by writing queries in our DBMS shell, we can just make changes to our models. Then, we c an use Django to generate migrations from those model changes and run those migrations to make chan ges to our database schema.

56. What are views in Django?
Ans: A view in Django is a class and/or a function that receives a request and returns a response. A view i s usually associated with urlpatterns, and the logic encapsulated in a view is run when a request to the U RL associated with it is run. A view, among other things, gets data from the database using models, pass es that data to the templates, and sends back the rendered template to the user as an HttpResponse.

57. What is the use of the include function in the urls.py file in Django?
Ans: As in Django there can be many apps, each app may have some URLs that it responds to. Rather th an registering all URLs for all apps in a single urls.py file, each app maintains its own urls.py file, and in th e project’s urls.py file we use each individual urls.py file of each app by using the include function.

58. Why is Django called a loosely coupled framework?
Ans: Django is called a loosely coupled framework because of its MVT architecture, which is a variant of t he MVC architecture. It helps in separating the server code from the client-related code. Django’s models and views take care of the code that needs to be run on the server like getting records from database, etc ., and the templates are mostly HTML and CSS that just need data from models passed in by the views to render them. Since these components are independent of each other, Django is called a loosely coupled framework.

59.Mention the architecture of Django architecture?
Ans: Django architecture consists of

Models: It describes your database schema and your data structure
Views: It controls what a user sees, the view retrieves data from appropriate models and execute any calc ulation made to the data and pass it to the template
Templates: It determines how the user sees it. It describes how the data received from the views should b e changed or formatted for display on the page
Controller: The Django framework and URL parsing

60. Why Django should be used for web-development?
Ans:

It allows you to divide code modules into logical groups to make it flexible to change
To ease the website administration, it provides auto-generated web admin
It provides pre-packaged API for common user tasks
It gives you template system to define HTML template for your web page to avoid code duplication
It enables you to define what URL be for a given function
It enables you to separate business logic from the HTML
Everything is in python

61. Explain how you can create a project in Django?
Ans: To start a project in Django, you use command $ django-admin.py and then use the command
Project
init.py
manage.py
settings.py
urls.py

62. Explain how you can set up the Database in Django?
Ans: You can use the command edit mysite/setting.py , it is a normal python module with module level rep resenting Django settings.
Django uses SQLite by default; it is easy for Django users as such it won't require any other type of install ation. In the case your database choice is different that you have to the following keys in the DATABASE ' default' item to match your database connection settings

Engines: you can change database by using 'django.db.backends.sqlite3' , 'django.db.backeneds.mysql', 'django.db.backends.postgresql_psycopg2', 'django.db.backends.oracle' and so on
Name: The name of your database. In the case if you are using SQLite as your database, in that case dat abase will be a file on your computer, Name should be a full absolute path, including file name of that file.
If you are not choosing SQLite as your database then setting like Password, Host, User, etc. must be add ed.

63. Give an example how you can write a VIEW in Django?
Ans: Views are Django functions that take a request and return a response. To write a view in Django we take a simple example of "Guru99_home" which uses the template Guru99_home.html and uses the date -time module to tell us what the time is whenever the page is refreshed. The file we required to edit is call ed view.py, and it will be inside mysite/myapp/

Copy the below code into it and save the file

```
       from datatime import datetime
      from django.shortcuts import render
     def home (request):
return render(request, 'Guru99_home.html', {'right_now': datetime.utcnow()})
```

Once you have determined the VIEW, you can uncomment this line in urls.py

```
# url ( r '^$' , 'mysite.myapp.views.home' , name 'Guru99'),
```

The last step will reload your web app so that the changes are noticed by the web server.

64. Explain how you can setup static files in Django?
Ans: There are three main things required to set up static files in Django

Set STATIC_ROOT in settings.py
run manage.py collectsatic
set up a Static Files entry on the PythonAnywhere web tab

65. Mention what does the Django templates consists of?
Ans: The template is a simple text file. It can create any text-based format like XML, CSV, HTML, etc. A t emplate contains variables that get replaced with values when the template is evaluated and tags (% tag %) that controls the logic of the template.

66. Explain the use of session framework in Django?
Ans: In Django, the session framework enables you to store and retrieve arbitrary data on a per-site-visito r basis. It stores data on the server side and abstracts the receiving and sending of cookies. Session can be implemented through a piece of middleware.

67. Explain how you can use file based sessions?

Ans: To use file based session you have to set the SESSION_ENGINE settings to "django.contrib.session s.backends.file"

68. Explain the migration in Django and how you can do in SQL?
Ans: Migration in Django is to make changes to your models like deleting a model, adding a field, etc. into your database schema. There are several commands you use to interact with migrations.

Migrate
Makemigrations
Sqlmigrate
To do the migration in SQL, you have to print the SQL statement for resetting sequences for a given app name.
django-admin.py sqlsequencreset
Use this command to generate SQL that will fix cases where a sequence is out sync with its automatically incremented field data.

69. Mention what command line can be used to load data into Django?
Ans: To load data into Django you have to use the command line Django-admin.py loaddata. The comma nd line will searches the data and loads the contents of the named fixtures into the database.

70. Explain what does django-admin.py makemessages command is used for?
Ans: This command line executes over the entire source tree of the current directory and abstracts all the strings marked for translation. It makes a message file in the locale directory.

71. List out the inheritance styles in Django?
Ans: In Django, there is three possible inheritance styles

Abstract base classes: This style is used when you only wants parent's class to hold information that you don't want to type out for each child model
Multi-table Inheritance: This style is used If you are sub-classing an existing model and need each model to have its own database table
Proxy models: You can use this model, If you only want to modify the Python level behavior of the model, without changing the model's fields
72. Mention what does the Django field class types?
Ans: Field class types determines

The database column type
The default HTML widget to avail while rendering a form field
The minimal validation requirements used in Django admin and in automatically generated forms
73. What constitutes Django templates ?
Ans: Template can create formats like XML,HTML and CSV(which are text based formats). In general ter ms template is a simple text file. It is made up of variables that will later be replaced by values after the te mplate is evaluated and has tags which will control template's logic.

74. List some typical usage of middlewares in Django.
Ans: Some of the typical usage of middlewares in Django are: Session management, user authentication, cross-site request forgery protection, content Gzipping, etc.

75. How do you use views in Django?
Ans: Views will take request to return response. Let's write a view in Django : "example" using template example.html , using the date-time module to tell us exact time of reloading the page. Let's edit a file call ed view.py, and it will be inside randomsite/randomapp/
To do this save and copy following into a file:
Default

```
from datatime import datetime

from django.shortcuts import render

def home (request):

return render(request, ‘Guru99_home.html’, {‘right_now’: datetime.utcnow()})
```

Default

You have to determine the VIEW first, and then uncomment this line located in file urls.py

```
# url ( r ‘^$’ , ‘randomsite.randomapp.views.home’ , name ‘example’),
```

76. How do you make a Django app that is test driven and will display Fibonacci’s sequence?
This will reload the site making changes obvious.

Ans: Keep in mind that it should take an index number and output the sequence. Additionally, there should be a page that shows the most recent generated sequences.

Following is one of the solution for generating fibonacci series:
Default

```
def fib(n):

“Complexity: O(log(n))”

if n <= 0:

return 0

i = n – 1

(a, b) = (1, 0)

(c, d) = (0, 1)

while i > 0:

if i % 2:

(a, b) = (d * b + c * a,  d * (b + a) + c * b)

(c, d) = (c * c + d * d, d * (2 * c + d))

i = i / 2

return a + b
```

Default

Below is a model that would keep track of latest numbers:

```
from django.db import models

class Fibonacci(models.Model):
```

```
parameter = models.IntegerField(primary_key=True)

result = models.CharField(max_length=200)

time = models.DateTimeField()
```
DefaultFor view, you can simply use the following code:

```
from models import Fibonacci

def index(request):

result = None

if request.method==”POST”:

try:

n=int(request.POST.get(‘n’))

except:

return Http404

try:

result = Fibonacci.objects.get(pk=n)

result.time = datetime.now()

except DoesNotExist:

result = str(fib(n))

result = Fibonacci(n, result, datetime.now())

result.save()

return direct_to_template(request, ‘base.html’, {‘result’:result.result})
```
You could use models to get last ‘n’ entities.

77.What makes up Django architecture?
Ans: Django runs on MVC architecture. Following are the components that make up django architecture:

Models: Models elaborate back-end stuffs like database schema.(relationships)
Views: Views control what is to be shown to end-user.
Templates: Templates deal with formatting of view.
Controller: Takes entire control of Models.A MVC framework can be compared to a Cable TV with remote. A Television set is View(that interacts with end user), cable provider is model(that works in back-end) an d Controller is remote that controls which channel to select and display it through view.
78. What does session framework do in django framework ?
Ans: Session framework in django will store data on server side and interact with end-users. Session is g

enerally used with a middle-ware. It also helps in receiving and sending cookies for authentication of a us
er.

79.Can you create singleton object in python?If yes, how do you do it?
Ans: Yes, you can create singleton object. Here’s how you do it :

Default

12
3

4

5

class Singleton(object):def __new__(cls,*args,**kwargs):
if not hasattr(cls,’_inst’):

cls._inst = super(Singleton,cls).__new__(cls,*args,**kwargs)

return cls._inst

80. Mention caching strategies that you know in Django!
Ans: Few caching strategies that are available in Django are as follows:

File sytem caching
In-memory caching
Using Memcached
Database caching
81. What are inheritance type in Django?
Ans: There are 3 inheritance types in Django

Abstract base classes
Multi-table Inheritance
Proxy models

82. What do you think are limitation of Django Object relation mapping(ORM) ?
Ans: If the data is complex and consists of multiple joins using the SQL will be clearer.

If Performance is a concern for your, ORM aren’t your choice. Genrally. Object-relation-mapping are consi
dered good option to construct an optimized query, SQL has an upper hand when compared to ORM.

83. How to Start Django project with ‘Hello World!’? Just say hello world in django project.
Ans: There are 7 steps ahead to start Django project.

Step 1: Create project in terminal/shell

f2finterview:~$ django-admin.py startproject sampleproject

Step 2: Create application

```
f2finterview:~$ cd sampleproject/

f2finterview:~/sampleproject$ python manage.py startapp sampleapp
```

Step 3: Make template directory and index.html file

```
f2finterview:~/sampleproject$ mkdir templates

f2finterview:~/sampleproject$ cd templates/

f2finterview:~/sampleproject/templates$ touch index.html
```

Step 4: Configure initial configuration in settings.py

Add PROJECT_PATH and PROJECT_NAME

```
import os

PROJECT_PATH = os.path.dirname(os.path.abspath(__file__))

PROJECT_NAME = ‘sampleproject’
```

Add Template directories path

```
TEMPLATE_DIRS = (

os.path.join(PROJECT_PATH, ‘templates’),

)
```

Add Your app to INSTALLED_APPS

```
INSTALLED_APPS = (

‘sampleapp’,

)
```

Step 5: Urls configuration in urls.py

```
from django.conf.urls.defaults import patterns, include, url

urlpatterns = patterns(”,

url(r’^$’, ‘sampleproject.sampleapp.views.index’, name=’index’),

)
```

Step 6: Add index method in views.py

```
from django.shortcuts import render_to_response, get_object_or_404
from django.template import RequestContext

def index(request):

welcome_msg = ‘Hello World’

return render_to_response(‘index.html’,locals(),context_instance=RequestContext(request))
```

Step7: Add welcome_msg in index.html

```
<!DOCTYPE html>

<html>

<body>

<h1>My First Heading For Say…</h1>

<p>{{welcome_msg}}</p>

</body>

</html>
```

84. How to login with email instead of username in Django?
Ans: Use bellow sample method to login with email or username.

```
from django.conf import settings
from django.contrib.auth import authenticate, login, REDIRECT_FIELD_NAME
from django.shortcuts import render_to_response
from django.contrib.sites.models import Site
from django.template import Context, RequestContext
from django.views.decorators.cache import never_cache
from django.views.decorators.csrf import csrf_protect
@csrf_protect
@never_cache
def signin(request,redirect_field_name=REDIRECT_FIELD_NAME,authentication_form=LoginForm):
redirect_to = request.REQUEST.get(redirect_field_name, settings.LOGIN_REDIRECT_URL)
form = authentication_form()
current_site = Site.objects.get_current()
if request.method == “POST”:
pDict =request.POST.copy()
form = authentication_form(data=request.POST)
if form.is_valid():
username = form.cleaned_data[‘username’]
password = form.cleaned_data[‘password’]
try:
user = User.objects.get(email=username)
username = user.username
except User.DoesNotExist:
username = username
user = authenticate(username=username, password=password)
# Log the user in.
```

```
login(request, user)
return HttpResponseRedirect(redirect_to)
else:
form = authentication_form()
request.session.set_test_cookie()
if Site._meta.installed:
current_site = Site.objects.get_current()
else:
current_site = RequestSite(request)
return render_to_response(‘login.html’,locals(), context_instance=RequestContext(request))
```

85. How Django processes a request?
Ans: When a user requests a page from your Django-powered site, this is the algorithm the system follow s to determine which Python code to execute:
Django determines the root URLconf module to use. Ordinarily, this is the value of the ROOT_URLCONF setting, but if the incoming HttpRequest object has an attribute called urlconf (set by middleware request processing), its value will be used in place of the ROOT_URLCONF setting.
Django loads that Python module and looks for the variable urlpatterns. This should be a Python list, in th e format returned by the function django.conf.urls.patterns()
Django runs through each URL pattern, in order, and stops at the first one that matches the requested UR L.

Once one of the regexes matches, Django imports and calls the given view, which is a simple Python func tion (or a class based view). The view gets passed an HttpRequest as its first argument and any values c aptured in the regex as remaining arguments.

If no regex matches, or if an exception is raised during any point in this process, Django invokes an appro priate error-handling view.

86. How to filter latest record by date in Django?
Ans: Messages(models.Model):

```
    message_from = models.ForeignKey(User,related_name=”%(class)s_from”)
message_to = models.ForeignKey(User,related_name=”%(class)s_to”)
message=models.CharField(max_length=140,help_text=”Your message”)
created_on = models.DateTimeField(auto_now_add=True)
class Meta:
db_table = ‘messages’
```

Query:messages = Messages.objects.filter(message_to = user).order_by(‘-created_on’)[0]

Output:

```
message_from | message_to  | message              | created_on

——————|—————–|——————–|——————

Stephen      | Anto         | Hi, How are you? | 2012-10-09 14:27:48
```

87.How to filter data from Django models using python datetime?
Ans: Assume Bellow model for storing messages with timelines
class Message(models.Model):

```
from = models.ForeignKey(User,related_name = “%(class)s_from”)
to = models.ForeignKey(User, related_name = “%(class)s_to”)
msg = models.CharField(max_length=255)
rating = models.IntegerField(blank=’True’,default=0)
created_on = models.DateTimeField(auto_now_add=True)
updated_on = models.DateTimeField(auto_now=True)
```

Filter messages with specified Date and Time

```
today = date.today().strftime(‘%Y-%m-%d’)

yesterday = date.today() – timedelta(days=1)
yesterday = yesterday.strftime(‘%Y-%m-%d’)

this_month = date.today().strftime(‘%m’)
last_month = date.today() – timedelta(days=32)
last_month = last_month.strftime(‘%m’)
this_year = date.today().strftime(‘%Y’)

last_year = date.today() – timedelta(days=367)
last_year = last_year.strftime(‘%Y’)

today_msgs = Message.objects.filter(created_on__gte=today).count()
yesterday_msgs = Message.objects.filter(created_on__gte=yesterday).count()
this_month_msgs = Message.objects.filter(created_on__month=this_month,created_on__year=this_year)
.count()
last_month_msgs = Message.objects.filter(created_on__month=last_month,created_on__year=this_year)
.count()
this_year_msgs = Message.objects.filter(created_on__year=this_year).count()
last_year_msgs = Message.objects.filter(created_on__year=last_year).count()
```

88. What does Django mean?
Ans: Django is named after Django Reinhardt, a gypsy jazz guitarist from the 1930s to early 1950s who is known as one of the best guitarists of all time.

89. Which architectural pattern does Django Follow?
Ans: Django follows Model-View Controller (MVC) architectural pattern.

90. Is Django a high level web framework or low level framework?
Ans: Django is a high level Python’s web framework which was designed for rapid development and clean realistic design.

91. How would you pronounce Django?
Ans: Django is pronounced JANG-oh. Here D is silent.

92. How does Django work?
Ans: Django can be broken into many components:

Models.py file: This file defines your data model by extending your single line of code into full database ta bles and add a pre-built administration section to manage content.

Urls.py file: It uses regular expression to capture URL patterns for processing.

Views.py file: It is the main part of Django. The actual processing happens in view.
When a visitor lands on Django page, first Django checks the URLs pattern you have created and uses inf ormation to retrieve the view. After that view processes the request, querying your database if necessary, and passes the requested information to template.
After that the template renders the data in a layout you have created and displays the page.

93. Which foundation manages Django web framework?
Ans: Django web framework is managed and maintained by an independent and non-profit organization n amed Django Software Foundation (DSF).

94. Is Django stable?
Ans: Yes, Django is quite stable. Many companies like Disqus, Instagram, Pinterest, and Mozilla have be en using Django for many years.

95. What are the features available in Django web framework?
Ans: Features available in Django web framework are:

Admin Interface (CRUD)
Templating
Form handling
Internationalization
Session, user management, role-based permissions
Object-relational mapping (ORM)
Testing Framework
Fantastic Documentation

96. What are the advantages of using Django for web development?
Ans:

It facilitates you to divide code modules into logical groups to make it flexible to change.
It provides auto-generated web admin to make website administration easy.
It provides pre-packaged API for common user tasks.
It provides template system to define HTML template for your web page to avoid code duplication.
It enables you to define what URL is for a given function.
It enables you to separate business logic from the HTML.
97. How to create a project in Django?
Ans: To start a project in Django, use the command $django-admin.py and then use the following comma nd:
Project
init.py
manage.py
settings.py
urls.py

98. What are the inheritance styles in Django?
Ans: There are three possible inheritance styles in Django:

1) Abstract base classes: This style is used when you only want parent's class to hold information that yo
u don't want to type out for each child model.

2) Multi-table Inheritance: This style is used if you are sub-classing an existing model and need each mod
el to have its own database table.

3) Proxy models: This style is used, if you only want to modify the Python level behavior of the model, wit
hout changing the model's fields.

99. How can you set up the database in Djanago?
Ans: A: To set up a database in Django, you can use the command edit mysite/setting.py , it is a normal p
ython module with module level representing Django settings.
By default, Django uses SQLite database. It is easy for Django users because it doesn't require any other
 type of installation. In the case of other database you have to the following keys in the DATABASE 'defau
lt' item to match your database connection settings.

Engines: you can change database by using 'django.db.backends.sqlite3' , 'django.db.backeneds.mysql',
'django.db.backends.postgresql_psycopg2', 'django.db.backends.oracle' and so on

Name: The name of your database. In the case if you are using SQLite as your database, in that case dat
abase will be a file on your computer, Name should be a full absolute path, including file name of that file.
Note: You have to add setting likes setting like Password, Host, User, etc. in your database, if you are not
 choosing SQLite as your database.

100. What does the Django templates contain?
Ans: A template is a simple text file. It can create any text-based format like XML, CSV, HTML, etc. A tem
plate contains variables that get replaced with values when the template is evaluated and tags (%tag%) th
at controls the logic of the template.

101. Is Django a content management system (CMS)?
Ans: No, Django is not a CMS. Instead, it is a Web framework and a programming tool that makes you abl
e to build websites.

102.What is the use of session framework in Django?
Ans: The session framework facilitates you to store and retrieve arbitrary data on a per-site visitor basis. It
 stores data on the server side and abstracts the receiving and sending of cookies. Session can be imple
mented through a piece of middleware.

103. Explain Django Admin Interface?
Ans. The Django Admin interface is predefined interface made to fulfill the need of web developers as the
y won't need to make another admin panel which is time-consuming and expensive.

Django Admin is application imported from django.contrib packages. It is operated by the organization itse
lf and thus doesn't need the extensive frontend.

Admin interface of Django has its own user authentication and most of the general features. It also offers l
ots of advanced features like authorization access, managing different models, CMS (Content Manageme
nt System), etc.

104. Explain Django.

Ans. Django is web application framework which is a free and open source. Django is written in Python. It
 is a server-side web framework that provides rapid development of secure and maintainable websites.

105. What does Django mean?
Ans. Django Reinhardt, was a gypsy jazz guitarist from the 1930s to early 1950s who is known as one of t
he best guitarists of all time. The name was given Django after this person.

106. Which architectural pattern does Django follow?
Ans. Django follows Model-View-Template (MVT) architectural pattern.

The graph below shows the MVT based control flow.

107. Explain Django architecture.
Ans. Django follows MVT (Model View Template) pattern. It is slightly different from MVC.

Model: It is the data access layer. It contains everything about the data, i.e., how to access it, how to valid
ate it, its behaviors and the relationships between the data.

Let’s see an example. We are creating Employee model who has two fields first_name and last_name.

```
from django.db import models

class Employee(models.Model):

first_name = models.CharField(max_length=30)

last_name = models.CharField(max_length=30)
```

View: It is the business logic layer. This layer contains the logic that accesses the model and defers to the
 appropriate template. It is like a bridge between the model and the template.

```
import datetime

# Create your views here.

from django.http import HttpResponse

def index(request):

now = datetime.datetime.now()

html = “&lt;html&gt;&lt;body><h3>Now time is %s.</h3>&lt;/body&gt;&lt;/html>” % now

return HttpResponse(html)    # rendering the template in HttpResponse
```

Template: It is a presentation layer. This layer contains presentation-related decisions, i.e., how somethin
g should be displayed on a Web page or other type of document.

For the configuration of the templates, we have to provide some entries in settings.py file.

```
TEMPLATES = [
{

‘BACKEND’: ‘django.template.backends.django.DjangoTemplates’,

‘DIRS’: [os.path.join(BASE_DIR,’templates’)],

‘APP_DIRS’: True,

‘OPTIONS’: {

‘context_processors’: [

‘django.template.context_processors.debug’,

‘django.template.context_processors.request’,

‘django.contrib.auth.context_processors.auth’,

‘django.contrib.messages.context_processors.messages’,

],

},

},

]
```

108. Explain the working of Django?
Ans. Django can be broken into following components:

Models.py : Models.py file will define your data model by extending your single line of code into full datab
ase tables and add a pre-built administration section to manage content.

Urls.py : Uses a regular expression to capture URL patterns for processing.

Views.py : This is main part of Django. The presentation logic is defined in this.

When a visitor visits Django page, first Django checks the URLs pattern you have created and use this in
formation to retrieve the view. Then it is the responsibility of view to processes the request, querying your
database if necessary, and passes the requested information to a template.

Then template renders the data in a layout you have created and displayed the page.

109. Name the foundation that manages the Django web framework?
Ans. Django is managed and maintained by an independent and non-profit organization named . Goal of f
oundation is to promote, support, and advance the Django Web framework.

110. Comment about Django’s stability?

Ans. Django is quite stable web development framework.There are many companies like Disqus, Instagr am, Pinterest, and Mozilla that are using Django for many years.

111. Specify the features available in Django web framework?
Ans. Features available in Django web framework are:

Admin Interface (CRUD)

Templating

Form handling

Internationalization

A Session, user management, role-based permissions

Object-relational mapping (ORM)

Testing Framework

Fantastic Documentation

112. Explain the advantages of Django?
Ans. Advantages of Django:

Web development framework Django is a Python’s framework which is easy to learn.

It is clear and readable.

It is versatile.

It is fast to write.

No loopholes in design.

It is secure.

It is scalable.

It is versatile.

113. What are the disadvantages of Django?
Ans. Following is the list of disadvantages of Django:

Django’ modules are bulky.

It is completely based on Django ORM.

Components are deployed together.

You must know the full system to work with it.

114. What are the inheritance styles in Django?
Ans. There are three possible inheritance styles in Django:

1) Abstract base class: In this only parent's class to hold information that you don't want to type out for ea ch child model then this style is used.

2) Multi-table Inheritance: This inheritance style is used if you are sub-classing an existing model and nee d each model to have its database table.

3) Proxy models: Proxy models is used, if you only want to modify the Python level behavior of the model, without changing the model's fields.

115. Is Django a CMS i.e. content management system?
Ans. No, Django is not a CMS. But, it is a Web framework and a programming tool that makes you able to build websites.

116. Can you set up static files in Django? How?
Ans. Yes we can. We need to set three main things to set up static files in Django:

1) Set STATIC_ROOT in settings.py

2) run manage.py collect static

3) Static Files entry on the PythonAnywhere web tab

117. What is some typical usage of middlewares in Django?
Ans. Some usage of middlewares in Django is:

Session management,

Use authentication

Cross-site request forgery protection

Content Gzipping

118. What is the use of Django field class type?
Ans. Django field class type specifies:

The database column type.

Default HTML widget used to avail while rendering a form field.

The minimal validation requirements used in Django admin.

Automatic generated forms.

119. Explain the use of Django-admin.py and manage.py?
Ans. admin.py: This is Django's command line utility for administrative tasks.

Manage.py: This file is created automatically in each Django project. It is a thin wrapper around the Djang o-admin.py. It has the following usage:

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

DJANGO_SETTING_MODULE is the environment variable used to points to your project's setting.py file.

120. What are the signals in Django?
Ans. Signals in Django are pieces of code which contain information about what is happening. A dispatch er is used to sending the signals and listen for those signals.

121. What are the two important parameters in signals?
Ans. Two important parameters in signals are:

Receiver: It specifies the callback function which connected to the signal.

Sender: It specifies a particular sender from where a signal is received.

122. How to handle URLs in Django?
In order to handle URL in Django, django.urls module is used by the Django framework.

Given below is the urls.py file of the project, lets see how it looks:

```
// urls.py

from django.contrib import admin

from django.urls import path

urlpatterns = [

path('admin/', admin.site.urls),

]
```

We can see that, Django has already mentioned a URL here for the admin. Function path takes the first argument as a route of string or regex type.

Argument view is a view function which is used to return a response (template) to the user.

Module django.urls contains various functions, path(route,view,kwargs,name) is one of those which is use d to map the URL and call the specified view.

123. What is Django Session?
Ans. In Django session is a mechanism to store information on the server side during the interaction with the web application. Session stores in the database and also allows file-based and cache based sessions, by default,

124 Explainthe role of Cookie in Django?
Ans. Cookie is nothing but a small piece of information which is stored in the client browser. Cookies are used to store user’s data in a file permanently (or for the specified time). There is an expiry date and time for each cookie and removes automatically when gets expire. There are built-in methods to set and fetch cookie provided by Django.

set_cookie() method is used to set a cookie and get() method is used to get the cookie.

request.COOKIES[‘key’] is an array which canbe used to get cookie values.

```
from django.shortcuts import render

from django.http import HttpResponse

def setcookie(request):

response = HttpResponse(“Cookie Set”)

response.set_cookie(‘java-tutorial’, ‘javatpoint.com’)

return response

def getcookie(request):

tutorial  = request.COOKIES[‘java-tutorial’]

return HttpResponse(“java tutorials @: “+  tutorial);
```

125. What is the difference between Flask and Django?

Comparison Factor	Django	Flask
Project Type	Supports large projects	Built for smaller projects
Templates, Admin and ORM	Built-in	Requires installation
Ease of Learning	Requires more learning and practice	Easy to learn
Flexibility	Allows complete web development without the need for third-party tools	More flexible as the user can select any third-party tools according to their choice and requirements
Visual Debugging	Does not support Visual Debug	Supports Visual Debug
Type of framework	Batteries included	Simple, lightweight
Bootstrapping-tool	Built-it	Not available

126. How do you check for the version of Django installed on your system?
Ans:

To check for the version of Django installed on your system, you can open the command prompt and ente r the following command:

python -m django –version
You can also try to import Django and use the get_version() method as follows:

1
2
importdjango
print(django.get_version())

127. What is the usage of middlewares in Django?
Ans: Middlewares are used go to modify the request i.e. HttpRequest object which is sent to the view, to modify the HttpResponse object returned from the view and to perform an operation before the view exec utes.

128. What are the roles of receiver and sender in signals?
The receiver is the callback function which will be connected to a signal
The sender specifies a particular sender to receive signals from
129. What does Django templates contain ?
Ans: Django templates contains the static parts of the desired HTML output as well as some special synta x describing how dynamic content will be inserted.

130. How to create super user in django ?
To create a super user,,

Create project using the django-admin startproject command.
Move into the project location and run python manage.py makemigrations && python manage.py migrate && python manage.py createsuperuser
131. How to create simple application in django ?
To create a simple application use the command django-admin startproject followed by the application’s name.

132. What is ORM ? Advantages of ORM ?
ORM (Object-relational mapping) is a programming technique for converting data between incompatible t ype systems using object-oriented programming languages.

Advantages include:

Concurrency support
Cache management
133. How to create a model in django ?
Add the model object in the models.py file, updated settings for the newly created app by adding it to the I NSTALLED_APPS section in settings.py, make migrations, and verify the database schema.

134. What is migration in django ?
Migrations are a way of propagating changes made in the model into the database schema (adding a field , deleting a model, etc.)

135. How to do migrations in django ?
To do migrations , create or update a model and in the app directory, run the command ./manage.py mak

emigrations <app name> && ./manage.py migrate <app name>

136. How to clear cache in django ?
To clear cache, run the clear() method from django.core.cache in a python script.

137. What is Rest API ?
A REST API is an application program interface that uses HTTP requests to GET, PUT, POST and DELE
TE data.

138. How to Create APIs in Django ?
Create a project directory, create python virtual environment,   and activate it, install Django and djangore
stframework using the pip install command. In the same project directory, create project using the comm
and django-admin.py startproject api. Start the app. Add the rest_framework and the Djano app to INSTA
LLED_APPS to settings. Open the api/urls.py and add urls for the Django app. We can then create model
s and make migrations, create serializers, and finally wiring up the views.

139. What is DRF of Django Rest Frame work ?
Django Rest Framework (DRF) is a powerful module for building web APIs. It’s very easy to build model-b
acked APIs that have authentication policies and are browsable.

140. How to Fetch data from apis using Django ?
We use the Fetch API and SessionAuthentication by adding it to the settings.py file on the server and on
the client, include the getCookie method. Finally, use the fetch method to call your endpoint.

141. How to update the data from apis ?
We update data by sending PUT requests. Add a new path in the data model urlpatterns from which the u
pdate will be sent to. We then add an update method to the serializer that will do the update.

142. What is Authentication ?
Authentication is the process or action of verifying the identity of a user or process.

143. Types of Authentication in REST API ?
Token based authentication and Session based authentication.

144. What is token based authentication system ?
A token based authentication system is a security system that authenticates the users who attempt to log
in to a server, a network, or some other secure system, using a security token provided by the server

145. Can i use django apis in mobile application development ?
Yes
146. Explain Mixins in Django ?
A mixin is a special kind of multiple inheritance. There are two main situations where mixins are used: to p
rovide a lot of optional features for a class and to use one particular feature in a lot of different classes

147. Different types caching strategies in django ?
Different types of caching strategies include Filesystem caching, in-memory caching, using memcached
and database caching.

148. How a request is process in Django ?
When the user makes a request of your application, a WSGI handler is instantiated, which:

imports your settings.py file and Django’s exception classes.
loads all the middleware classes it finds in the MIDDLEWARE_CLASSES or MIDDLEWARES(depending

on Django version) tuple located in settings.py
builds four lists of methods which handle processing of request, view, response, and exception.
loops through the request methods, running them in order
resolves the requested URL
loops through each of the view processing methods
calls the view function (usually rendering a template)
processes any exception methods
loops through each of the response methods, (from the inside out, reverse order from request middleware
s)
finally builds a return value and calls the callback function to the web server
149. When to use iterator in Django ORM ?
The iterator is used when processing results that take up a large amount of available memory (lots of sm
all objects or fewer large objects).

150. What are signals in Django ?

Signals allow certain senders to notify a set of receivers that some action has taken place. They're especi
ally useful when many pieces of code may be interested in the same events.

151. How to implement social login authentication in Django ?
Run the development server to make sure all is in order. The install python-social-auth using the pip instal
l command. Update settings.py to include/register the library in the project Update the database by maki
ng migrations. Update the Project's urlpatterns in urls.py to include the main auth URLs. Create a new ap
p https://apps.twitter.com/app/new and make sure to use the callback url http://127.0.0.1:8000/complete/t
witter. In the project directory, add a config.py file and grab the consumer key and consumer secret and a
dd them to the config file. Finally add urls to the config file to specify the login and redirect urls. Do a sanit
y check and add friendly views.

152. Where to store static files in django ?
Static files are stored in the folder called static in the Django app.

The Questions
General
153. Explain the use of session framework in Django? ↑
In Django, the session framework enables you to store and retrieve arbitrary data on a per-site-visitor basi
s. It stores data on the server side and abstracts the receiving and sending of cookies. Session can be im
plemented through a piece of middleware.

154. List out the inheritance styles in Django? ↑
In Django, there is three possible inheritance styles

Abstract base classes: This style is used when you only wants parent's class to hold information that you
don't want to type out for each child model
Multi-table Inheritance: This style is used If you are sub-classing an existing model and need each model t
o have its own database table
Proxy models: You can use this model, If you only want to modify the Python level behavior of the model,
without changing the model's fields
155. What is Django? ↑
Django is a web development framework that was developed in a fast-paced newsroom. It is a free and o
pen-source framework that was named after Django Reinhardt who was a jazz guitarist from the 1930s. D
jango is maintained by a non-profit organization called the Django Software Foundation. The main goal of
Django is to enable Web Development quickly and with ease.

156. How do you connect your Django project to the database? ↑
Django comes with a default database which is SQLite. To connect your project to this database, use the f

ollowing commands:

python manage.py migrate (migrate command looks at the INSTALLED_APPS settings and creates datab ase tables accordingly)
python manage.py makemigrations (tells Django you have created/ changed your models)
python manage.py sqlmigrate (sqlmigrate takes the migration names and returns their SQL)
157. What are Models? ↑
Models are a single and definitive source for information about your data. It consists of all the essential fiel ds and behaviors of the data you have stored. Often, each model will map to a single specific database ta ble.

In Django, models serve as the abstraction layer that is used for structuring and manipulating your data. D jango models are a subclass of the django.db.models.Model class and the attributes in the models repres ent database fields.

158. What are views? ↑
Django views serve the purpose of encapsulation. They encapsulate the logic liable for processing a user' s request and for returning the response back to the user. Views in Django either return an HttpResponse or raise an exception such as Http404. HttpResponse contains the objects that consist of the content that is to be rendered to the user. Views can also be used to perform tasks such as read records from the data base, delegate to the templates, generate a PDF file, etc.

159. What are templates? ↑
Django's template layer renders the information to be presented to the user in a designer-friendly format. Using templates, you can generate HTML dynamically. The HTML consists of both static as well as dyna mic parts of the content. You can have any number of templates depending on the requirement of your pr oject. It is also fine to have none of them.

Django has its own template system called the Django template language (DTL). Regardless of the backe nd, you can also load and render templates using Django's standard admin.

160. What is the difference between a Project and an App? ↑
An app is basically a Web Application that is created to do something for example, a database of employe e records. A project, on the other hand, is a collection of apps of some particular website. Therefore, a sin gle project can consist of 'n' number of apps and a single app can be in multiple projects.

161. What is mixin? ↑
Mixin is a type of multiple inheritance wherein you can combine behaviors and attributes of more than one parent class. Mixins provide an excellent way to reuse code from multiple classes. For example, generic class-based views consist of a mixin called TemplateResponseMixin whose purpose is to define rendertor esponse() method. When this is combined with a class present in the View, the result will be a TemplateVi ew class.

One drawback of using these mixins is that it becomes difficult to analyze what a child class is doing and which methods to override in case of its code being too scattered between multiple classes.

162. When can you use iterators in Django ORM? ↑
Iterators in Python are basically containers that consist of a countable number of elements. Any object tha t is an iterator implements two methods which are, the init() and the next() methods. When you are makin g use of iterators in Django, the best situation to do it is when you have to process results that will require a large amount of memory space. To do this, you can make use of the iterator() method which basically e valuates a QuerySet and returns the corresponding iterator over the results.

163. What are the signals in Django? ↑
Signals are pieces of code which contain information about what is happening. A dispatcher is used to se

nding the signals and listen for those signals.

164. What is the role of Cookie in Django? ↑
A cookie is a small piece of information which is stored in the client browser. It is used to store user's data in a file permanently (or for the specified time). Cookie has its expiry date and time and removes automati cally when gets expire. Django provides built-in methods to set and fetch cookie.

The set_cookie() method is used to set a cookie and get() method is used to get the cookie.

165. Django is an MVC based framework, how does this framework implement MVC? ↑
Django is based on MTV architecture which is a variant of MVC architecture. MVC is an acronym for Mod el, View, and Controller. There are different parts of a website so that they can develop and execute in diff erent machines to achieve faster and more responsive websites. Django implements MTV architecture by having 3 different components and they are all handled by Django itself.

Models are the part which is models.py file in a Django application, which defines the data structure of the particular application.

View are the mediators between models and templates, they receive the data from the Model and make it a dictionary and return the same as a response to a request to the Template.

The Template is the component with which user interacts, and it generates both statically and dynamically in the Django server.

That’s how the Django implements 3 components and work in coordination with each other.

166. How is Django’s code reusability feature different from other frameworks? ↑
Django framework offers more code-reusability then other frameworks out there. As Django Project is a c ollection of different applications like login application, signup application. These applications can be just c opied from one directory to another with some tweaks to settings.py file and you won’t need to write new s ignup application from scratch.

That’s why Django is a rapid development framework and this level of code reusability is not there in other frameworks.

167. What happens when a typical Django website gets a request? ↑
When a user enters a URL in the browser the same request is received by the Django Server. The server then looks for the match of the requested URL in its URL-config and if the URL matches, it returns the cor responding view function. It will then request the data from the Model of that application, if any data is req uired and pass it to the corresponding template which is then rendered in the browser, otherwise, a 404 er ror is returned.

168. Why is Django called loosely coupled framework? ↑
Django is called a loosely coupled framework because of the MTV architecture it’s based on. Django’s arc hitecture is a variant of MVC architecture and MTV is useful because it completely separates server code from the client’s machine.

Django’s Models and Views are present on the client machine and only templates return to the client, whic h are essentially HTML, CSS code and contains the required data from the models.

These components are totally different from each other and therefore, front-end developers and backend developers can work simultaneously on the project as these two parts changing will have little to no effect on each other when changed.

Therefore, Django is called a loosely coupled framework.

169. Explain the importance of settings.py file and what data/ settings it contains. ↑
When Django server starts, it first looks for settings.py. As the name settings, it is the main settings file of your web application. Everything inside your Django project like databases, backend engines, middlewares, installed applications, main URL configurations, static file addresses, templating engines, allowed hosts and servers and security key stores in this file as a list or dictionary.

So, when your Django server starts it executes settings.py file and then loads particular engines and databases so that when a request is given it can serve the same quickly.

170. Why does Django use regular expressions to define URLs? Is it necessary to use them? ↑
Django uses a very powerful format for storing URLs, that is regular expressions. RegEx or regular expression is the format for sophisticated string searching algorithms. It makes the searching process faster. Although it’s not necessary to use RegEx when defining URLs.

They can be defined as normal string also, Django server should still be able to match them, but when you need to pass some data from the user via URL, then RegEx is used. The RegEx also makes much cleaner URLs then other formats.

171. Django is too monolithic. Explain this statement. ↑
Django framework is too monolithic, that is true to some extent. Django is MTV architecture based framework and since Django is the controller of the architecture, it requires some rules that the developer should follow so that the framework can find and execute appropriate files at the right time. Therefore, Django is one of the frameworks where file structure is as important as its architecture. In Django, you get great customisability with the implementations. There is just one condition that you cannot change the file names, the pre-defined lists and variable names.

You can create new ones but you can’t change the pre-defined variables for which people say that they always have to follow a certain pattern while working on Django.

Django’s file structure is one of the most logical workflows. The monolithic behavior is actually helping the developers to easily understand the project. Even, when the company changes, the project layout remains the same. Therefore, the developer would take less time to understand every aspect, will be able to perform more work productively.

172. Why permanent redirecting is not a good option? ↑
Permanent redirecting is not good an option because the browser caches the response generated by the permanent redirect. This is the difference between permanent and temporary redirect. It causes all sorts of issues when you change that redirect to something different.

Since the browser has cached the redirect before, this time it won’t look on the server for the changed redirection and will load the previously saved redirect. So, even though the developer might have redirected the user to a different page, it will still load the same page. It is browser/ client-side operation, therefore, the user can’t even do anything about the same.

Because of this reason, permanent redirecting is not a good option as informing the users to clear their internal caching data is not good for any website.

173. Explain user authentication in Django? ↑
Django comes with a built-in user authentication system, which handles objects like users, groups, user-permissions and some cookie-based user sessions. Django’s User authentication not only authenticates (verifying the user identity) the user but also authorizes him (determines what permissions user have).

The system consists and operates on these objects: - users - Permissions - Groups - Password Hashing System - Forms Validation - A pluggable backend system

There are many third-party web applications which we can use in place of the default system as they provi de much more control over user authentication with more features.

174. Explain context variable lookups in Django. ↑
Context variables are variables passed in templates. The DTL (Django Templating Language) replaces th ese variables. A context dictionary is used to perform the replacement. We pass the context dictionary alo ngside the render() alongside template information.

DTL has its own way of filling these variables and it's in order. The template system can handle complex Python data structures as variables. The context variable lookups come in the role when the data is a dicti onary, class object, etc.

The context lookup will fill the value in this order. - Dictionary Values: The template system will look for dic tionaries matching the variable name. It matches the key name with the name after the dot(.) operator. - C lass Object Attribute values: If it didn't find any dictionaries it will look for a class object. First lookups are done for attributes. - Class Object Methods: It looks for a particular attribute of that name. If the attribute is not found then it looks for methods. - List-Index lookup: At last, it will look for any lists with corresponding names. If it didn't find any lists then it considers variable as an Invalid Variable.

175. What are custom validation rules in form data? ↑
Custom validation rules are the customized rules used for form validation. Suppose we have a feedback f orm. There are fields like messages, email, and subject. If we get message data of 1 or 2 words that are o f no use to us. To check the issue, we use custom validation rules.

We simply define a method in our forms.py by the name clean_message().

This is the Django way to do it. The method's name for custom validation should start with a clean_fieldna me(). Django form system automatically looks for this type of method. Thus, these are called custom valid ation rules in Django.

It is always important to return the cleaned data of the field in a custom validation method. If not done, the method will return none instead resulting in loss of data.

176. What is the difference between authentication and authorization? ↑
Authentication and authorization are two different terms. The authentication means the verification of the entity(user) in Django's context. Authentication will verify that the user is what they claim to be.

Authorization is the step after authentication. It sets the actions that can be performed by the authenticate d user. Authorization is a grouping of users and allowing limited actions.

Both of these functionalities are achieved by django.contrib.auth application. It is a built-in application in D jango.

177. What is Pagination? ↑
Pagination is a concept where we separate or divide data into different pages. It's divided into multiple pa ges. The pages are provided as per user-request.

In the context of web applications: Suppose you search for anything on Google. Though Google claims th ey have found thousands of results in the resulting page, they give you only 10-20 results. Then you press the next page and you get more results. That is pagination.

Showing user only limited data is good for both of them. The user can sort the results easily and smoothly reach the information he/she wants. The bandwidth cost is not much for some results. Since the transfer-data is small, it gives results on low network speeds.

Pagination is also good for the server. The server can simply store the data in queryset or database and wait for the client to request more. This reduces the template overhead which would have been to generat e a response. Generating a response with 1000 data is much more expensive than 10-20 data.

Pagination can drastically improve performance if done the right way.

178 Explain the use of migrate command in Django? ↑
In Django, migrations are used to propagate changes made to the models. The migrate command is basic ally used to apply or unapply migrations changes made to the models. This command basically synchroni zes the current set of models and migrations with the database state. You can use this command with or without parameters. In case you do not specify any parameter, all apps will have all their migrations runni ng.

179. What are the roles of receiver and sender in signals? ↑
The roles of receiver and sender in signals are: - Receiver: It specifies the callback function which will be connected to the signal. - Sender: It specifies a particular sender to receive a signal from.

180. What is CSRF? ↑
CSRF – Cross Site Request Forgery. Csrf tokens could also be sent to a client by an attacker due to sessi on fixation or other vulnerabilities or guessed via a brute-force attack, rendered on a malicious page that g enerates thousands of failed requests.

181. What is CRUD? ↑
The most common task in web application development is to write create, read, update and delete functio nality (CRUD) for each table. It refers to the set of common operations that are used in web applications t o interact with data from the database. It provides a CRUD interface that allows users to create, read, upd ate or delete data in the application database.

Django helps us with its simplified implementation for CRUD operations using Function-Based views and class-based views:

Function- based views are simple to implement and easy to read but they are hard to customize or extend the functionality. Code reuse is not allowed and so there is repetitiveness.

Class-based views - In no time CRUD operations can be implemented using CBVs. As the model evolves changes would be reflected automatically in CBVs. CBVs are easily extendable and allow code reuse. Dja ngo has built-in generic CBVs which makes it easy to use.

182. Can you tell us something about the Django admin interface? ↑
Django is actually of preloaded interface designed to fulfill the requirement of web developers. Basically, it indicates any requirements for the web developer to make other admin because the whole process is tim e-consuming and costly. Django admin interface supports user authentication and follows most of the incl uded features. The application Django admin is imported from the Django.contrib package. This imported application is also expected to get control by the corresponding organization hence it does not require an additional front end.

The Django admin interface provides a number of advanced features like-

Authorization access

Managing multiple models

Content management system

183. How is the reusability feature of Django different from the rest of the frameworks? ↑
Django offers maximum code reusability to the developers as compared to other frameworks. This frame work is also a collection of various applications including login application or signup application. With the h elp of this framework, a large number of applications can directly be copied from one directory to the next following some of the settings.py files. With this framework, developers can easily work over the applicatio n without writing the new sign up application. This is the reason which supports the rapid development fra mework in Django and there are no other compatible frameworks supporting this level of code reusability.

184. How does Django's admin interface support customization? ↑
Django admin interface easily supports the customization and developer can download various other third -party applications and install them in a completely different view. Also, if the developer wants overall cont rol over their admin then they can make their own admin application. The admin site can also be customiz ed for example changing the properties of admin.site.object. this admin interface also supports the chang es in modifications in models and to apply them in Django admin for specific applications. The Django ad min interface supports customization exact from the lowest level and the developer can also create new a dmin interfaces.

185. How are RESTful APIs beneficial for developers? ↑
RESTful APIs are extremely beneficial for web developers to build web applications as they consume the l owest bandwidth and their design in such a way to communicate well with the internet mainly like PUT, P OST, GET, etc.

186. What is the use of the include function in the urls.py file in Django? ↑
As in Django there can be many apps, each app may have some URLs that it responds to. Rather than re gistering all URLs for all apps in a single urls.py file, each app maintains its own urls.py file, and in the proj ect’s urls.py file we use each individual urls.py file of each app by using the include function.

187. Explain how you can use file based sessions? ↑
To use file based session you have to set the SESSION_ENGINE settings to “django.contrib.sessions.ba ckends.file”

188. What is the typical usage of middlewares in Django? ↑
Some usage of middlewares in Django is:

Session management
Use authentication
Cross-site request forgery protection
Content Gzipping
189. What is DRF of Django Rest Frame work? ↑
Django Rest Framework (DRF) is a powerful module for building web APIs. It’s very easy to build model-b acked APIs that have authentication policies and are browsable.

190. What is token based authentication system? ↑
A token based authentication system is a security system that authenticates the users who attempt to log i n to a server, a network, or some other secure system, using a security token provided by the server

191. How to implement social login authentication in Django? ↑
Run the development server to make sure all is in order. The install python-social-auth using the pip instal l command. Update settings.py to include/register the library in the project Update the database by makin g migrations.

Update the Project’s urlpatterns in urls.py to include the main auth URLs. Create a new app https://apps.t witter.com/app/new and make sure to use the callback url http://127.0.0.1:8000/complete/twitter. In the pr oject directory, add a config.py file and grab the consumer key and consumer secret and add them to the config file.

Finally add urls to the config file to specify the login and redirect urls. Do a sanity check and add friendly vi ews.

192. Mention the differences between Django, Pyramid and Flask. ↑
Flask is a “microframework” primarily build for a small application with simpler requirements. In flask, you have to use external libraries. Flask is ready to use.
Pyramid is built for larger applications. It provides flexibility and lets the developer use the right tools for th eir project. The developer can choose the database, URL structure, templating style and more. Pyramid is heavy configurable.
Django can also used for larger applications just like Pyramid. It includes an ORM.
41. Explain the use of decorators. ↑
Decorators in Python are used to modify or inject code in functions or classes. Using decorators, you can wrap a class or function method call so that a piece of code can be executed before or after the execution of the original code. Decorators can be used to check for permissions, modify or track the arguments pas sed to a method, logging the calls to a specific method, etc.

Advanced
193. How would you compare Node.js and Django? ↑
It heavily depends on the priorities you set in your project. The most common criteria are: database type (e.g. Django for a relational one), security (Django offers great security), rapid development (also Django). Other criteria may include: better performance (Node.js is more fitting), creating features from the ground up (also Node.js) or better client-side processing (Node.js).

A distinctive caveat of Node.js lies in its asynchronous elements: they require the developer to be ever so vigilant because code errors may not reveal themselves until late in production.

194. How can we use model inheritance? ↑
Django’s object-oriented models can be easily mapped to database table structures, creating inheritance properties that they will be sharing. That way, the inheriting model will not interfere with the base one. It is common for a web project to include multiple models; to showcase inheritance, we can use a base model (which we can name “content” and have it store general description values related to it: title, description, d ates of creation and/or modification, etc.) together with another model (that we can name “audio”). The “a udio” model will inherit the properties of the base model “content”, while also utilizing its own: source, link, embed code.

Having created the base model, we let Django map it to a table named content — and makes it inherit fro m model.Model. We then create the “audio” model but make it inherit from “content” instead of model.Mod el. The convenience of this method comes from the fact that Django manages the inheritance systems aut onomously, creating two tables: content and audio respectively. This relation can also be accessed via S QL methods.

195. How can we optimize a Django project’s performance? ↑
Although it depends on the databases and models used in the project, there are some methods that alway s prove to be effective:

Analyzing the runtime of functions via the line_profiler module. Sometimes we may notice that the code ru ns slowly — and we need to examine our functions line-by-line. Using the IPython debugger with the line_ profiler module, we can see exactly how much time each function takes to execute — and then we optimiz e the code, if needed.
SQL logging. To dissect a function suspected of low performance even further, we can utilize SQL logging , getting a list of every SQL query that gets executed. It is important to limit this type of logging to a single function — otherwise, the printout would simply flood us with information.
196. What is Unicode, what is UTF-8 and how do they relate? ↑
Unicode is an international encoding standard that works with different languages and scripts. It consists

of letters, digits or symbols representing characters from across the world. UTF-8 is a type of encoding, a way of storing the code points of Unicode in a byte form, so you can send Unicode strings over the network or store them in files.

197. How would you scale an existing application when starting a new project? ↑
I see performance and scaling as two separate things. Performance is how fast a user is served and scaling refers to the number of users that can be served by an app at the same time. Usually, time is best spent developing during the early stages of a project. When scale does become an issue, usually business is good and there are sufficient funds to optimize the application.

198. Are there situations where you wouldn't use Python/Django? ↑
Sure. For example, if a project involves some kind of reasoning it might be better to use Prolog and have Python interface with it. Of course, I would be mindful about
adding more complexity to the stack by introducing a new language.
199. What is Django Admin Interface?

Ans. Django Admin is the preloaded interface made to fulfill the need of web developers as they won't need to make another admin panel which is time-consuming and expensive.

Django Admin is application imported from django.contrib packages.

It is meant to be operated by the organization itself and therefore doesn't need the extensive frontend.

Django's Admin interface has its own user authentication and most of the general features.

It also offers lots of advanced features like authorization access, managing different models, CMS (Content Management System), etc.

200. How is Django's code reusability feature different from other frameworks?

Ans. Django framework offers more code-reusability than other frameworks out there.

As Django Project is a collection of different applications like login application, signup application.

These applications can be just copied from one directory to another with some tweaks to settings.py file and you won't need to write new signup application from scratch.

That as why Django is a rapid development framework and this level of code reusability is not there in other frameworks.