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, pragmatic 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 appropriate 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 be changed or formatted for display on the page.

Controller: Controller is the heart of the system. It handles requests and responses, setting up database connections 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 named Django Software Foundation (DSF).

---

6. Is Django stable?

Ans: Yes, Django is quite stable. Many companies like Disqus, Instagram, Pinterest, and Mozilla have been 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 command:

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 you 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 model 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, without 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 python 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 '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 database 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 template contains variables that get replaced with values when the template is evaluated and tags (%tag%) that 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 able 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 implemented 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 file 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 users 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.

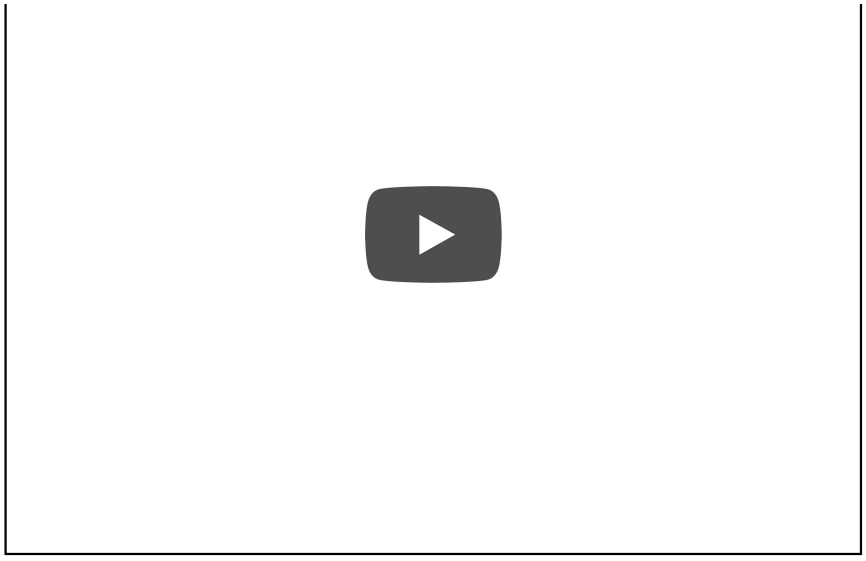
B. Run the `manage.py validate` command, and then start a new shell.

C. Run the `manage.py syncdb` command.

D. In settings.py, add the app to the PROJECT_APPS variable.

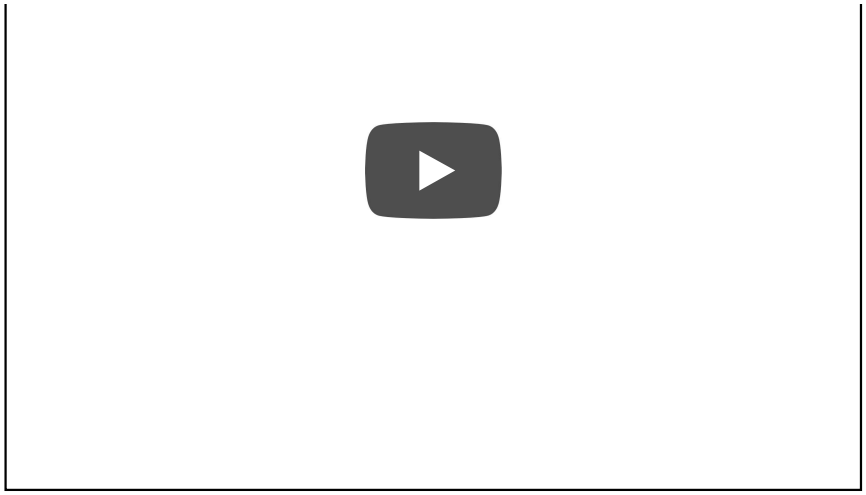
E. In settings.py, add the new app to the INSTALLED_APPS variable.

Ans: E

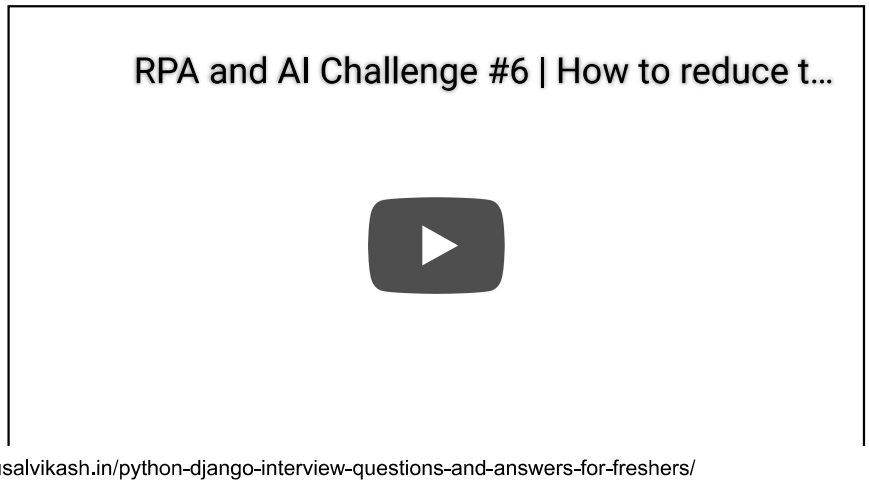



Challenge #3 : Why Service Delivery is not ...


Challenge#4 Why Digital workforce will au...




RPA & AI Challenge #5 Virtual Integration ...


RPA and AI Challenge #6 | How to reduce t...

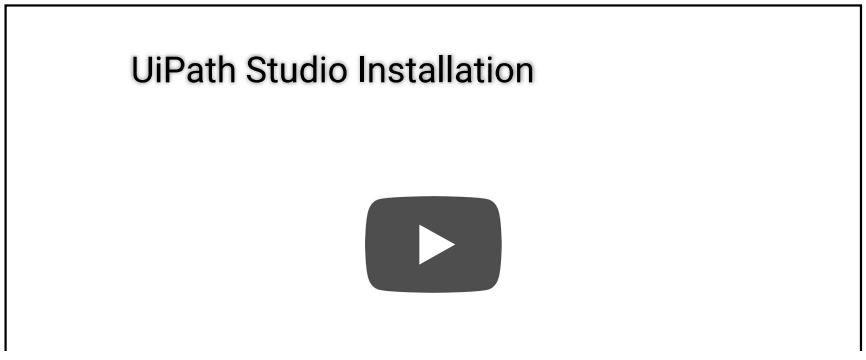
Our Interview Q & A

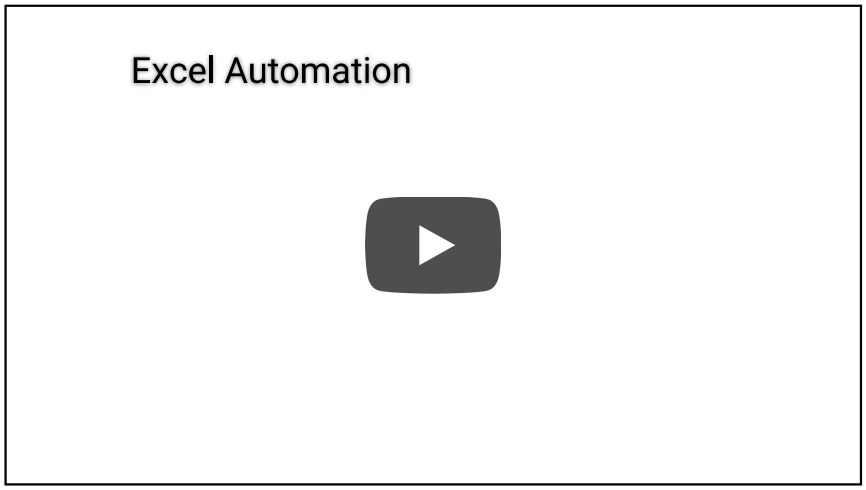
- **Blue Prism Interview Questions & Answers**
- **UiPath Interview Questions & Answers**
- **Python Interview Questions & Answers**
- **Django Interview Questions & Answers**
- **Pandas Interview Questions & Answers**
- **Machine Learning Interview Questions And Answers**

Our Blogs

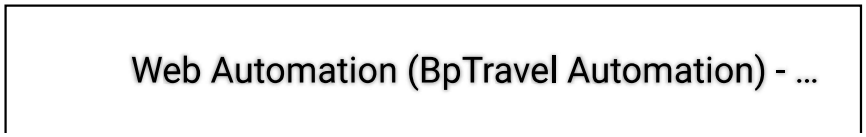
Cognitive RPA And The Need For The Change
Non_IT Person UiPath RPA Journey
Future of Manufacturing Using AI
The Difference Between Artificial Intelligence, Machine learning And Deep Learning
Top 10 Key Features to Grow your Career with RPA
How Robotics Industry Overcome the Awareness Gap?
RPA : Is it the fresh IT job killer?
THE HISTORY OF DATA SCIENCE
Fundamentals of Robotic Process Automation (RPA)

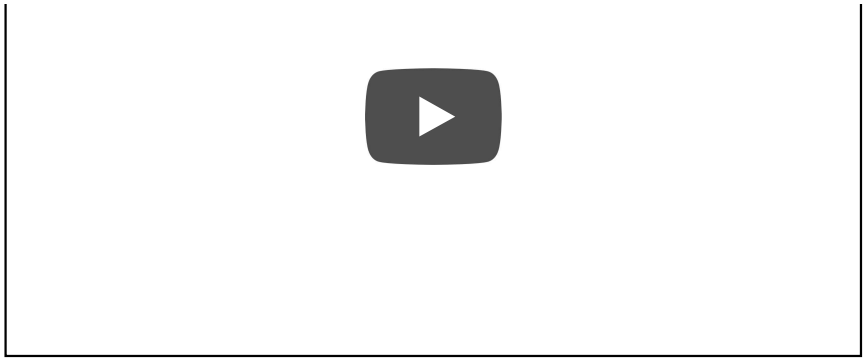
Our Learning Videos

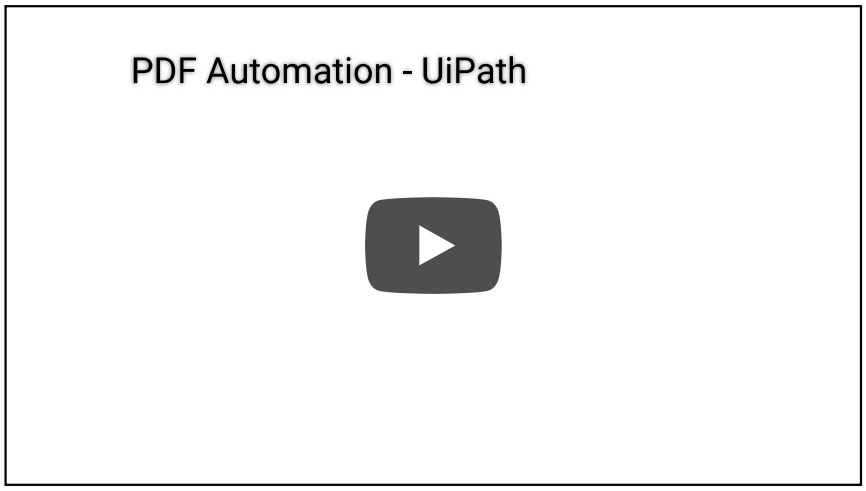



Excel Automation

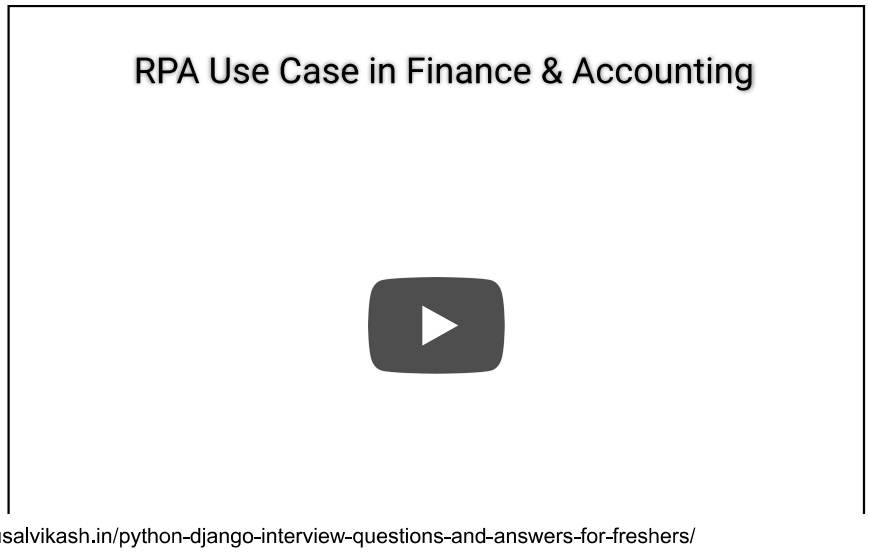

Email Automation


Web Automation (BpTravel Automation) - ...





RPA Use Cases



RPA Use Cases in ITSM IT Infrastructure

RPA Use Case | Account Payable Automati...

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 model 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 model 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. This 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 have 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 changes to all the data in the database of registered apps and models. To use a database table with the admin interface, 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 looks for a matching URL in the urlpatterns defined for the project. If no matching URL is found, then a response with 404 status code is returned. If a URL matches, then the corresponding code in the view file associated 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 database 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 database schema by writing queries in our DBMS shell, we can just make changes to our models. Then, we can use Django to generate migrations from those model changes and run those migrations to make changes 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 is usually associated with urlpatterns, and the logic encapsulated in a view is run when a request to the URL associated with it is run. A view, among other things, gets data from the database using models, passes 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 than registering all URLs for all apps in a single urls.py file, each app maintains its own urls.py file, and in the 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 the 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 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 be 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 representing Django settings.
Django uses SQLite by default; it is easy for Django users as such it won't require any other type of installation. 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 database 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 added.

---

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 called 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 template 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-visitor 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.sessions.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 command 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 terms template is a simple text file. It is made up of variables that will later be replaced by values after the template 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 called 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) and 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 generally used with a middle-ware. It also helps in receiving and sending cookies for authentication of a user.

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  class Singleton(object):def __new__(cls,*args,**kwargs):
3   if not hasattr(cls,'_inst'):
4   cls._inst = super(Singleton,cls).__new__(cls,*args,**kwargs)
5   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 considered 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_for
m=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 follows 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 the 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 URL.

Once one of the regexes matches, Django imports and calls the given view, which is a simple Python function (or a class based view). The view gets passed an HttpRequest as its first argument and any values captured 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 appropriate 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 tables 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 information 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 named Django Software Foundation (DSF).

---

94. Is Django stable?

Ans: Yes, Django is quite stable. Many companies like Disqus, Instagram, Pinterest, and Mozilla have been 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 command:
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 you 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 model 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, without 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 python 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 '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 database 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 template contains variables that get replaced with values when the template is evaluated and tags (%tag%) that 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 able 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 implemented 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 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 operated by the organization itself 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 lots of advanced features like authorization access, managing different models, CMS (Content Management 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 the 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.

Request is made by the user for a resource to the Django, Django works as a controller and check to the available resource in URL.

When the mapping of URL is found , a view is called that interact with model and template, it renders a template.

After that Django responds back to the user and sends a template as a response.

---

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 validate 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 something 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 database 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 information 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 foundation 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, Instagram, 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 each 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 need 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 Django-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 dispatcher 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 used 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 enter 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 executes.

---

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 syntax 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 type 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 **INSTALLED_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 makemigrations <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 DELETE data.

---

138. How to Create APIs in Django ?

Create a project directory, create python virtual environment, and activate it, install Django and **djangorestframework** using the **pip install command**. In the same project directory, create project using the command **django-admin.py startproject api.** Start the app. Add the **rest_framework** and the Djano app to INSTALLED_APPS to settings. Open the **api/urls.py** and add urls for the Django app. We can then create models 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-backed 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 update 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 provide 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 middlewares)
- 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 small 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 especially 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 install command. Update settings.py to include/register the library in the project Update the database by making migrations. Update the Project's urlpatterns in urls.py to include the main auth URLs. Create a new app https://apps.twitter.com/app/new and make sure to use the callback url http://127.0.0.1:8000/complete/twitter. In the project 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 views.

152. Where to store static files in django ?

Static files are stored in the folder called **static** in the Django app.

Erandwane, Pune,
Maharashtra

info@kausalvikash.in

+91-735-007-0755

Kharadi

Supreme Arcade, 1st Floor,
Office No -5,

Above More Store, Kharadi,
Pune, Maharashtra

info@kausalvikash.in

+91-735-007-0755

Wakad

Sonigara Landmark Building,
3rd Floor , 302

Near Chatrpati Chowk ,
Kaspate Wasti , Wakad Pune,
Maharashtra

info@kausalvikash.in

+91-735-007-0755

Keep In Touch

Noida:

Unboxed Coworking & Shared Office
Space

C 15, C Block Road, C Block, Sector 65,

Noida, Uttar Pradesh 201301

Quick Links

- About Us
- RPA Training in Pune | Mumbai | Delhi
- Blue Prism Training Pune | Mumbai
- Blue Prism Training in Delhi | Noida
- Cognitive RPA Training

info@kausalvikash.in
+91-762-082-4981

Gurgaon:

NM13, Old DLF Colony, Sector 14,

Gurugram, Haryana 122001

info@kausalvikash.in

+91-762-082-4981

- RPA & Cognitive for Strategic Management Training
- RPA UiPath Training Course in Noida
- Python Interview Questions and Answers for Freshers
- Python | Django Interview Questions And Answers for Freshers
- Python | Pandas Interview Questions And Answers for Freshers
- Machine Learning Interview Questions And Answers for Freshers