In project

What Is init.py?

- This file is generated by Django for us. It is mandatory to have an **inti**.py file in a directory to denote that the project is a python package and can be imported into other files. This file usually remains empty.
- If this file gets missing, you will see “package not found error” in the absence of this file.

What Is settings.py?

- This is the core file of our Django projects.
- It contains the configuration values which are needed by web apps to work properly such as database settings, static files location, template location, etc. We will keep coming to this file to edit the project configuration throughout this course.

What is urls.py?

- Url declaration and mapping are made under this file.

What is wsgi.py?

- WSGI stands for web-server gateway interface.
- WSGI is a specification that describes the communication between a web server and a web application.

What is manage.py?

- Command-line utility for performing administrative tasks.
- We will be using manage.py frequently while developing a Django project.

Website related Terms

What is HTTP?

- The ***Hypertext Transfer Protocol*** (HTTP) is designed to enable communications between clients and servers.
- HTTP works as a request-response protocol between a client and server.

> Example: A client (browser) sends an HTTP request to the server; then the server returns a response to the client. The response contains status information about the request and may also contain the requested content.

What Is A Client?

- Computer hardware or software device that can make requests.
- The client accesses the web pages made available by the server.

What Is A Server?

- A physical computer that is responsible for running services to fulfill the requirements of the other computers(clients).
- Example: A shared hosting, VPS, etc (More on hosting and servers later)

What Is A Website?

A website is nothing but a combination of HTML (Hypertext Markup Language), CSS (Cascading Style Sheets), and Javascript. But, we need a back-end system like Django, Flask, PHP, etc., to make sure that the front-end of a website is served as per the user's request.
02:01 - Why We Are Using Python?

Python is an amazing scripting language that is capable of doing much more than building logic. It will be super easy for us to use Python's machine learning and data science libraries on our websites if we use Python as our backend.
We use HTML, CSS, and JavaScript for the front-end development while Python is used for back-end development.
03:10 - What Have We Learnt In This Video?

Difference Between front-end and back-end:

Front-End:

- It is a combination of HTML, CSS, and JavaScript.
- Users can directly view and interact with this data.
- It is also referred to as client-side or frontend design.

Back-end:

- Back-end takes care of when and which webpage should be visible to the client corresponding to a given request.
- Users can not directly view or interact with this data. The back-end is all about the behind-the-scene activities.
- It is also referred to as the server-side.

What Is A Web Browser?

- A software application that allows us to access information on the World Wide Web.
- The browser takes raw HTML, CSS, and JavaScript from a web server and then displays a web page on the screen.

With this, we have completed our discussion on the working of a website and the purpose of using Python while building a website. I hope all the concepts that we have discussed in this blog are crystal clear to you.

Views.py

```
from . import views
```

Where "." represent same directory.

```
urlpatterns = [

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

    path('', views.index, name='index'),

    path('about/', views.about, name='about'),
]
```

What is the path function?

- Path function is contained within the django.urls module.
- It helps in routing URLs to the appropriate view function within a Django project.

Now, let' s discuss the arguments passed in the path function.

1. about/: It is the endpoint of the URL. Example: Let's suppose a user tries to access https://www.codewithharry.com/blog/. In this URL, 'blog' is the endpoint.
2. views.about: It is the function defined in the views.py file. Here, we are passing the function that should be executed whenever someone tries to access the 'about' page of our website.
3. name='about': It is the name of the path. Naming a path will help us to access it from anywhere in our project. In future, we will see how to access a path from templates.

Creating Functions In views.py :

```
from django.http import HttpResponse

def index(request):
    return HttpResponse('''Hello Everyone''')

def about(request):
    return HttpResponse("About Sharique")
```

Now, let's understand the above code line by line:

- First of all, we have imported the HttpResponse object from django.http module.
- Whenever a user requests a page, Django creates an HttpRequest object that contains the metadata of the request. After creating an HttpRequest object, Django passes this object inside the view function as the first argument. Hence, a specific HttpResponse is generated for each view function.

In the second line, we have created a function named index, which returns "Harry Django CodeWithHarry" as HttpResponse. Each view function takes request as a default argument. Similarly, we have created another function called about which returns "About Harry Bhai."

What is bootstrap?

- Bootstrap is a front-end development framework.
- It is a library of HTML, CSS, and JS, which is used to create modern websites and web applications.

Note: Search for css alert-dissmissal

HTTP Methods

- **GET**
- **POST**
- PUT
- HEAD
- DELETE
- PATCH
- OPTIONS

The GET Method

- GET is used to request data from a specified resource.
- GET is one of the most common HTTP methods.
- Note that the query string (name/value pairs) is sent in the URL of a GET request:

> /test/demo_form.php?name1=value1&name2=value2

- Some other notes on GET requests:
    - GET requests can be cached
    - GET requests remain in the browser history
    - GET requests can be bookmarked
    - GET requests should never be used when dealing with sensitive data
    - GET requests have length restrictions
    - GET requests are only used to request data (not modify)

Get request and Post request

- Why the input text exposed in the URL?
- What if we do not want to show the text in the URL?

Now, I will answer the above questions one by one. So let's start with question 1.

Why the input text exposed in the URL?

Ans: The text is visible in the URL because we are using the GET method for passing the information. HTTP is a request-response protocol that takes the request from the user, and then a server returns the response to the user. So, to establish this request-response connection, we need some methods. The two most commonly used HTTP methods are HTTP GET and POST. So, let's understand the GET method :
HTTP GET :

- The GET method is the default submission method for a form.
- The GET method sends the data in the form of URL parameters. Therefore, any data sent with the help of the GET method remains visible in the URL.
- Since the data is exposed in the URL, the GET method is not considered to send sensitive information such as passwords.
- The GET method reveals the data in the URL bar; therefore, the length of the URL increases. The maximum length of a URL is 2048 characters, so only a limited amount of data can be sent using the GET method. The following error occurs when we try to send more than 2048 characters using GET :

The POST Method

POST is used to send data to a server to create/update a resource.

The data sent to the server with POST is stored in the request body of the HTTP request:

```
POST /test/demo_form.php HTTP/1.1
Host: w3schools.com
name1=value1&name2=value2
```

POST is one of the most common HTTP methods.

Some other notes on POST requests:

- POST requests are never cached
- POST requests do not remain in the browser history
- POST requests cannot be bookmarked
- POST requests have no restrictions on data length

Answer of the second question...

What if we do not want to show the text in the URL?

Ans: In case we do not want to expose the data in the URL bar, then we can use the POST method instead of GET. Let's start our discussion on the POST method :

POST :

- Data sent by the POST method never gets visible in the URL box, and therefore it is more secure than the GET method, and sensitive information can be sent with the help of this method.
- Since the data is not visible in the URL query, the length of the URL remains less than 2048 characters, and a large amount of data can be sent with the help of the POST method.
- Data is sent to the server in the form of packages in a separate communication with the processing script.

WHAT ARE CSRF TOKENS?

- CSRF stands for Cross-Site Request Forgery.
- The server-side application generates and transmits a huge, random, and unpredictable number to the client to make sure that the request is coming from the original client and not from a malicious website.
- CSRF tokens are used to protect the site against CSRF attacks.

```
<form action='Analyze' method ="post">{% csrf_token %}
```

Django models:

- Models in Django are the single and definitive source of information.
- With the help of Django models, we can manipulate and retrieve the data instead of writing complex SQL to perform the same task.

- Whenever we create a model, Django automatically executes SQL and creates a corresponding table in the database.

Django admin :

In the last tutorial, we successfully made changes to the database by using Django models but how do we verify that the changes that we made are applied correctly or not? So, to do so we need to create a superuser in Django.

- Django admin or superuser is like the root user of Linux OS.
- Django admin is the most dominant user which means it has all powers to read, view, update, create and delete the data.

```
STATIC_URL= "/static/"
MEDIA_ROOT= os.path.join(BASE_DIR, "media")
MEDIA_URL="/media/"
```

1. **STATIC_URL**: It is simply the prefix of the URL that will be visible to the user while accessing the static files.

> Example: Suppose a user tries to access a static file named "mystatic.txt" of the shop app. Then, he or she will access the file at http://127.0.0.1:8000/static/shop/mystatic.txt.

2. **MEDIA_ROOT**: It is the path to the directory in which all the media files will be saved. Here, we have written(BASE_DIR, "media") which means that Django will create a folder named "media" under the base directory and all the media files will be saved in the "media" folder.
3. **MEDIA_URL**: Similar to the STATIC_URL, it is also the prefix of the URL that will be visible to the user while accessing the media files.