Introduction to Django

Presented by –

Lokendra Khedekar
0801IT211046
Naveen Kumar Jatav
0801IT211054
Harsh Raj Sharma
0801IT211034

Guided by –

Dr. Lalit Purohit Sir
Mr. Upendra Singh Sir

Agenda

What is Web Framework?

Web framework is a set of components designed to simplify your web development process. It has basic structuring tools in it, which serve as a solid base for your project. It allows you to focus on the most important details and project’s goals instead of creating things, that you can simply pull out of the framework.

What is Django?

01 Django is a web application framework written in Python programming language.

02 It is based on MVT (Model View Template) design pattern.

03 The Django is very demanding due to its rapid development feature.

04 It takes less time to build application after collecting client requirement.

05 This framework uses a famous tag line: The web framework for perfectionists with deadlines.

Version	Date	Description
0.90	16 Nov 2005	
0.91	11 Jan 2006	magic removal
0.96	23 Mar 2007	newforms, testing tools
1.0	3 Sep 2008	API stability, decoupled admin, unicode
1.1	29 Jul 2009	Aggregates, transaction based tests
1.2	17 May 2010	Multiple db connections, CSRF, model validation
1.3	23 Mar 2011	Timezones, in browser testing, app templates.
1.5	26 Feb 2013	Python 3 Support, configurable user model
1.6	6 Nov 2013	Dedicated to Malcolm Tredinnick, db transaction management, connection pooling.

1.7	**2 Sep 2014**	**Migrations, application loading and configuration.**
1.8 LTS	**2 Sep 2014**	**Migrations, application loading and configuration.**
1.8 LTS	**1 Apr 2015**	**Native support for multiple template engines.*Supported until at least April 2018***
1.9	**1 Dec 2015**	**Automatic password validation. New styling for admin interface.**
1.10	**1 Aug 2016**	**Full text search for PostgreSQL. New-style middleware.**
1.11 LTS	**1.11 LTS**	**Last version to support Python 2.7.*Supported until at least April 2020***
2.0	**Dec 2017**	**First Python 3-only release, Simplified URL routing syntax, Mobile friendly admin.**

Features of Django

1. Rapid Development
2. Scalable
3. Secure
4. Versatile
5. Open Source
6. Vast and Supported Community

Django Installation

To install Django, first visit to django official site (https://www.djangoproject.com) and download django by clicking on the download section. Here, we will see various options to download The Django.

Django requires pip to start installation. Pip is a package manager system which is used to install and manage packages written in python. For Python 3.4 and higher versions pip is used to manage packages.

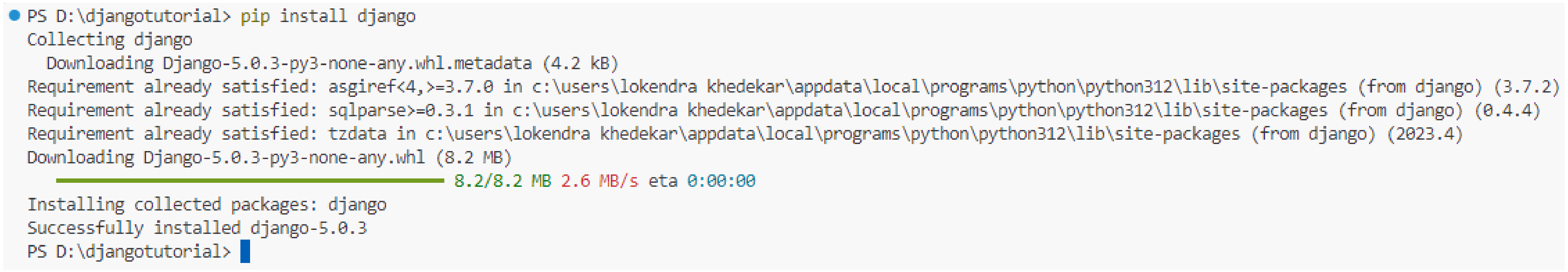
Django Installation

We are installing Django in Windows operating system.

The complete installation process is described below.
Before installing make sure pip is installed in local system.

Here, we are installing Django using pip3, the installation command is given below.

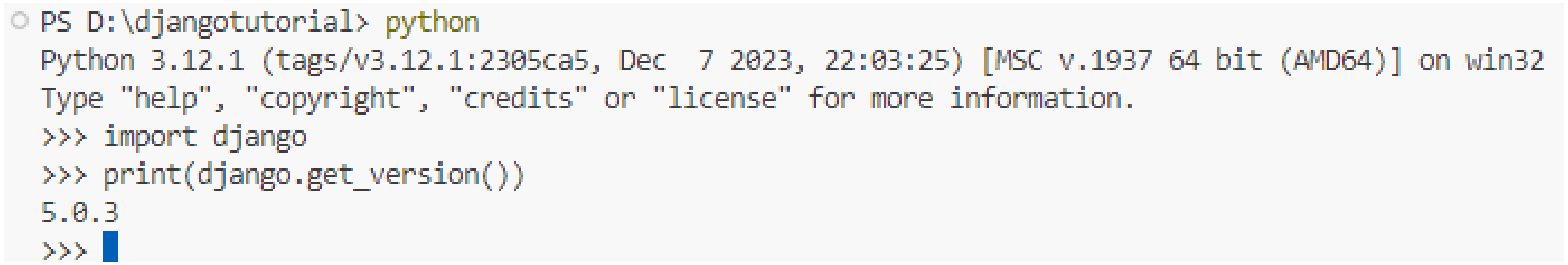
pip install django



```
PS D:\djangotutorial> pip install django
Collecting django
  Downloading Django-5.0.3-py3-none-any.whl.metadata (4.2 kB)
Requirement already satisfied: asgiref<4,>=3.7.0 in c:\users\lokendra khedekar\appdata\local\programs\python\python312\lib\site-packages (from django) (3.7.2)
Requirement already satisfied: sqlparse>=0.3.1 in c:\users\lokendra khedekar\appdata\local\programs\python\python312\lib\site-packages (from django) (0.4.4)
Requirement already satisfied: tzdata in c:\users\lokendra khedekar\appdata\local\programs\python\python312\lib\site-packages (from django) (2023.4)
Downloading Django-5.0.3-py3-none-any.whl (8.2 MB)
   ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 8.2/8.2 MB 2.6 MB/s eta 0:00:00
Installing collected packages: django
Successfully installed django-5.0.3
PS D:\djangotutorial>
```

Verify Django Installation

After installing Django, we need to verify the installation. Open terminal and write python3 and press enter. It will display python shell where we can verify django installation.



```
PS D:\djangotutorial> python
Python 3.12.1 (tags/v3.12.1:2305ca5, Dec  7 2023, 22:03:25) [MSC v.1937 64 bit (AMD64)] on win32
Type "help", "copyright", "credits" or "license" for more information.
>>> import django
>>> print(django.get_version())
5.0.3
>>> 
```

Django Project

A project is our website.

Projects contain the configuration information for our site.

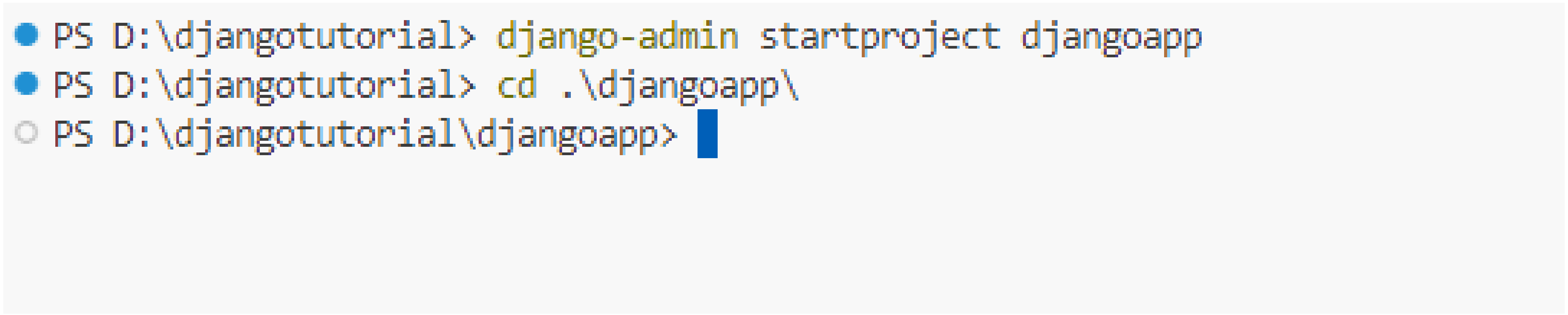
Create a project:
django-admin.py startproject [PROJECT_NAME]

Projects contain 3 files:
- settings.py - configuration information for your project
- urls.py - URL routes defined by your project
- manage.py - alias of django-admin.py tuned to your project

Django Project Example

Here, we are creating a project djangoapp in the current directory.

```
django-admin startproject djangoapp
```



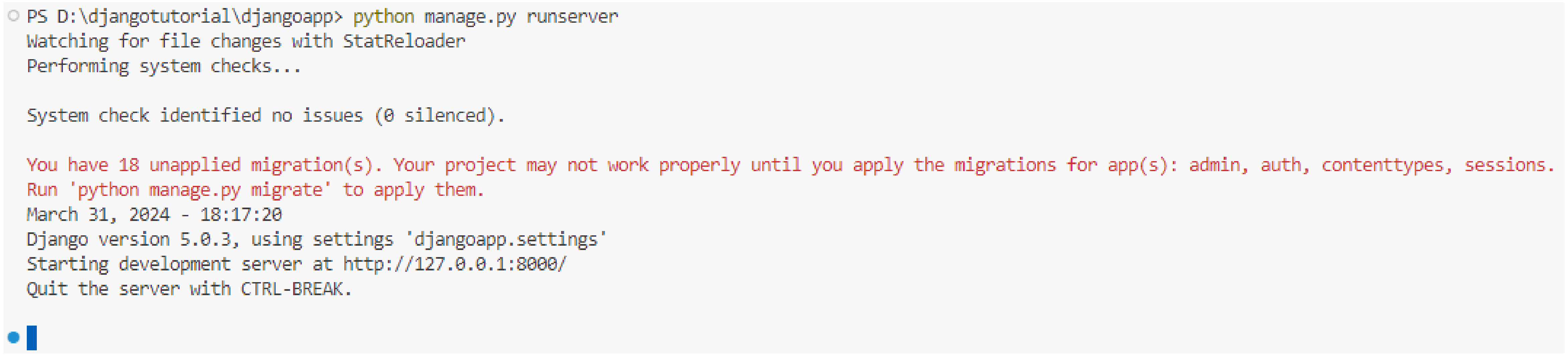
```
PS D:\djangotutorial> django-admin startproject djangoapp
PS D:\djangotutorial> cd .\djangoapp\
PS D:\djangotutorial\djangoapp>
```

Running the Django Project

Django project has a built-in development server which is used to run application instantly without any external web server. It means we don't need of Apache or another web server to run the application in development mode.

To run the application, we can use the following command.

```
python manage.py runserver
```

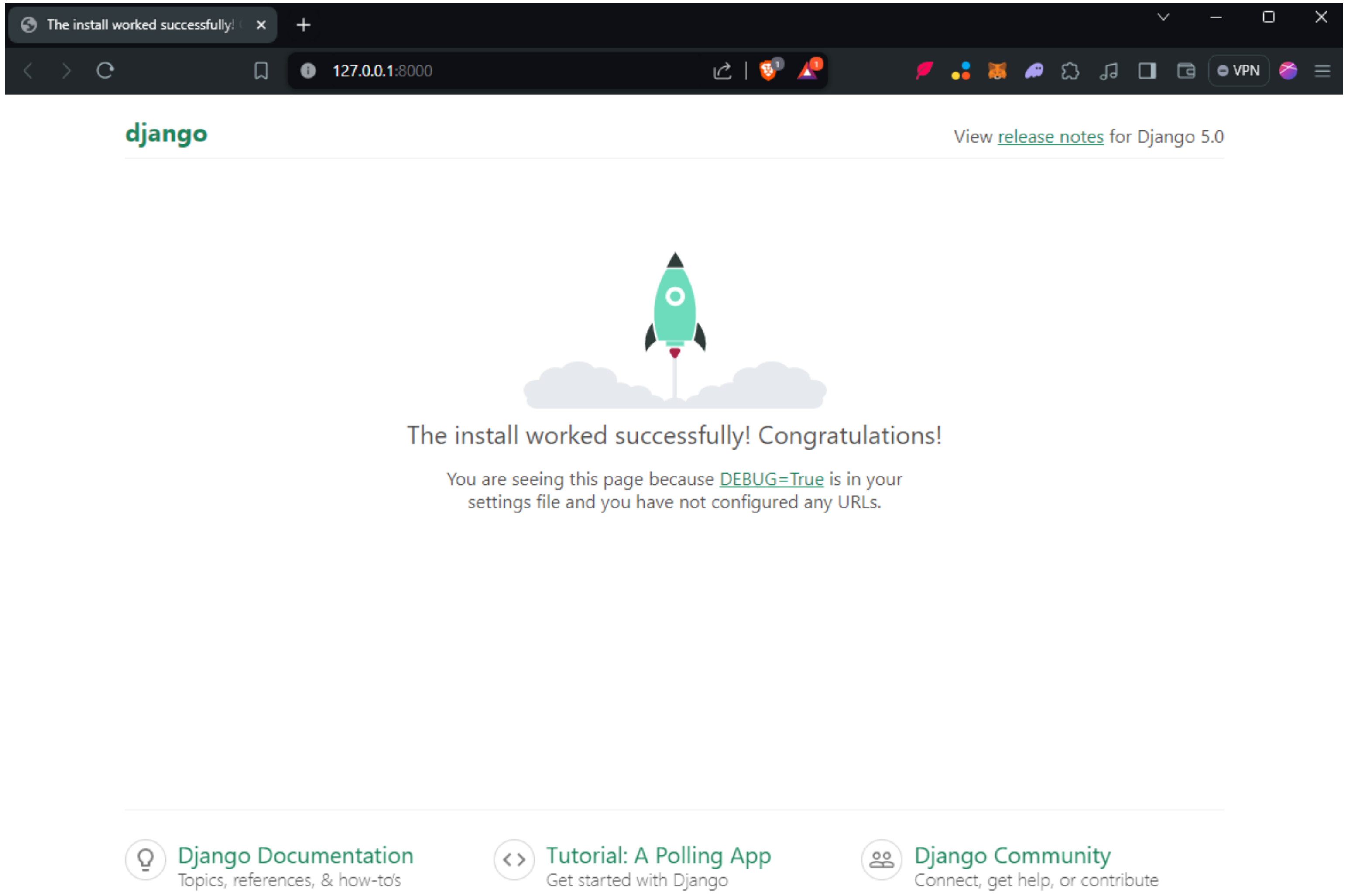


```
PS D:\djangotutorial\djangoapp> python manage.py runserver
Watching for file changes with StatReloader
Performing system checks...

System check identified no issues (0 silenced).

You have 18 unapplied migration(s). Your project may not work properly until you apply the migrations for app(s): admin, auth, contenttypes, sessions.
Run 'python manage.py migrate' to apply them.
March 31, 2024 - 18:17:20
Django version 5.0.3, using settings 'djangoapp.settings'
Starting development server at http://127.0.0.1:8000/
Quit the server with CTRL-BREAK.
```

The server has started and can be accessed at localhost with port 8000. Let's access it using the browser, it looks like the below.



View release notes for Django 5.0

The install worked successfully! Congratulations!

You are seeing this page because DEBUG=True is in your settings file and you have not configured any URLs.

Django Documentation
Topics, references, & how-to's

Tutorial: A Polling App
Get started with Django

Django Community
Connect, get help, or contribute

DJANGO MODELS AND DATABASE

Agenda

01 What is Model

02 Create First Model

03 Model Fields

04 Databases

What is Model?

- ❑ A model is the single, definitive source of information about your data.
- ❑ It contains the essential fields and behaviors of the data you’re storing
- ❑ Generally, each model maps to a single database table.
- ❑ Each model is a Python class that subclasses **django.db.models.Model**.
- ❑ Each attribute of the model represents a database field.

CREATE YOUR FIRST MODEL

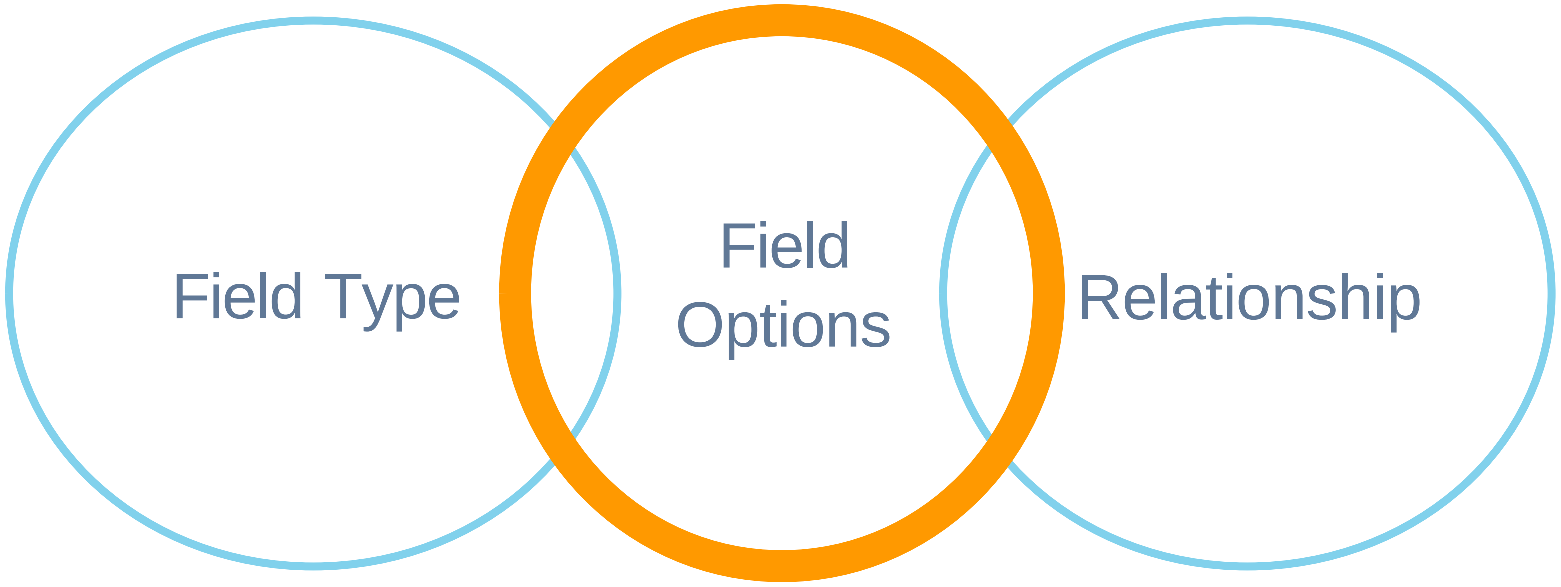
```
from django.db import models


class Person(models.Model):
    first_name = models.CharField(max_length=30)
    last_name = models.CharField(max_length=30)
```

MODEL FIELDS

Fields are organized into records, which contain all the information within the table relevant to a specific entity.

There are concepts to know before creating fields:



1. FIELD TYPE

The fields defined inside the Model class are the columns name of the mapped table

E.g.

AutoField()

An integer field that automatically increments

BooleanField()

Store true/false value and generally used for checkboxes

CharField()

A string field for small to large-sized strings.

DateField()

A date field represents python datetime. date instance.

2. FIELD OPTIONS

Field option are used to customize and put constraint on the table rows.

E.g.

```
name= models.CharField(max_length = 60)
```

here "max_length" specifies the size of the VARCHAR field.

The following are some common and mostly used field option:

01 **Null**

to store empty values as NULL in database.

02 **Blank**

if True, the field s allowed to be blank.

03 **default**

store default value for a field

04 **primary_key**

this field will be the primary key for the table

05 **unique_key**

puts unique key constraint for column.

3. MODEL FIELD RELATIONSHIP

The power of relational databases lies in relating tables to each other Django offers ways to define the three most common types of database relationships:

1. **many-to-one**
2. **many-to-many**
3. **one-to-one.**

1) Many-to-one relationships:

To define a many-to-one relationship, use django.db.models.ForeignKey.

You use it just like any other Field type: by including it as a class attribute of your model.

E.g.

```
class Manufacturer(models.Model)
    pass
class Car(models.Model):
    manufacturer = models.ForeignKey(Manufacturer,
on_delete=models.CASCADE)
```

2) Many-to-many relationships

To define a many-to-many relationship, use ManyToManyField. You use it just like any other Field type: by including it as a class attribute of your model.

For example, if a Pizza has multiple Topping objects – that is, a Topping can be on multiple pizzas and each Pizza has multiple toppings – here's how you'd represent that:

```
from django.db import models

class Topping(models.Model):
    # …
    pass

class Pizza(models.Model):
    # …
    toppings = models.ManyToManyField(Topping)
```

3) One-to-one relationships

To define a one-to-one relationship, use OneToOneField. You use it just like any other Field type: by including it as a class attribute of your model.

E.g.

```
from django.conf import settings
from django.db import models


class MySpecialUser(models.Model):
        user = models.OneToOneField(settings.AUTH_USER_MODEL)
        supervisor = models.OneToOneField(settings.AUTH_USER_MODEL)
```

Meta Option

- A metaclass is the class of a class.
- A class defines how an instance of the class behaves while a metaclass defines how a class behaves.
- A class is an instance of a metaclass.
- Give your model metadata by using an inner class Meta.

E.g.

```
from django.db import models

class Student(models.Model):
        name = models.CharField(max_length =50)


        class Meta:
                ordering =["name"]
                db_table = "students"
```

Databases

Django officially supports the following databases:

PostgreSQL

MariaDB

SQLite

MySQL®

ORACLE®
DATABASE

Telling Django About Your Database

Before we can create any models, we must first setup our database configuration. To do this, open the settings.py and locate the dictionary called DATABASES.

modify the default key/value pair so it looks something like the following example.

```
DATABASES = {
   'default': {
      'ENGINE': 'django.db.backends.sqlite3',
      'NAME': DATABASE_PATH,
   }
}
```

Also create a new variable called "DATABASE_PATH" and add that to the top of your settings.py

```
DATABASE_PATH = os.path.join(PROJECT_PATH, 'rango.db')
```

Views

- **Perform a task and render output (HTML/JSON/XML)**
- **Python function that takes an HttpRequest and returns an HttpResponse**
- **Defined in an app's views.py file**

View Example:-

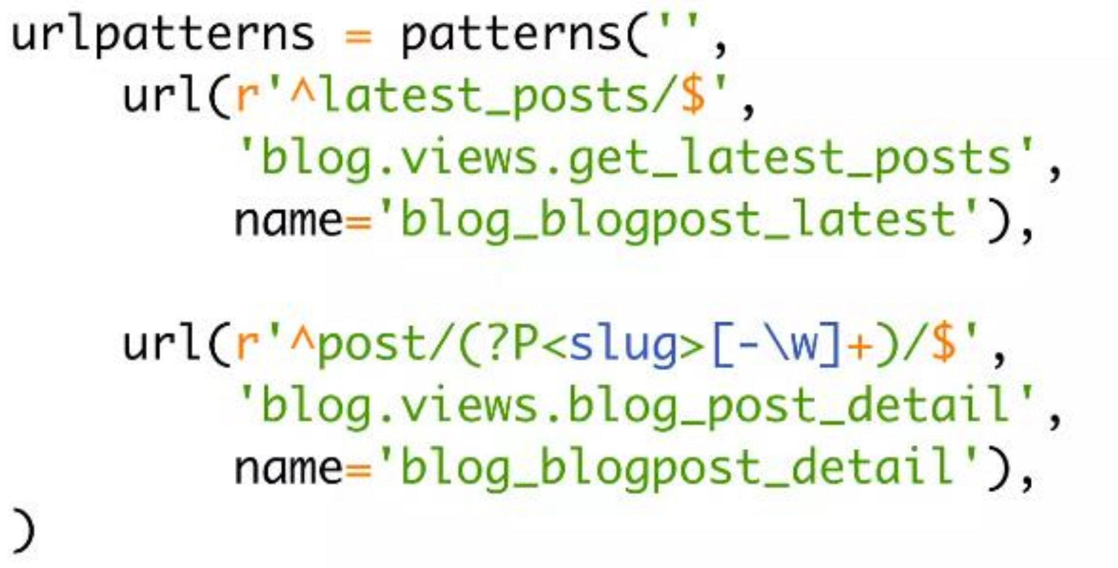
```
from django.http import HttpResponse

def hello_world(request):
        return HttpResponse("Hello World“)
```

URL Routes

- **Defines the URLs used in your project**
- **Maps a URL to a View Function**
- **Defined using Regular Expressions**
- **Defined in an app's urls.py file.**

URL Example:-



```
urlpatterns = patterns('',
    url(r'^latest_posts/$',
        'blog.views.get_latest_posts',
        name='blog_blogpost_latest'),

    url(r'^post/(?P<slug>[-\w]+)/$',
        'blog.views.blog_post_detail',
        name='blog_blogpost_detail'),
)
```

Templates

- Describes the Presentation of your data
- Separates Logic from Presentation
- Simple Syntax & Designer Friendly
- Supports Reuse through inheritance & inclusion

Template Syntax :-

Variables: {{variable-name}}

Tags: Perform logic
{% include "_form.html" %}

Filters: Operate on data
{{post.publish_date|date:"m/d/Y"}}

Thank You!