PROJECT 10 (Render from field manually)

Copy the project 9 require some change

Step1: change in HTML code:

Render Form Field Manually

Each field is available as an attribute of the form using {{form.name_of_field}}

{{ field.label }} - The label of the field.

Example:- {{ form.name.label }}

{{ field.label_tag }} - The field's label wrapped in the appropriate HTML <label> tag. This includes the form's label_suffix. The default label_suffix is a colon:

Example:- {{ form.name.label_tag }}

{{ field.id_for_label }} - The ID that will be used for this field.

Example:- {{ form.name.id_for_label }}

{{ field.value }} - The value of the field.

Example:- {{ form.name.value }}

{{ field.html_name }} - The name of the field that will be used in the input element's name field. This takes the form prefix into account, if it has been set.

Example:- {{ form.name.html_name }}

{{ field.help_text }} - Any help text that has been associated with the field.

Example:- {{ form.name.help_text }}

{{ field.field }} - The Field instance from the form class that this BoundField wraps. You can use it to access Field attributes.

Example:- {{form.name.field.help_text}}

{{ field.is_hidden }} - This attribute is True if the form field is a hidden field and False otherwise. It's not particularly useful as a template variable, but could be useful in conditional tests such as:

```
{% if field.is_hidden %}

   {# Do something special #}

{% endif %}
```

```html
<!DOCTYPE html>
<html lang="en">
<head>
 <meta charset="UTF-8">
 <meta name="viewport" content="width=device-width, initial-scale=1.0">
 <title>User Registration</title>
</head>
<body>
 <form action="" method="get">
   <div>
    <label for="id_name">Name:</label>
    <input type="text" name="name" required id="id_name">
   </div>
   <div>
    <label for="{{form.name.id_for_label}}">Name:</label>
    {{form.name}}
   </div>
   <div>
    {{form.name.label_tag}}
    {{form.name}}
   </div>
   <div>
    {{form.name.label}} <br>
    {{form.name.value}} <br>
    {{form.name.html_name}} <br>
    {{form.name.help_text}} <br>
    {{form.name.field.required}} <br>
   </div>
 </form>
</body>
</html>
```

Step-2: Change in views.py code and output:

Views.py

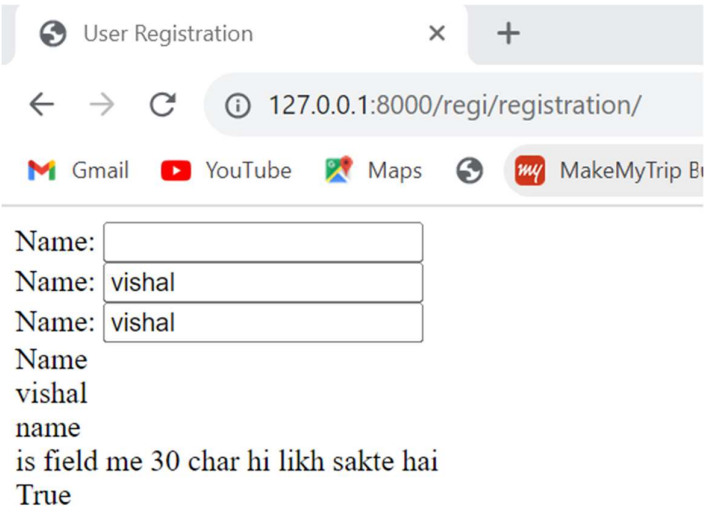
```python
from django.shortcuts import render
from .forms import StudentRegistration
# Create your views here.

def showformdata(request):
 fm = StudentRegistration()

 return render(request, 'roll/userregistration.html', {'form':fm})
```

step-3:

change in forms.py

```
from django import forms
class StudentRegistration(forms.Form):
 name = forms.CharField(initial="vishal", help_text="is field me 30 char hi likh sakte
hai")
 email = forms.EmailField()
```

output:



Note:

A form's fields are themselves classes; they manage form data and perform validation when a form is submitted.

Syntax:- FieldType(**kwargs)

Example:-

IntegerField()

CharField(required)

CharField(required, widget=forms.PasswordInput)

from django import forms

class StudentRegistration(forms.Form):

 name = forms.CharField()

required - It take True or False value. By default, each Field class assumes the required value is True.

label - The label argument lets you specify the “human-friendly” label for the field. This is used when the Field is displayed in a Form.

label_suffix - The label_suffix argument lets you override the form’s label_suffix on a per-field basis.

initial - The initial argument lets you specify the initial value to use when rendering this Field in an unbound Form.

disabled - The disabled boolean argument, when set to True, disables a form field using the disabled HTML attribute so that it won’t be editable by users. Even if a user tampers with the field’s value submitted to the server, it will be ignored in favor of the value from the form’s initial data.

required - It take True or False value. By default, each Field class assumes the required value is True.

label - The label argument lets you specify the “human-friendly” label for the field. This is used when the Field is displayed in a Form.

label_suffix - The label_suffix argument lets you override the form’s label_suffix on a per-field basis.

initial - The initial argument lets you specify the initial value to use when rendering this Field in an unbound Form.

disabled - The disabled boolean argument, when set to True, disables a form field using the disabled HTML attribute so that it won’t be editable by users. Even if a user tampers with the field’s value submitted to the server, it will be ignored in favor of the value from the form’s initial data.

Widget

The widget handles the rendering of the HTML, and the extraction of data from a GET/POST dictionary that corresponds to the widget.

The HTML generated by the built-in widgets uses HTML5 syntax, targeting <!DOCTYPE html>.

Whenever you specify a field on a form, Django will use a default widget that is appropriate to the type of data that is to be displayed.

Each field type has an appropriate default Widget class, but these can be overridden as required.

Form fields deal with the logic of input validation and are used directly in templates.

Widgets deal with rendering of HTML form input elements on the web page and extraction of raw submitted data.

Example:-

TextInput

Textarea

attrs - A dictionary containing HTML attributes to be set on the rendered widget.

If you assign a value of True or False to an attribute, it will be rendered as an HTML5 boolean attribute.

Example:-

```
feedback=forms.CharField(widget=forms.TextInput(attrs={'class':'somecss1 somecss2',
'id':'uniqueid'}))
```

GKTCS INNOVATIONS

Introduction to Django

Agenda

GKTCS INNOVATIONS

01 What is a Web Framework?

02 What is a Django?

03 History of Django

04 Features of Django

05 Django Installation

What is Web Framework?

GKTCS INNOVATIONS

“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?

GKTCS INNOVATIONS

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.

History

GKTCS INNOVATIONS

2003 — Django was design and developed by Lawrence journal world.

2005 — Publicly released under BSD license.

2008 — 1.0 version is launched

2017 — 2.0 version is launched

2018 — Its current stable version 2.0.3 is launched.

GKTCS INNOVATIONS

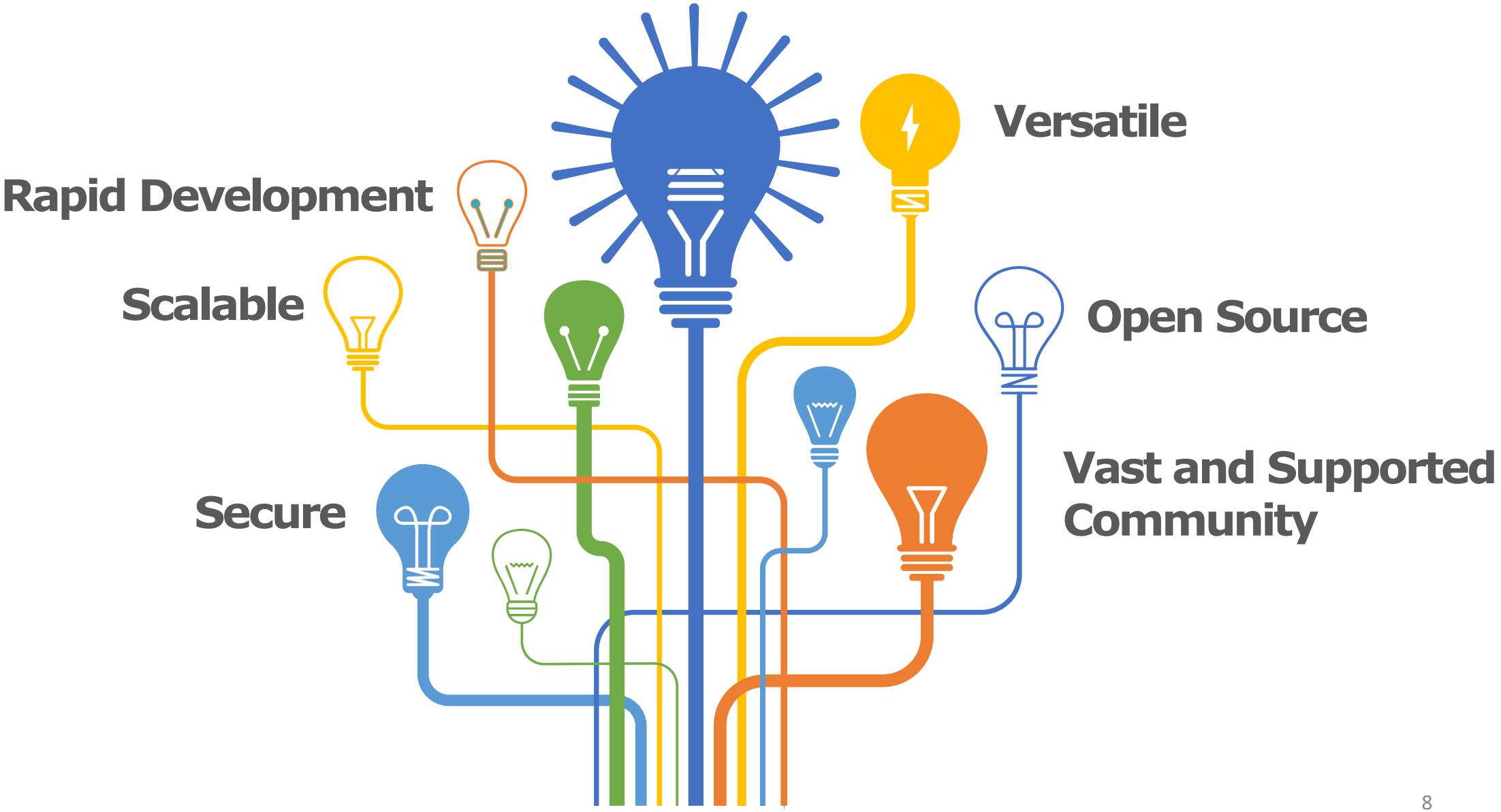
Version	Date	Description
0.90	16 Nov 2005	
0.91	11 Jan 2006	magic removal
0.96	23 Mar 2007	newforms, testing tools
1.0	3 Sep 2008	API stability, decoupled admin, unicode
1.1	29 Jul 2009	Aggregates, transaction based tests
1.2	17 May 2010	Multiple db connections, CSRF, model validation
1.3	23 Mar 2011	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.

GKTCS INNOVATIONS

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.**

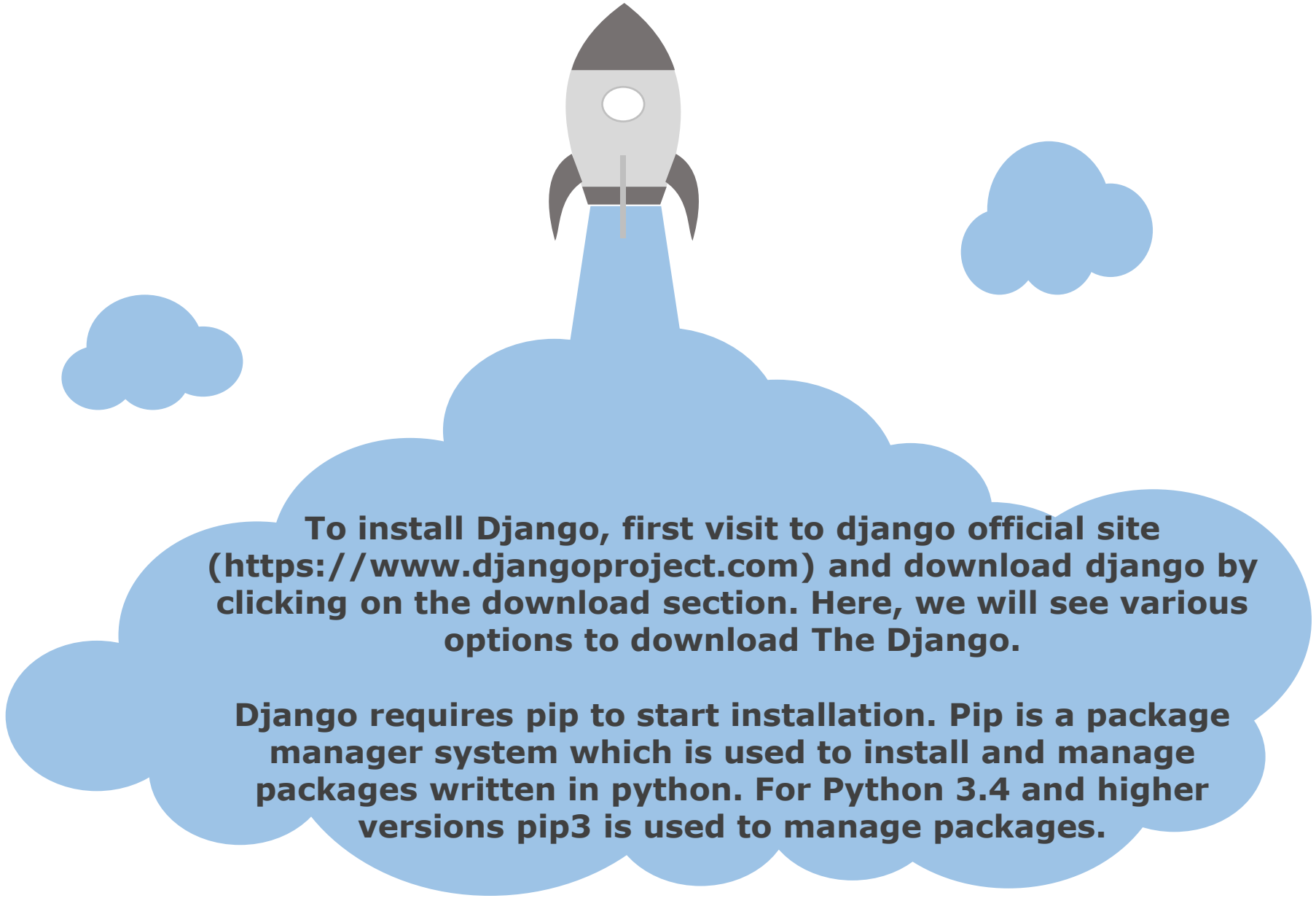
Features of Django

GKTCS INNOVATIONS



Django Installation

GKTCS INNOVATIONS

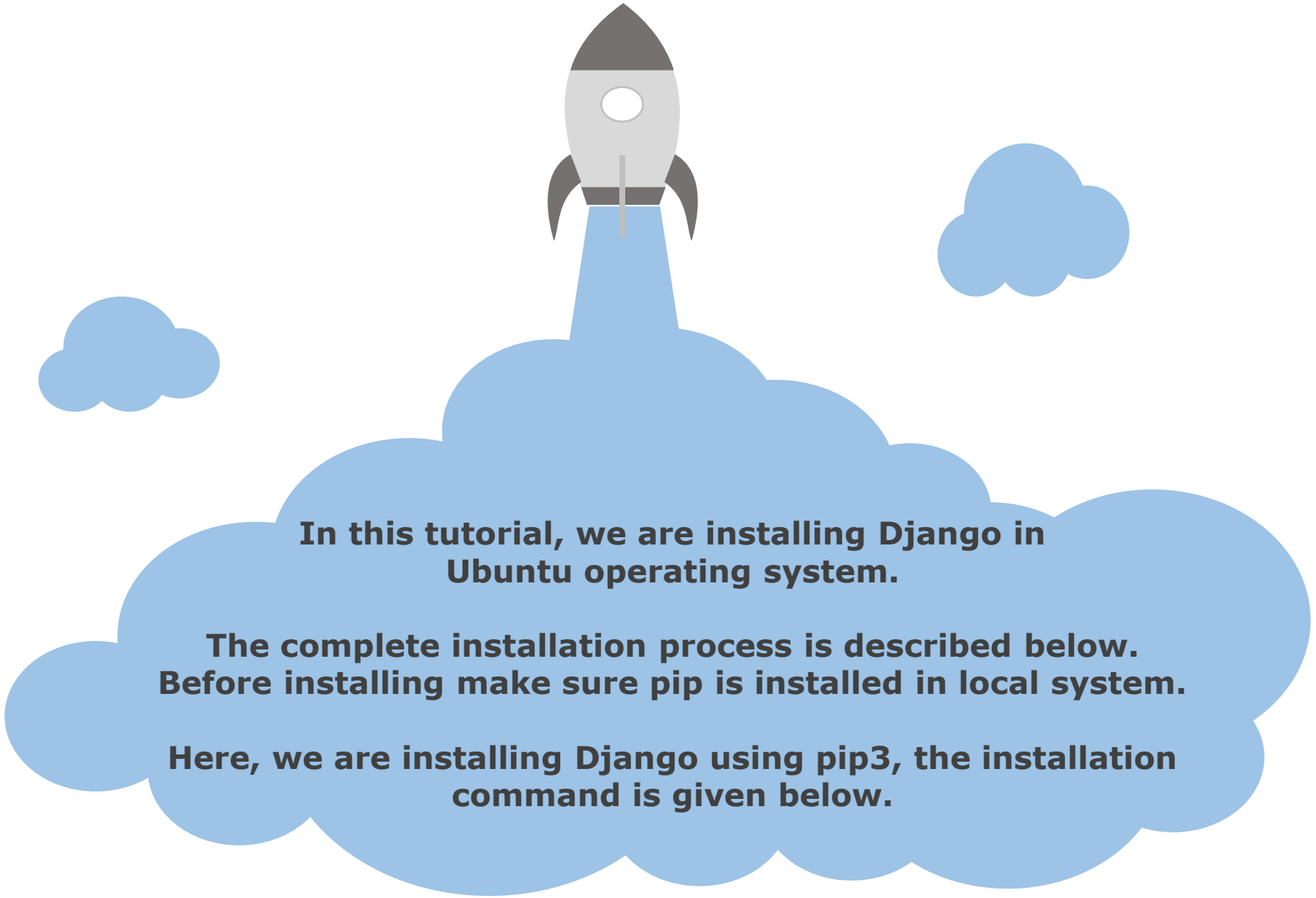


To install Django, first visit to django official site (https://www.djangoproject.com) and download django by clicking on the download section. Here, we will see various options to download The Django.

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 pip3 is used to manage packages.

Django Installation

GKTCS INNOVATIONS



In this tutorial, we are installing Django in Ubuntu 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.

GKTCS INNOVATIONS

$ pip3 install django==2.0.3

```
Terminal File Edit View Search Terminal Help
root@sssit-Inspiron-15-3567:/home/sssit# pip3 install django==2.0.3
Collecting django==2.0.3
  Using cached Django-2.0.3-py3-none-any.whl
Requirement already satisfied: pytz in /usr/local/lib/python3.5/dist-packages (f
rom django==2.0.3)
Installing collected packages: django
Successfully installed django-2.0.3
root@sssit-Inspiron-15-3567:/home/sssit#
```

Verify Django Installation

GKTCS INNOVATIONS

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.

```
Terminal File Edit View Search Terminal Help
root@sssit-Inspiron-15-3567:/home/sssit# python3
Python 3.5.2 (default, Nov 23 2017, 16:37:01)
[GCC 5.4.0 20160609] on linux
Type "help", "copyright", "credits" or "license" for more information.
>>> import django
>>> print(django.get_version())
2.0
>>>
```

Django Project

GKTCS INNOVATIONS

In the previous topic, we have installed Django successfully. Now, we will learn step by step process to create a Django application.

Django Project Example

GKTCS INNOVATIONS

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

```
$ django-admin startproject djangpapp
```

```
Terminal File Edit View Search Terminal Help
root@sssit-Inspiron-15-3567:/home/sssit# django-admin startproject djangpapp
root@sssit-Inspiron-15-3567:/home/sssit#
```

Locate into the Project

GKTCS INNOVATIONS

Now, move to the project by changing the directory. The Directory can be changed by using the following command.

cd djangpapp

```
Terminal File Edit View Search Terminal Help
root@sssit-Inspiron-15-3567:/home/sssit# django-admin startproject djangpapp
root@sssit-Inspiron-15-3567:/home/sssit# cd djangpapp/
root@sssit-Inspiron-15-3567:/home/sssit/djangpapp# ls
djangpapp  manage.py
root@sssit-Inspiron-15-3567:/home/sssit/djangpapp#
```

GKTCS INNOVATIONS

To see all the files and subfolders of django project, we can use tree command to view the tree structure of the application. This is a utility command, if it is not present, can be downloaded via apt-get install tree command.

```
Terminal File Edit View Search Terminal Help
root@sssit-Inspiron-15-3567:/home/sssit# cd djangpapp/
root@sssit-Inspiron-15-3567:/home/sssit/djangpapp# ls
djangpapp  manage.py
root@sssit-Inspiron-15-3567:/home/sssit/djangpapp# tree
.
├── djangpapp
│   ├── __init__.py
│   ├── settings.py
│   ├── urls.py
│   └── wsgi.py
└── manage.py

1 directory, 5 files
root@sssit-Inspiron-15-3567:/home/sssit/djangpapp#
```

Running the Django Project

GKTCS INNOVATIONS

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.

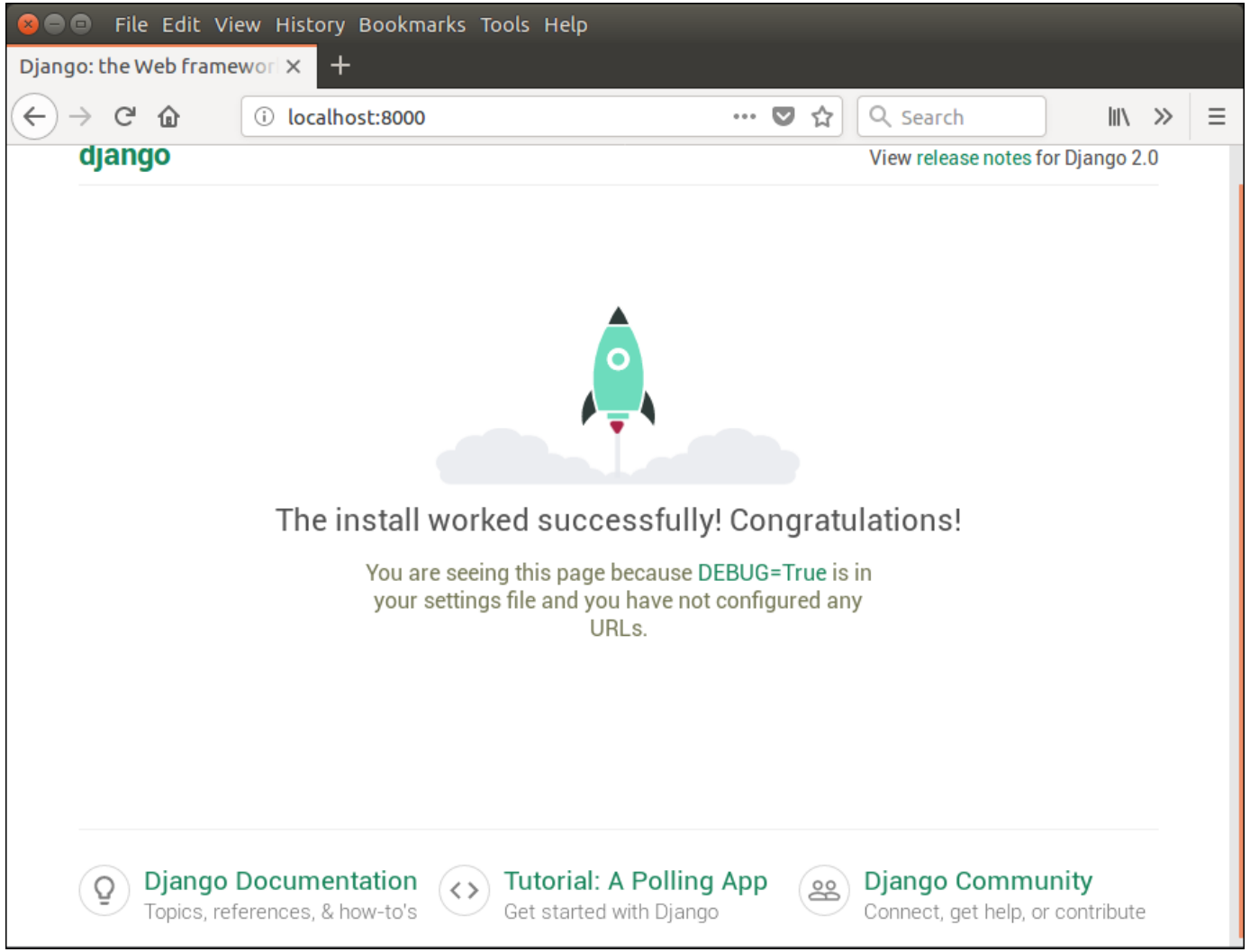
```
$ python3 manage.py runserver
```

GKTCS INNOVATIONS

```
Terminal File Edit View Search Terminal Help
root@sssit-Inspiron-15-3567:/home/sssit/djangpapp# python3 manage.py runserver
Performing system checks...

System check identified no issues (0 silenced).

You have 14 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 13, 2018 - 07:21:03
Django version 2.0, using settings 'djangpapp.settings'
Starting development server at http://127.0.0.1:8000/
Quit the server with CONTROL-C.
```

GKTCS INNOVATIONS

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



GKTCS INNOVATIONS

DJANGO MODELS AND DATABASE

Agenda

GKTCS INNOVATIONS

01	What is Model
02	Create First Model
03	Model Fields
04	Databases

What is Model?

GKTCS INNOVATIONS

- 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

GKTCS INNOVATIONS

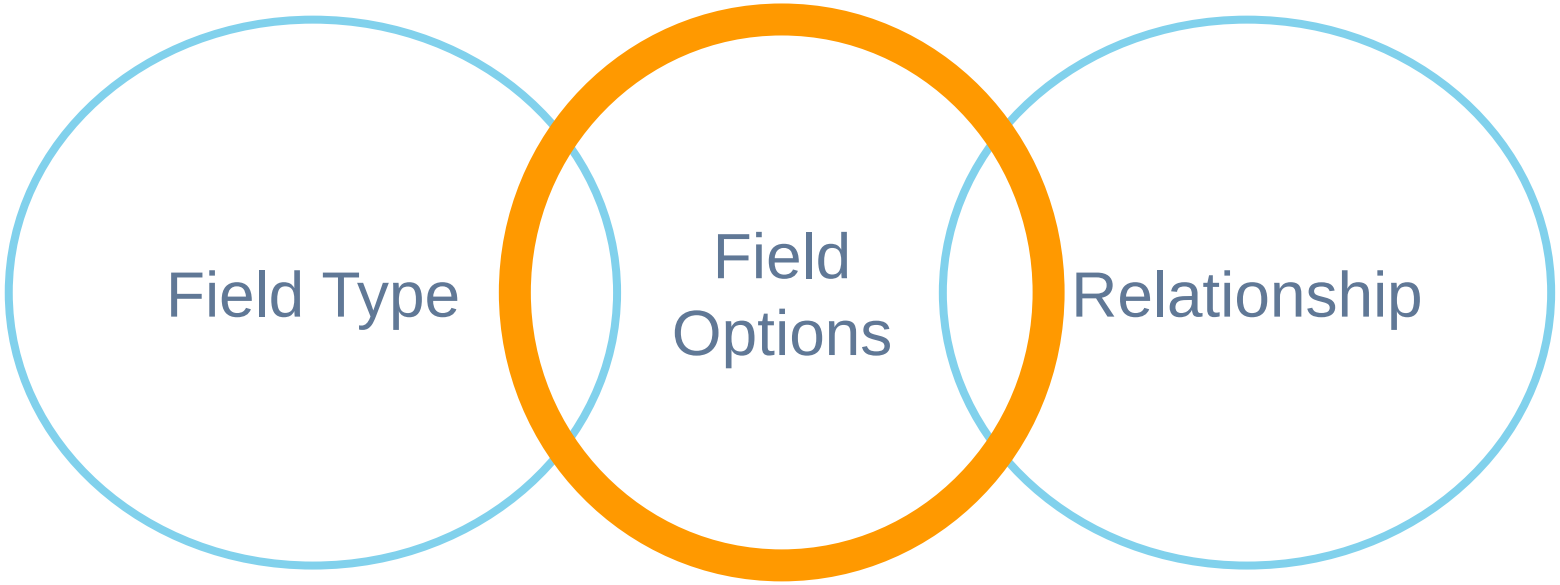
```
from django.db import models

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

MODEL FIELDS

GKTCS INNOVATIONS

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

GKTCS INNOVATIONS

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

GKTCS INNOVATIONS

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.

GKTCS INNOVATIONS

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

GKTCS INNOVATIONS

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.**

GKTCS INNOVATIONS

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)
```

GKTCS INNOVATIONS

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:

GKTCS INNOVATIONS

```
from django.db import models

class Topping(models.Model):
    # ...
    pass

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

GKTCS INNOVATIONS

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

GKTCS INNOVATIONS

- 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.

GKTCS INNOVATIONS

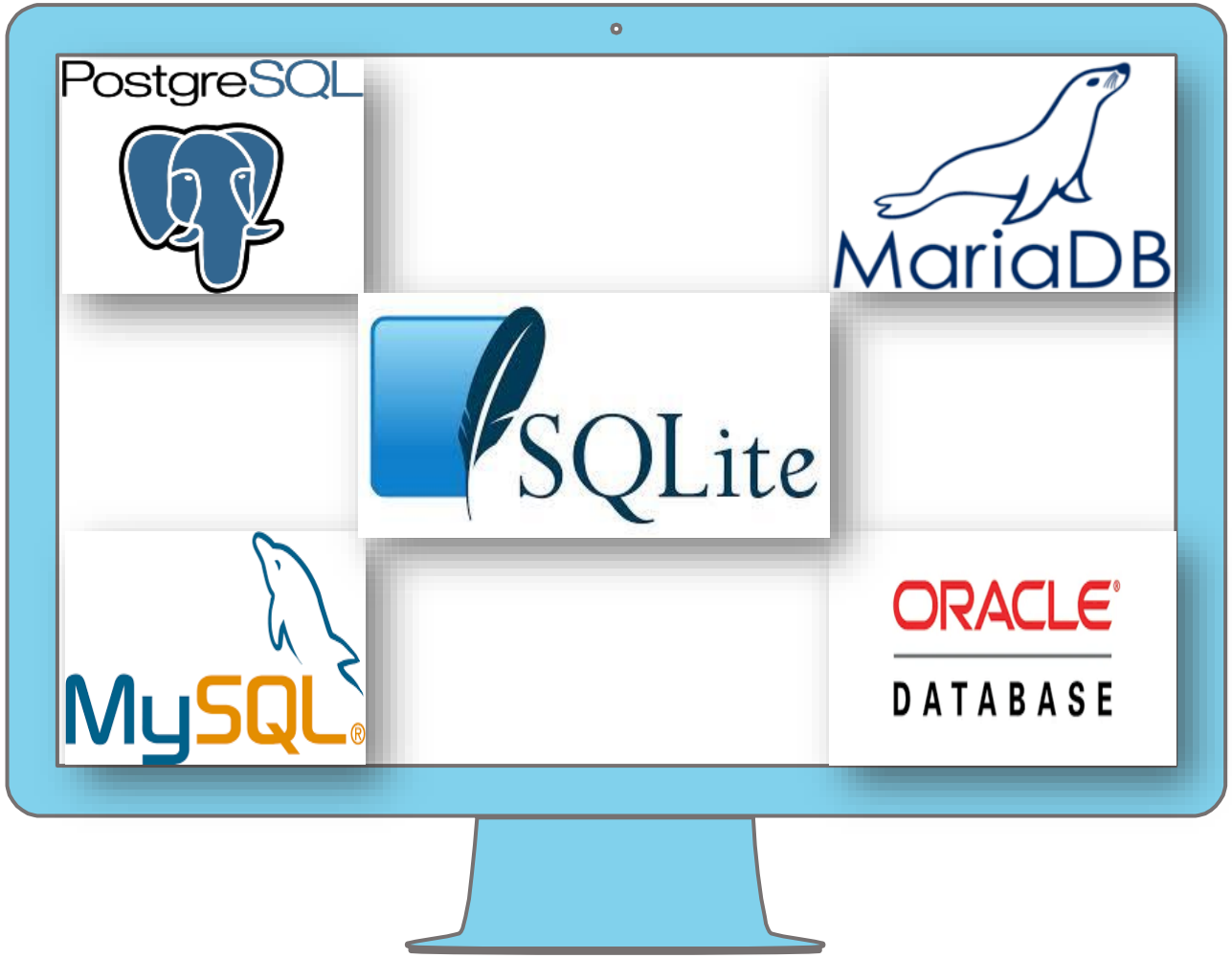
E.g.

```
from django.db import models

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

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

Databases

GKTCS INNOVATIONS

Django officially supports the following databases:



Telling Django About Your Database

GKTCS INNOVATIONS

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,
    }
}
```

GKTCS INNOVATIONS

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')
```