Installation

Python	https://www.python.org/downloads/
pip	https://pip.pypa.io/en/latest/installing/
Django	pip install Django

C'est quoi, Django?

Django est un **framework Web**. Parce que Django a été développé dans un environnement journalistique évoluant très rapidement, il a été conçu pour rendre le développement Web **rapide et facile**.

Son slogan en est la preuve : "Django: The Web framework for perfectionists with deadlines" .

Django s'inspire fortement de l'architecture MVC. On dit que Django est en **MTV** pour **Modèles, Templates et Vues**.

Django se **compose de modules appelés "applications"** qui peuvent être inclus ou retirés facilement. Grossièrement , dans un site conçu avec Django, la gestion des utilisateurs est une app et la chatbox en est une autre.

Django est open source, distribué sous **licence BSD 3-clauses** et developpé et maintenu par la Django Software Foundation.

Déploiement :

Attention : Django fournit un serveur web très simple pour déboguer et tester rapidement. Il ne doit jamais être utilisé en production car il n'a jamais passé d'audit de sécurité et ne supportera pas la charge de plusieurs utilisateurs.

git clone votreprojet

cd votreprojet

./manage.py check --deploy

./manage.py migrate

./manage.py createsuperuser

Configurer le fichier WSGI

lancer votre serveur en prod (apache, nginx etc...)

et se rendre à la page d'acceuil de votre site !

Avantages et Incovénients :

Accès à la librairie standard Python.	Pas d'AJAX.
Développement simple, efficace et rapide.	ORM capricieuse.
Interface d'Administration et autres générés automatiquement à partir des modèles.	Evolution lente.
Déboguage et tests.	
Communauté et Documentation.	

Modèles :

Le modèle est la seule et unique source d'information concernant les données. Il contient les attributs et le comportement des données stockées.

Chaque modèle correspond à une seule table dans la base de données.

Chaque modèle est une classe Python qui dérive de django.db.models.Model.

Chaque attribut du modèle représente une colonne de la table.

Avec tout ça, une API d'accès aux données est automatiquement générée par le cadre compatible CRUD. Inutile d'écrire des requêtes SQL associées à des formulaires, elles sont générées automatiquement par l'ORM.

Modèle exemples :

Dans cet exemple on définit le modèle Personne avec son prénom et son nom de famille

```
from django.db import models
class Personne(models.Model):
prenom= models.CharField(max_length=30)
nomdefamille= models.CharField(max_length=30)
```

Le modèle définit au-dessus créera une table de ce style:

```
CREATE TABLE myapp_personne (
"id" SERIAL NOT NULL PRIMARY KEY,
"prenom" VARCHAR(30) NOT NULL,
"nom de famille" VARCHAR(30) NOT NULL
);
```

Les fonctions qui agissent sur le modèle doivent se trouver avec le modèle dans models.py:

```
def personne_agee(self):
"Retourne un indicateur de vieillesse."
import datetime
if self.birth_date < datetime.date(1945, 8, 1):
return "salut l'antiquité"
else:
return "salut la jeunesse"
```

Templates:

Etant un framework Web, Django a besoin d'une méthode pratique pour générer du HTML dynamiquement. La solution la plus commune pour ce faire viennent de l'utilisation des templates.

Un template contient les parties statiques de la sortie HTML désirée mais aussi d'autres lignes suivant une syntaxe spécifique décrivant comment le contenu dynamique sera inséré.

Un projet Django peut être configuré avec une ou plusieurs moteurs de template.

Par défaut, Django utilise son propre système suivant le Django Template Language (DTL).

Un template Django est donc un fichier qui contient du texte répondant à une certaine syntaxe.

Par convention on les place dans le dossier mon_app/templates/mon_app

Template exemples :

Variables {{ a }}

Une variable a pour sortie la valeur du contexte qui est un dictionnaire liant un ou plusieurs couples clef-valeur.

```
Mon prénom est {{ prenom }}.
```

Tags {% a %}

```
{% csrf_token %}
```

Filtres {{ a|filtre }}

```
{{ django|title }}
```

Avec {'django': 'the web framework for perfectionists with deadlines'}, cette template affiche:
The Web Framework For Perfectionists With Deadlines
-> *la première lettre de chaque mot devient une majuscule*

Un commentaire ressemble à ça :

```
{# ça ne sera pas affiché à l'écran #}
```

Vues :

Une fonction de vue, ou une vue pour faire court, est simplement une fonction Python qui prend une requête Web et retourne une réponse Web.

Cette réponse peut être le contenu HTML d'une page,une redirection, une erreur 404, un document XML, une image etc...

La vue contient donc la logique nécessaire pour retourner cette réponse. Par convention, on mettra les vues dans le fichier views.py .

Vue exemple :

Voilà une vue qui retourne la date et le temps actuel sous forme d'un document HTML :

```
from django.http import HttpResponse
import datetime
def current_datetime(request):
now = datetime.datetime.now()
html = "<html><body>It is now %s.</body></html>" % now
return HttpResponse(html)
```

Routage (URL Dispatcher) :

Routage du projet vers l'app

mon_projet/mon_projet/urls.py

```
#Include mon_projet/mon_app/urls.py urlpatterns =
patterns('', url(r'^mainApp/',
include('mainApp.urls')),)
```

Routage de l'app vers la vue

mon_projet/mon_app/urls.py

```
from django.conf.urls import url, patterns from
mainApp import views urlpatterns = patterns('',
url(r'^$', views.index, name='index'),)
```

Ressources :

Site du projet	https://www.djangoproject.com
Snippets	https://djangosnippets.org/
Apps	https://djangopackages.org/
Docs	https://docs.djangoproject.com/fr/1.11/