Django
هو إطار عمل ويب
Python
سيسمح بكتابة كود ويب
Python
الذي سيمكن
HTML و CSS
المولدين ديناميكيًا ويسمح ببناء تطبيق ويب ديناميكي

Django

Django is a Python web framework which is going to allow writing Python web code that will enable dynamically generating HTML and CSS and allow building a dynamic web application

> ### BEFORE YOU BEGIN
>
> Each great journey begins with a humble step. Before we can do awesome things with Django, we need to be prepared with a productive environment. In this section, we will see how to do that.
>
> ### INSTALLING PYTHON
>
> Before using Django version 3 or later, you will need Python 3 installed on your computer. Mac and Linux operating systems usually have some version of Python installed, but it's best to make sure you're running the latest version. On Mac, for Homebrew users, you can just type the following:
>
> ```
> $ brew install python
> ```
>
> On Debian-based Linux distributions, you can check which version is available by typing the following:
>
> ```
> $ apt search python3
> ```
>
> Depending on the output, you can then type something along the lines of the following:
>
> ```
> $ sudo apt install python3 python3-pip
> ```

> For Windows, you can download the Python 3 installer here: https://www.python.org/downloads/windows/. Once you have the installer, click on it to run, and then follow the instructions. Be sure to select the **Add Python 3.x to PATH** option.
>
> Once installed, from Command Prompt, you can run **`python`** to launch a Python interpreter.
>
> Note that on macOS and Linux, depending on your configuration, the **`python`** command might launch Python version 2 or Python version 3. To be sure, make sure to specify **`python3`**. On Windows, you should just run **`python`** as this will always launch Python version 3.
>
> Similarly with the **`pip`** command. On macOS and Linux, specify **`pip3`**; on Windows, just **`pip`**.

Django
هو إطار عمل ويب ومجموعة من الأدوات

Django is a web framework and a set of tools

- Installing Django, using pip: Python package manager using the following command pip3 install Django: running this command will install Django on your system
- Run a command to create Django project

```
django-admin startproject PROJECT_NAME
```

When running this command Django will create some files for this project

There are couple of files, the most important file is manage.py نستخدمها لتنفيذ الأوامر في مشروع ديجانقو

- Manage.py: we use to execute commands on Django project
- Settings.py: contains important configuration settings for Django application, it is pre-loaded with default settings and we may change these settings to add features to the application or add modifications to how the application behaves

يحتوي على إعدادات تكوين مهمة لتطبيق ديجانقو ، وهو محمل مسبقًا بالإعدادات الافتراضية وقد نقوم بتغيير هذه الإعدادات لإضافة ميزات للتطبيق أو إضافة تعديلات على كيفية تصرف التطبيق

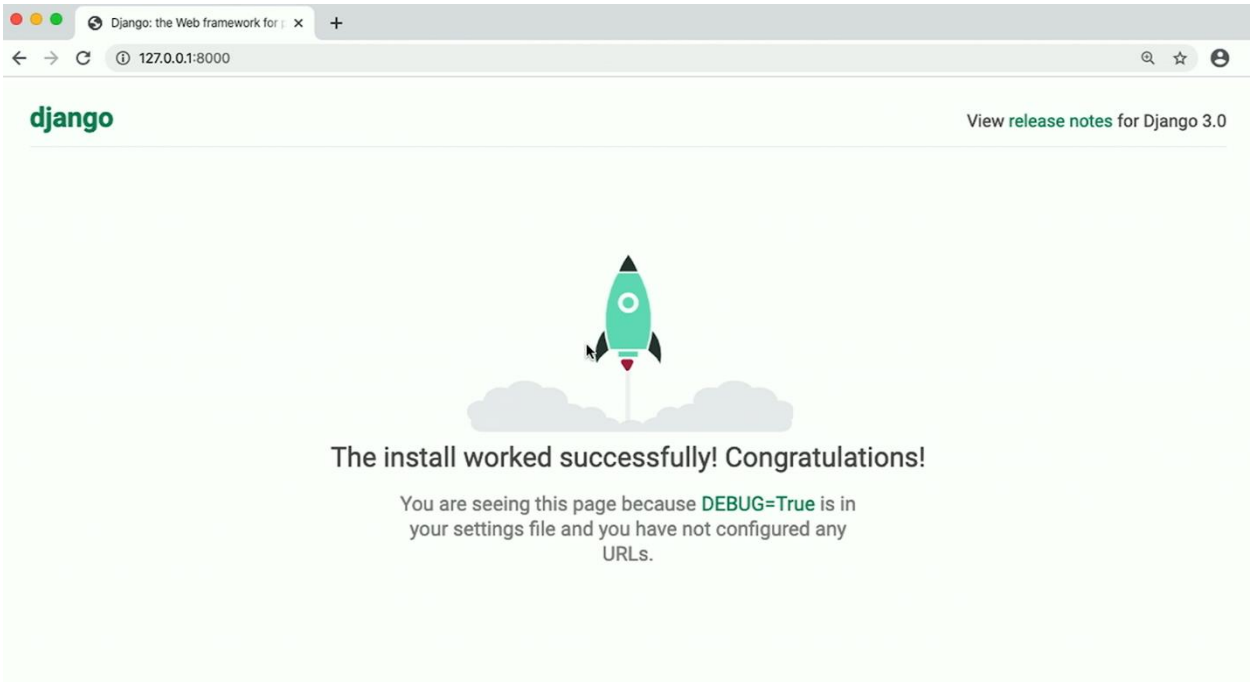
- Urls.py: it is like a table of contents for all the URLS that the web application that can be visited إنه يشبه جدول محتويات جميع عناوين URL التي يمكن زيارتها من خلال تطبيق الويب

To start the web application on the server, use the following command:

python manage.py runserver

```
workspace@Brian-MBP lecture3 % python manage.py runserver
Watching for file changes with StatReloader
Performing system checks...

System check identified no issues (0 silenced).

You have 17 unapplied migration(s). Your project may not work properly until you apply
admin, auth, contenttypes, sessions.
Run 'python manage.py migrate' to apply them.

January 28, 2020 - 16:18:33
Django version 3.0.2, using settings 'lecture3.settings'
Starting development server at http://127.0.0.1:8000/
Quit the server with CONTROL-C.
```

http://127.0.0.1:8000/: this URL means where the web application is currently running , in this case the web application is running on the local computer and the port number is 8000

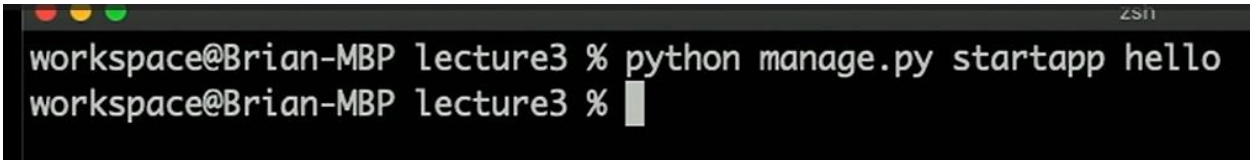


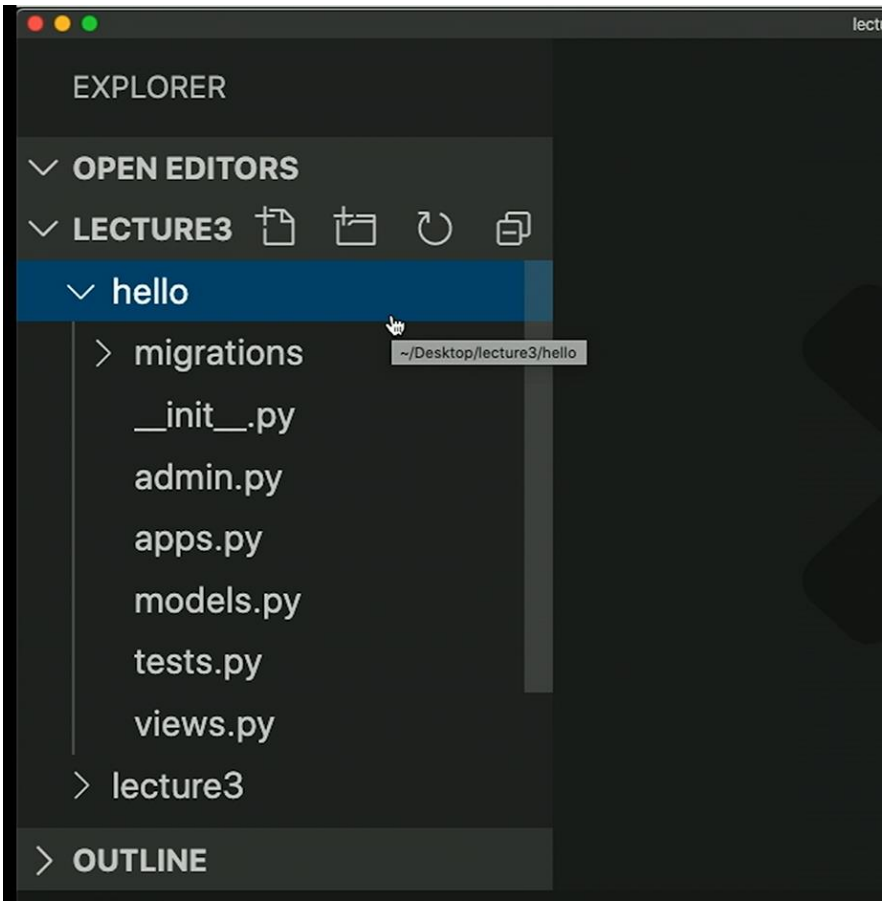
This is what is called Django project which could contain multiple applications

After creating a project, we start by creating app لدعم خدمات منفصلة في إطار مشروع كبير
Every Django project has multiple applications, to support separate services under a large project like Google, Amazon, … etc.

CTRL+C to exit the server

Creating Django application by running the following command:
python manage.py startapp hello

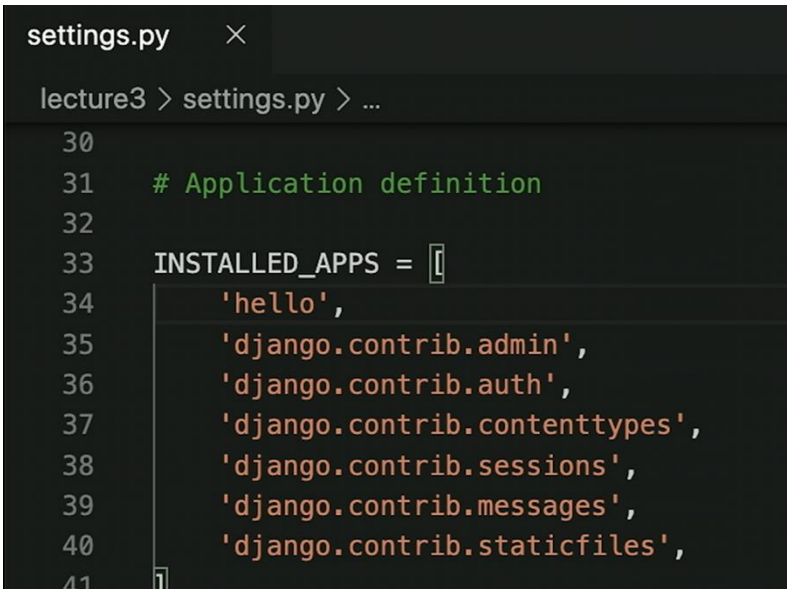




After creating the application, this app has to be installed in Django project by following these steps:

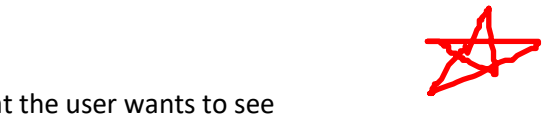
Lecture3->Settings.py

Write the following code to the INSTALLED_APPS

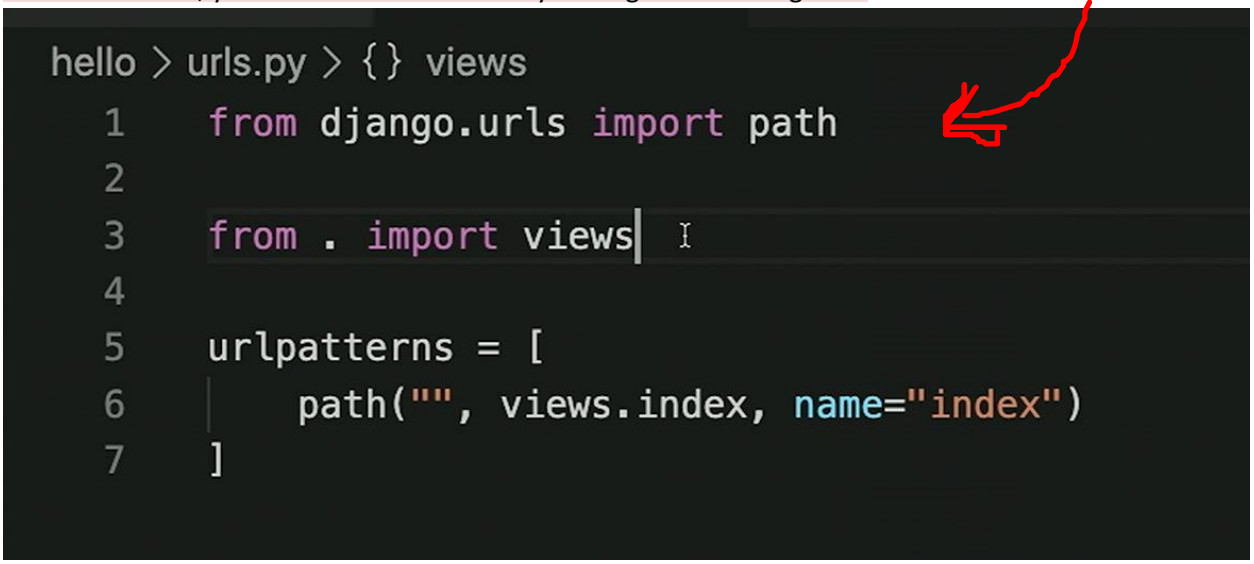


This means that ‘hello’ will be installed application on the Lecture3 django project

Making hello app to display something:
Go to hello directory->views.py
You can think of view that it is something that the user wants to see
To create a view, you can create a function by writing the following code



الكود هذا ينكتب في صفحة اليوارل متع التطبيق

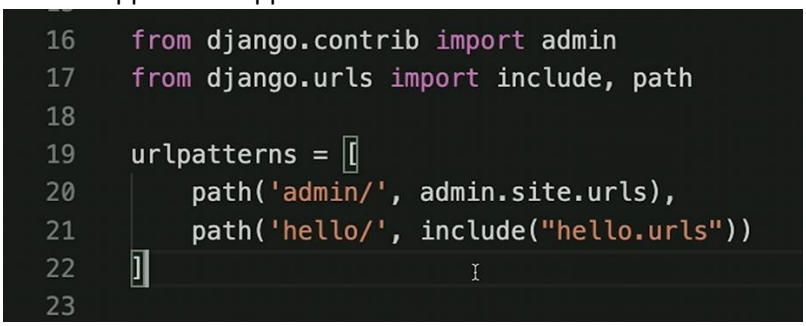


Urlpatterns: is a variable contains a list of all available urls that can be accessed in this application

هو متغير يحتوي على قائمة بجميع عناوين url
المتاحة التي يمكن الوصول إليها في هذا التطبيق

Last step:
Open urls.py from the project directory (Lecture3)

This is the list for the entire urls that can be visited in the entire Django project, and we need to add the application app:



Which means add the urls that are related to the ‘hello’ application and include it to the url list for the ‘Lecture3’ Django project

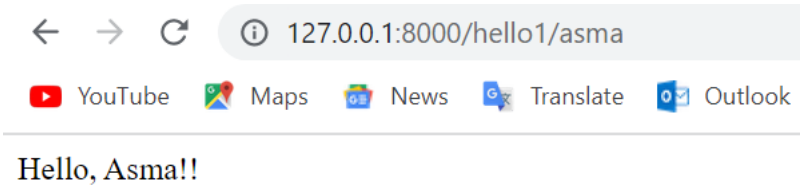
To run the application, type the command:
Python manage.py runserver
And follow the link:

We can create as many views as we like with additional functions and additional responses,

```python
# hello1 > views.py > asma
from django.shortcuts import render
from django.http import HttpResponse


# Create your views here.

def index(request):
    return HttpResponse("Hello!")

def asma(request):
    return HttpResponse("Hello, Asma!!")
```

الكود هذا ثاني حاجة نديرها بعدين نخش في جو اليوارل

```python
# hello1 > urls.py > ...
from django.urls import path
from . import views
urlpatterns=[
    path("", views.index, name="index"),
    path("asma", views.asma, name="asma"),
]
```



Example:
To create a custom URL with different parameters to write different names in the URL and used as a parameter, create a new function called great with a name parameter and change the URLS.py file accordingly

```
16
17  def greet(request,name):
18      return HttpResponse(f"Hello, {name}")
```

```
8       path("<str:name>",views.greet, name="great"),
```

To capitalize the name parameter change the great function accordingly:

```
14  def greet(request, name):
15      return HttpResponse(f"Hello, {name.capitalize()}!")
```

Including HTML to the views

Instead of returning a string in a response, it is possible to render an entire HTML document from a template by writing a function in the views as the following:
1. creating 'template**s**' directory inside the app directory in our case 'hello'
2. creating 'hello' directory inside the 'template' directory
3. creating 'index.html' inside the 'hello' directory.
4. writing the index function inside the views file

ندير فولدر جديد باسم templates
ضروري من حرف الs

```
5   def index(request):
6       return render(request, "hello/index.html")
```

In order to capitalize a variable name in the URL we can change the greet function as the following:
1. creating greet.html file inside the 'templates/hello' directory
2. writing the greet function in the views as follow:

```python
def greet(request, name):
    return render(request, "hello/greet.html", {
        "name": name.capitalize()
    })
```

3. writing the HTML code for greet.html file:
Passing the name variable in {{}}

hello > templates > hello > greet.html > html > body > h1

```html
<!DOCTYPE html>
<html lang="en">
    <head>
        <title>Hello</title>
    </head>
    <body>
        <h1>Hello, {{ name }}!</h1>
    </body>
</html>
```

The reason for all these different files is to help keep things separated:
1. urls.py is responsible for URLs and directing people to different views. مسؤول عن عناوين URL وتوجيه الأشخاص إلى وجهات نظر مختلفة .
2.views.py is responsible for different views and for a particular view what template to be rendered and what information to be passed in this context مسؤول عن طرق العرض المختلفة ولعرض معين ما هو النموذج الذي سيتم تقديمه وما هي المعلومات التي سيتم تمريرها في هذا السياق
3. html files to show what the page will look like.
4. Separating the components of the web application helps to build the structure of the Django application clear.

Is new year application
Views:

newyear > views.py > index

```python
import datetime
from django.shortcuts import render

# Create your views here.
def index(request):
    now=datetime.datetime.now()
    return render(request, "newyear/index.html", {
        "newyear": now.month==1 and now.day==1
    })
```

Index.html file

```
newyear > templates > newyear > index.html > html > body
<!DOCTYPE html>
<html lang="EN">
    <title>Is it new year?</title>
    <body>
        {% if newyear %}
        <h1>Yes</h1>
        {%else %}
        <h1>NO</h1>
        {%endif%}
    </body>
</html>
```

Urls.py file

```
newyear > urls.py > ...
from django.urls import path
from . import views
urlpatterns=[
    path("", views.index, name="index")
]
```