Tutorial

Instalar python:

Buscar instalador de python en https://www.python.org/downloads/ descargar ejemplo versión 2.7.9.

Instalar pip:

Buscar instalador de pip en https://pip.pypa.io/en/stable/installing/ descargar archivo get-pip.py y ejecutar. Luego de terminada la ejecución, ya tenemos instalado pip.

Instalar virtualenv y virtualenvwrapper:

Desde la consola de Windows ejecutar el siguiente comando.

```
pip install virtualenvwrapper-win
```

Tras la ejecución del comando ya tenemos instalado el virtualenv y virtualenvwrapper.

Luego se procede a crear un entorno virtual para un proyecto con el comando:

```
mkvirtualenv myproject
```

El nuevo entorno creado se activara inmediatamente y usted lo podrá ver en la línea de consola.
Una vez que salga de la pantalla de comandos puede regresar el ambiente del proyecto ejecutando el siguiente comando:

```
workon nombre_proyecto_creado
```

Instalar Django:

Para instalar Django es sencillo se ejecuta desde la consola el siguiente comando:

```
pip install django
```

Para saber si ya posee instalado django puede ejecutar el siguiente comando:
django-admin –version

Siempre desde dentro del ambiente que creado por usted. Es decir siempre ejecutando los comandos en consola después de haber llamado a workon nombre del ambiente.

Descargar psycopg2 desde http://www.stickpeople.com/projects/python/win-psycopg/ aquí se encuentran para las diferentes versiones de python y en 32 o 64 bit. Descargar el requerido según la versión de python instalada y luego ejecutar el instalador. Durante el proceso de instalación reconoce su versión de pyhton, de no ser así, tiene un instalador de una versión q no corresponde con la instalada en su ordenador o tiene una versión de 64 bit y una instalación de pyhton de 32 bit.

Instalar Postgresql:

Seleccionar la versión a emplear desde http://www.enterprisedb.com/products-services-training/pgdownload#windows
Así como la interfaz visual pgAdmin para el trabajo con postgresql desde http://www.pgadmin.org/download/windows.php

Ya contamos con todas las herramientas necesarias para la creación de un proyecto.

Creación de un proyecto:

Se procede a entrar en el ambiente creado con workon nombre_ambiente. Se procede a la creación de un proyecto. Primeramente con el empleo de cd se puede ubicar en el directorio donde desea crear el proyecto, ejemplo, cd C:\Users\user\Desktop\Web\ejemplo y luego con el comando django-admin.py startproject nombre_ proyecto se crea el proyecto el cual creara dentro de la ubicación deseada la siguiente estructura:

nombre_proyecto/

__init__.py

manage.py

settings.py

urls.py

- __init__.py: Un archivo vacío que le dice a Python que este directorio debería ser considerado un paquete Python. (Lee más sobre paquetes en la documentación oficial de Python si eres principiante).
- manage.py: Una utilidad de línea de comandos que te permite interactuar de distintas formas con este proyecto Django. Puedes leer todos los detalles sobre manage.py en django-admin.py y manage.py.
- settings.py: Configuración para este proyecto Django. En Django settings puedes entender más sobre cómo funciona la configuración.
- urls.py: Las URLs para este proyecto Django; una "tabla de contenidos" de tu sitio basado en Django.

Para verificar q lo siguiente funciona entre dentro del directorio nombre_proyecto desde la consola y ejecute: `python manage.py runserver` donde vera una salida similar a esta:

```
Validating models...
0 errors found.

Django version 1.0, using settings 'mysite.settings'
Development server is running at http://127.0.0.1:8000/
Quit the server with CONTROL-C.
```

Acabas de arrancar el servidor de desarrollo de Django, un servidor web liviano escrito completamente en Python. Este es un buen momento para que recuerdes: NO uses este servidor en nada que se parezca a un entorno de producción. Su objetivo es solo ser usado para desarrollo. Por defecto el comando `runserver` arranca el servidor como localhost y puerto 8000, este puede cambiar ejecutando:

```
python manage.py runserver 8080 #Para cambiar solo el puerto
python manage.py runserver 0.0.0.0:8000    #Para cambiar el ip y puerto
```

Para configurar la BD se procede a escribir dentro del archivo generado settings.py. Modificando el apartado database.

```
DATABASES = {
   'default': {
      'ENGINE': 'django.db.backends.postgresql_psycopg2',
      'NAME': 'ejemplo',                  # Nombre de la BD
      'USER': 'postgres',                 # Usuario para conectarse a la BD
      'PASSWORD': '',                     # Contraseña
      'HOST': '',                          # Vacio para localhost, o definer IP.
      'PORT': '5432',                     # Puerto de conexión a la BD.
   }
}
```

Con este paso aseguramos conectarnos a una BD previamente creada.

Luego procedemos a crear la estructura dentro de nuestro proyecto. Donde crearemos las carpeta apps que contendrán las aplicaciones, una carpeta templates y una carpeta static para almacenar (ccs, js, etc.)

Se procede a poner las siguientes líneas dentro del archivo settings.py

```
import os
import sys

BASE_DIR = os.path.dirname(os.path.dirname(os.path.abspath(__file__)))

sys.path.insert(0, os.path.join(BASE_DIR, 'apps'))
```

Para que todas las aplicaciones que creemos se reconozcan dentro de la carpeta apps creada con anterioridad.

Creación de una aplicación:

Para crear una aplicación desde el entorno previamente creado se ejecuta el comando:

```
python manage.py startapp nombre_ aplicacion
```

Creación de los modelos:

Se procede a crear los modelos en el archivo models.py ejemplo:

```
class Catalogo(models.Model):
    nombre = models.CharField(max_length=50)
    valor = models.IntegerField(default=0)
```

Luego se procede a ejecutar el siguiente comando desde la consola:

python manage.py makemigrations

luego:

```
python manage.py mígrate
```

y se crean los modelos.

Se crea el superusuario para la administración del sitio con el comando:

```
Python manage,py createsuperuser
```

Se completa la información con el nombre, correo y contraseña.