

**Manual DJANGO**

**¿Cómo instalar DJANGO en Windows?**

Los pasos de esta guía se han probado con Windows 7 y 8. En otras versiones, los pasos serían similares.

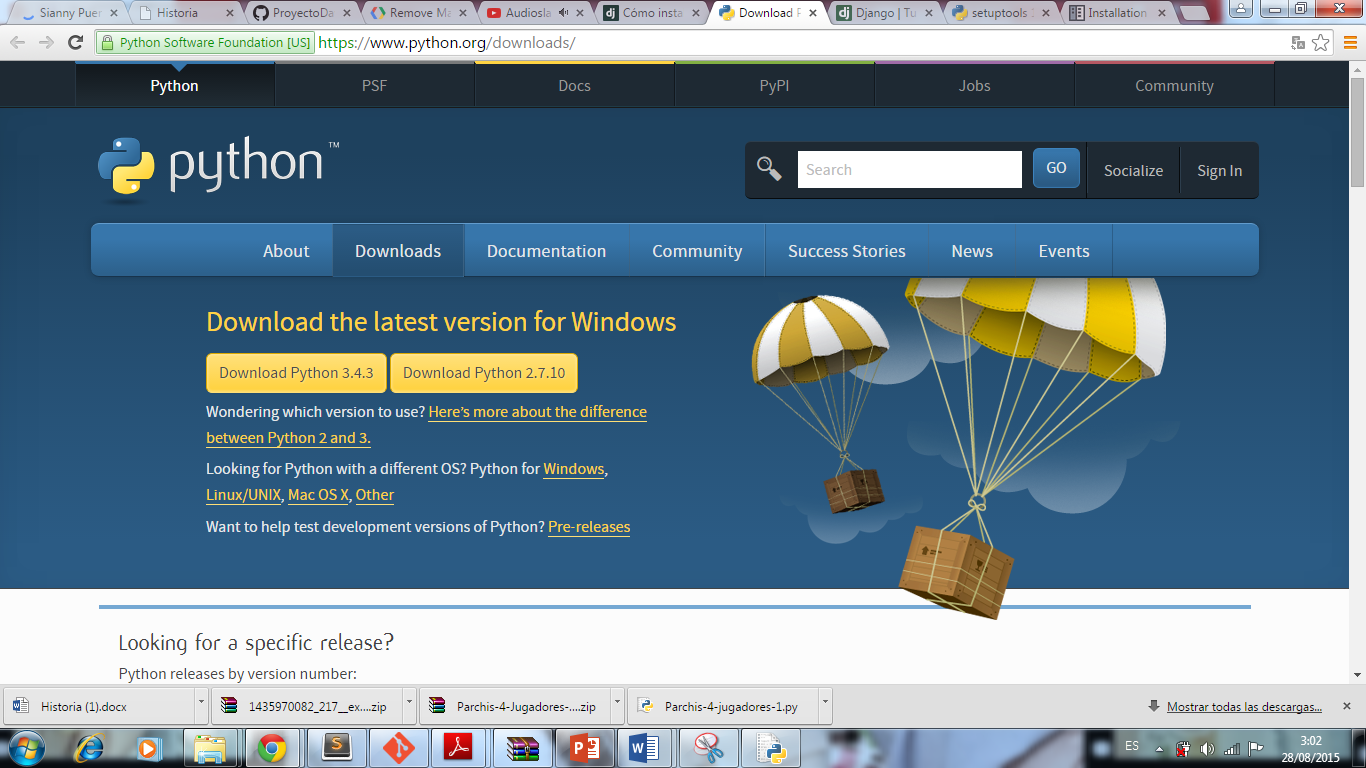
Pasos:

1. Instalar Python
2. Instalar Pip
3. Instalar Django

**Instalar Python**

Django es un framework web de Python, por lo que requiere Python para ser instalado en su máquina.

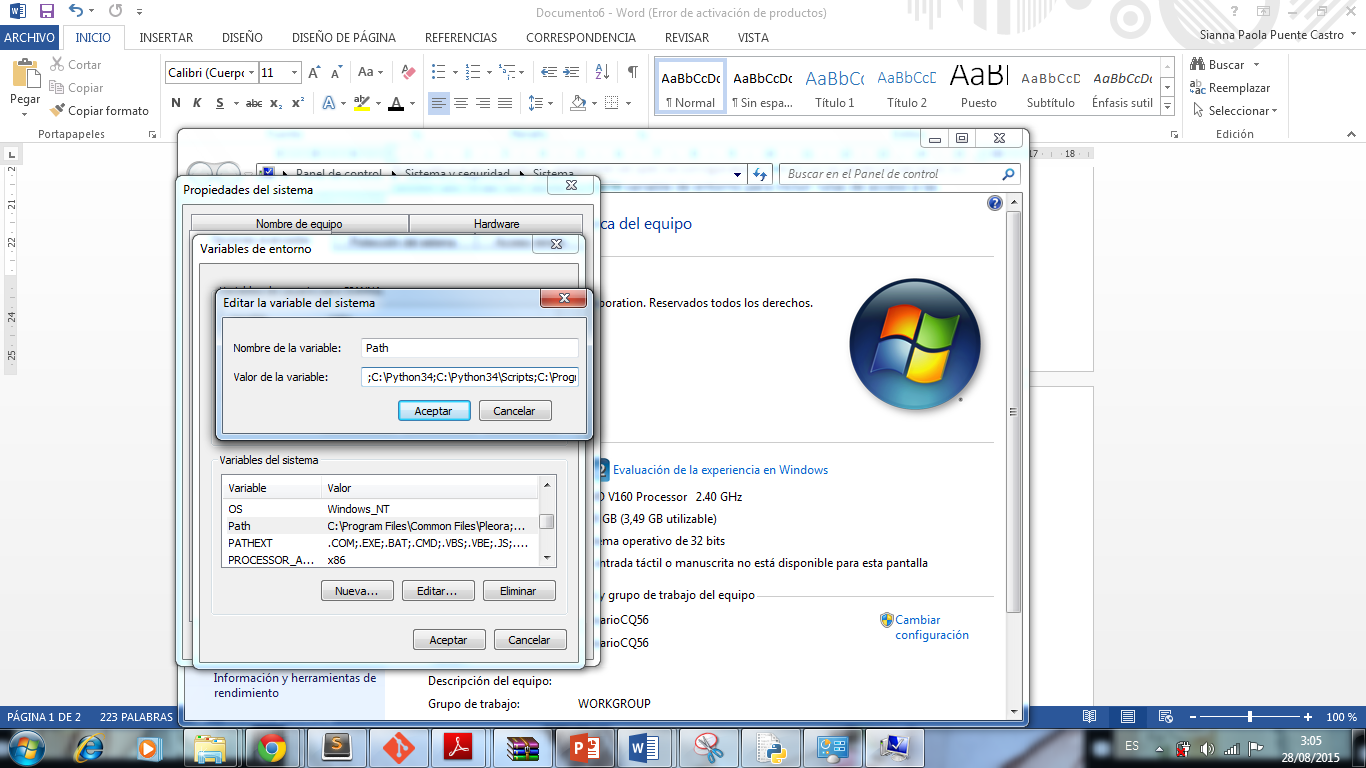
Para instalar Python en su máquina vaya a https://python.org/download/, y descarga un instalador MSI de Windows para Python. Una vez descargado, ejecute el instalador MSI y siga las instrucciones que aparecen en pantalla.

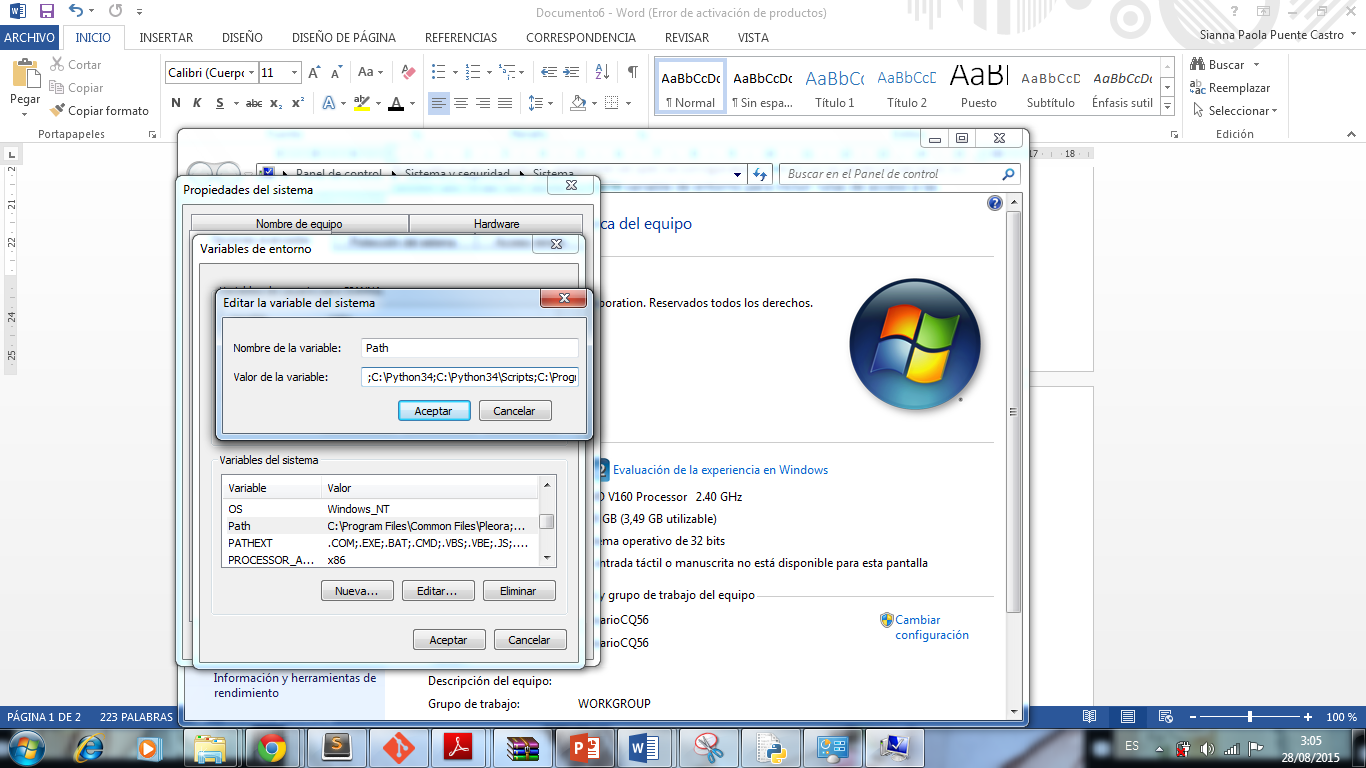


Después de la instalación, abra el símbolo del sistema y comprobar la versión de Python ejecutando **python --version.**

Si encuentra un problema, asegúrese de que ha configurado el **PATH** correctamente variable. Es posible que tenga que ajustar su **PATH** variable de entorno para incluir rutas de acceso a las secuencias de comandos ejecutables y adicionales Python. Por ejemplo, si su Python se instala en **C: \ Python34**\, las siguientes rutas deben añadirse al **PATH:**

**C:\Python34\;C:\Python34\Scripts;**



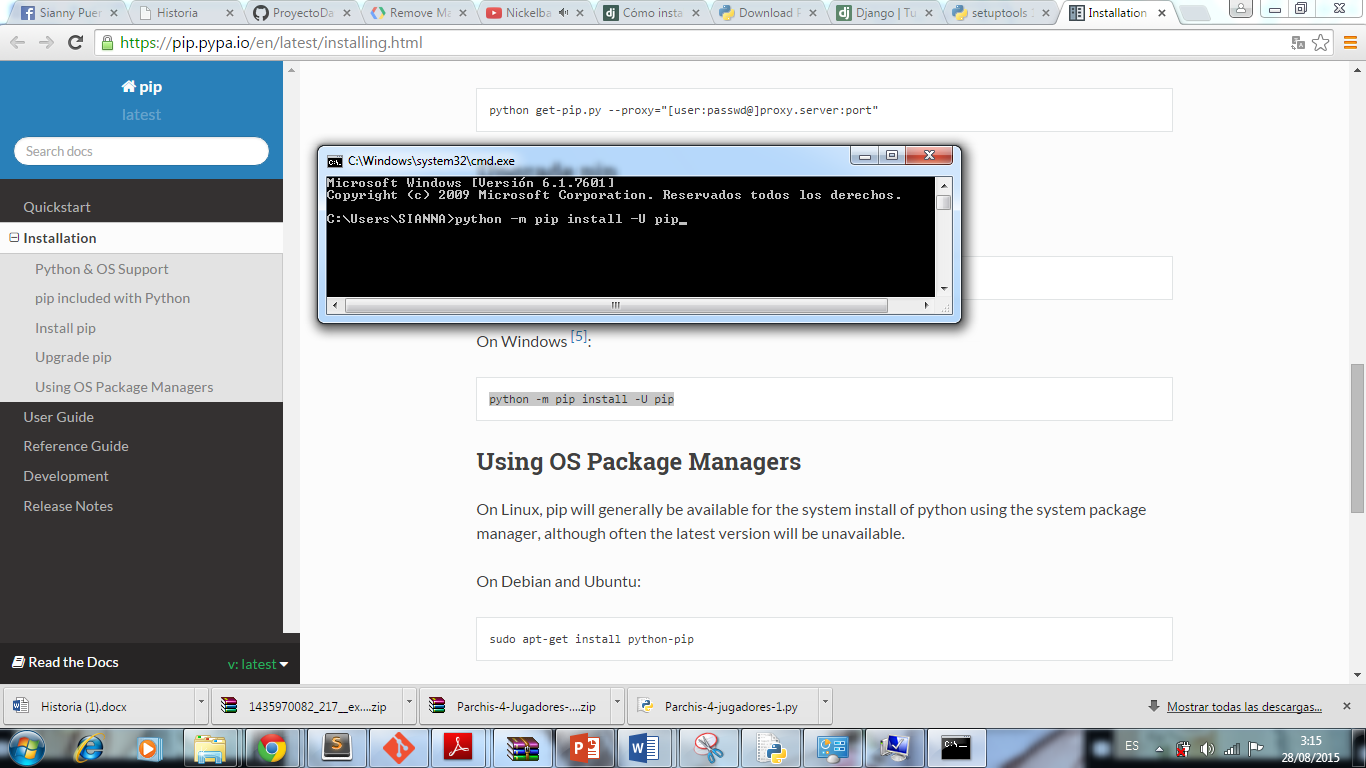


**Instalar PIP**

PIP es un gestor de paquetes para Python que utiliza el índice de paquetes de Python para instalar paquetes de Python. PIP tarde puede utilizar para instalar Django desde PyPI.

Si ha instalado Python 3.4, **pip** se incluye lo que puede saltarse esta sección. Caso contrario vaya a <http://www.pip-installer.org/en/latest/installing.html> para instalar / actualizar instrucciones.

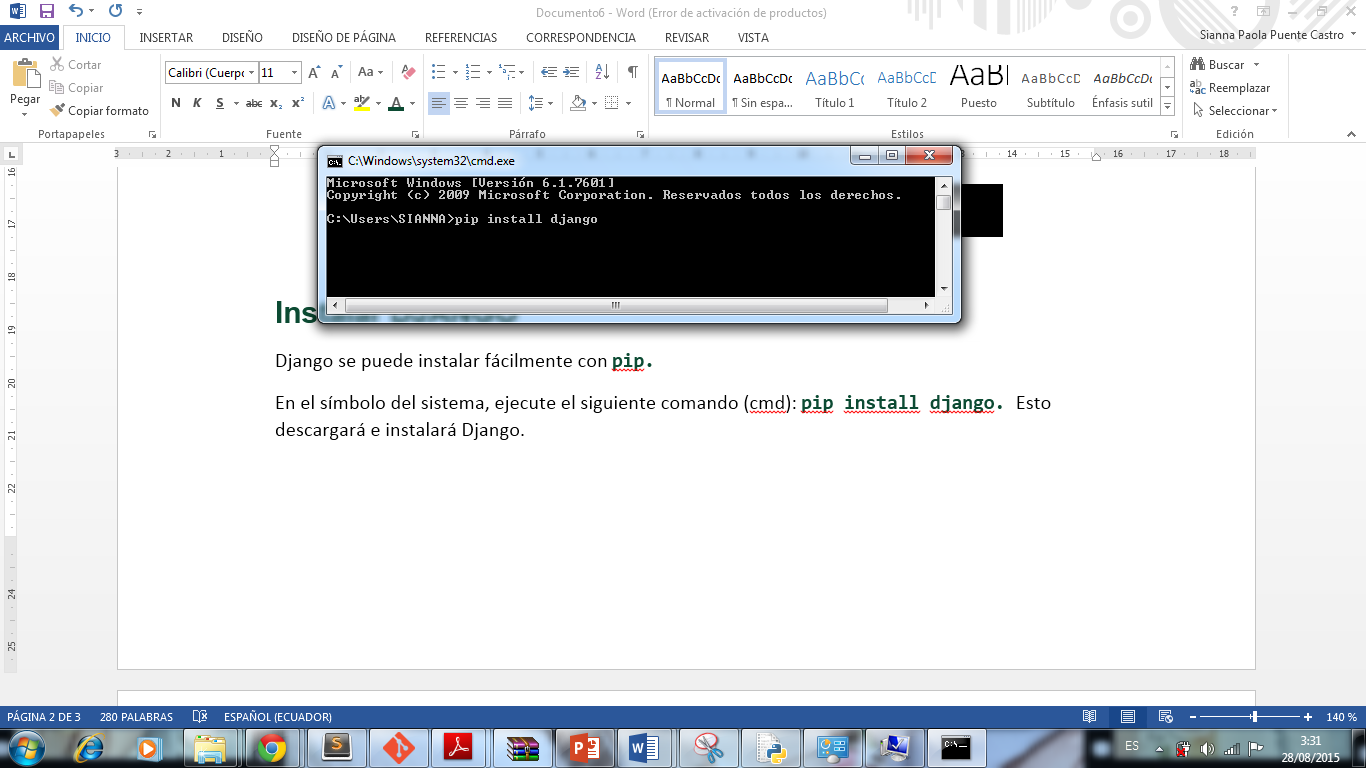
En Windows colocar en consola (cmd) :



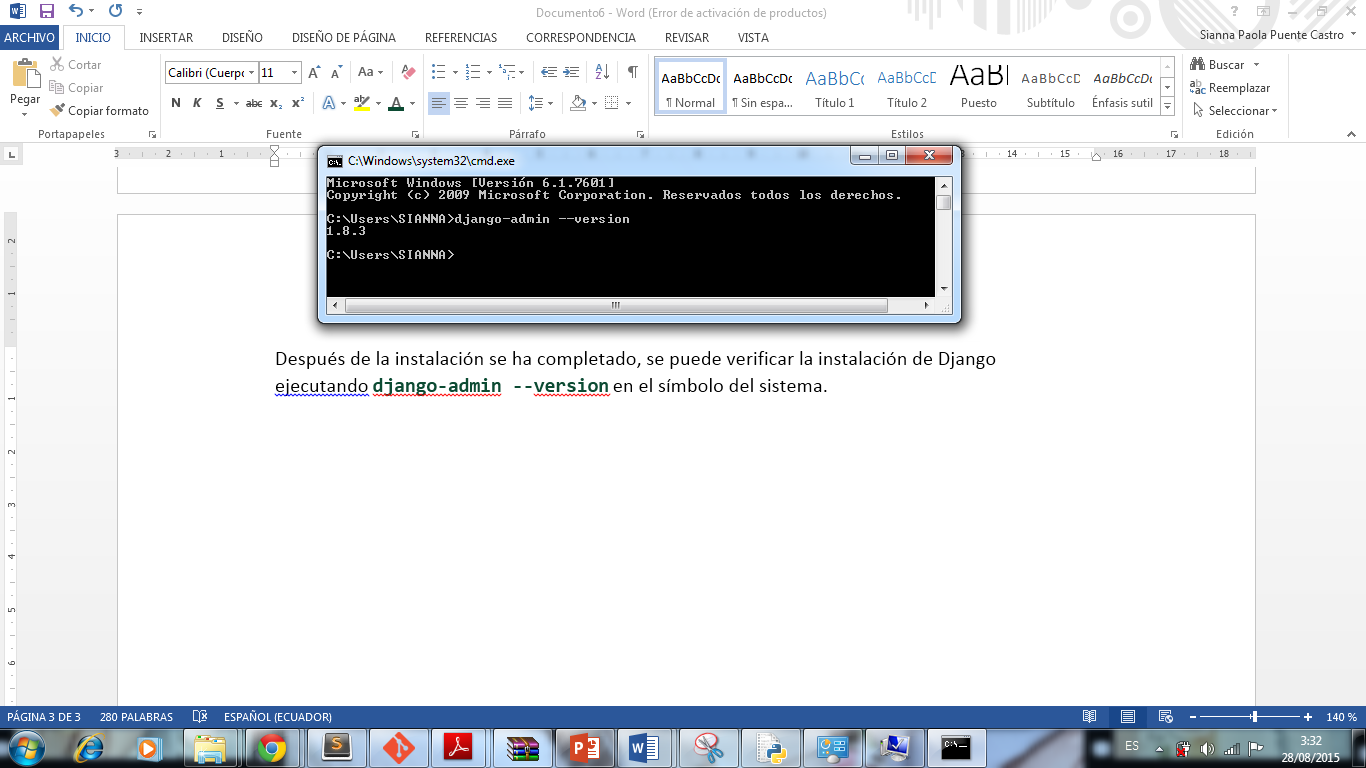
**Instalar DJANGO**

Django se puede instalar fácilmente con **pip.**

En el símbolo del sistema, ejecute el siguiente comando (cmd): **pip install django.**Esto descargará e instalará Django.



Después de la instalación se ha completado, se puede verificar la instalación de Django ejecutando **django-admin --version** en el símbolo del sistema.



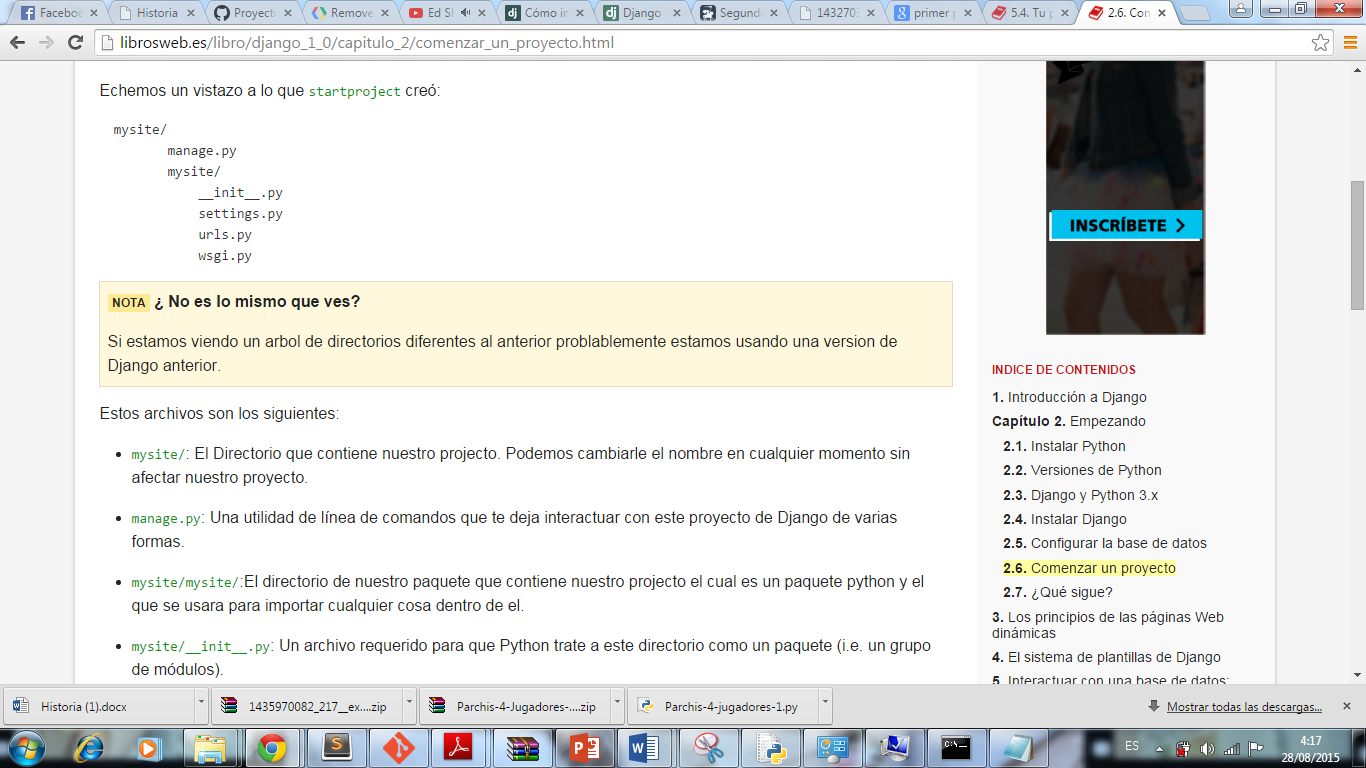
**Comenzar un proyecto**

Un **proyecto** es una colección de configuraciones para una instancia de Django, incluyendo configuración de base de datos, opciones específicas de Django y configuraciones específicas de aplicaciones.

Si esta es la primera vez que usas Django, tendrás que tener cuidado de algunas configuraciones iniciales. Crea un nuevo directorio para empezar a trabajar.

Para empezar un proyecto en Django ejecuta el comando **django-admin.py startproject** **nombredelproyecto** para crear el directorio con el nombre del proyecto escogido en el directorio actual. Para demostración de nuestro ejemplo crearemos un proyecto con el nombre **mysite**

Echemos un vistazo a lo que **startproject**creó:



Estos archivos son los siguientes:

**mysite/**: El Directorio que contiene nuestro proyecto. Podemos cambiarle el nombre en cualquier momento sin afectar nuestro proyecto.

**manage.py**: Una utilidad de línea de comandos que te deja interactuar con este proyecto de Django de varias formas. Este archivo es sustancial y tiene las funciones de Django para probar el proyecto, actualizar cambios en la base, etc.

**mysite/mysite/:**El directorio de nuestro paquete que contiene nuestro proyecto el cual es un paquete python y el que se usara para importar cualquier cosa dentro de él.

**mysite/\_\_init\_\_.py**: Un archivo requerido para que Python trate a este directorio como un paquete (i.e. un grupo de módulos).

**mysite/settings.py**: Opciones/configuraciones para este proyecto de Django.

**mysite/urls.py**: La declaración de las URL para este proyecto de Django; una "tabla de contenidos" de tu sitio hecho con Django. Script para mapear con patrones lo que se escriba en el url se re direccione a cierta función o parámetro (Muy importante!).

**mysite/wsgi.py**: El archivo encargado de ser compatible con el servidor web.

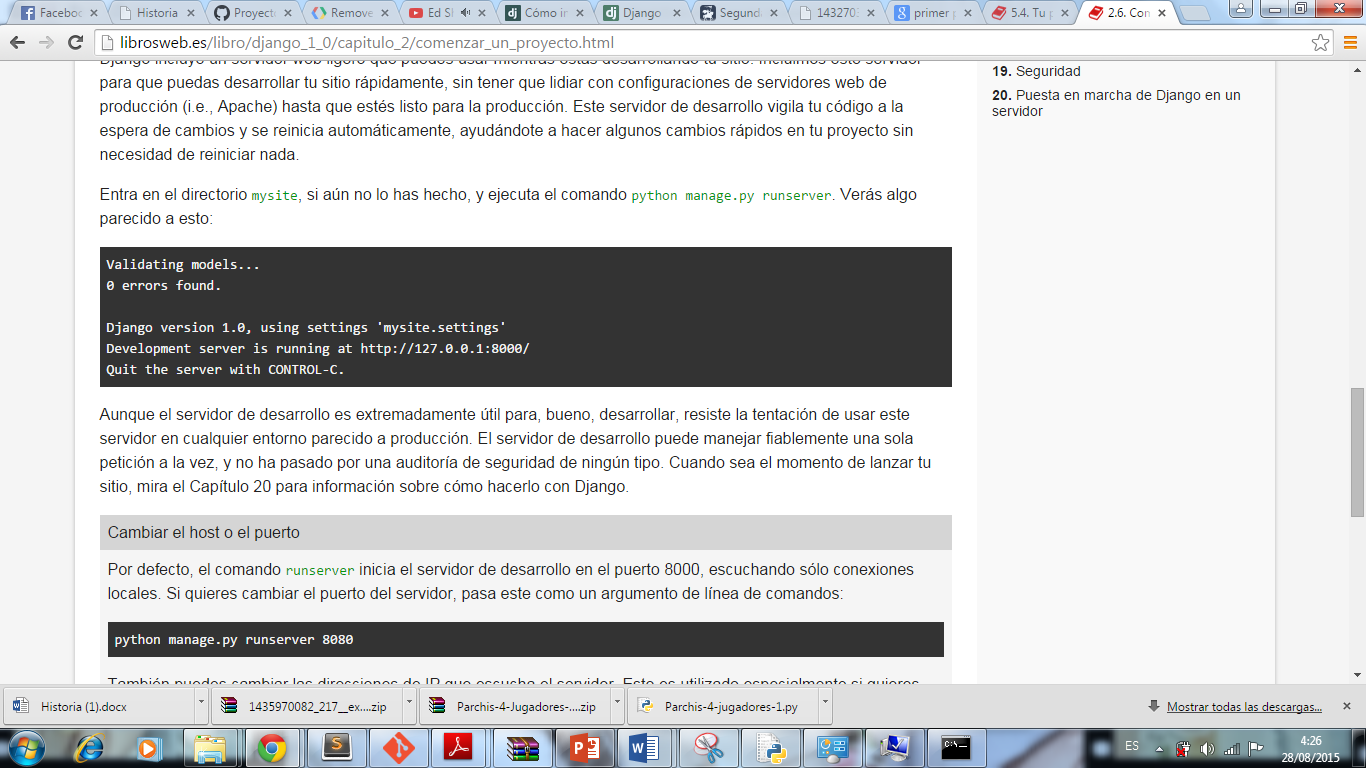
**El servidor de desarrollo**

Django incluye un servidor web ligero que puedes usar mientras estás desarrollando tu sitio. Incluimos este servidor para que puedas desarrollar tu sitio rápidamente, sin tener que lidiar con configuraciones de servidores web de producción (i.e., Apache) hasta que estés listo para la producción. Este servidor de desarrollo vigila tu código a la espera de cambios y se reinicia automáticamente, ayudándote a hacer algunos cambios rápidos en tu proyecto sin necesidad de reiniciar nada.

Entra en el directorio **mysite**, si aún no lo has hecho, y ejecuta el comando

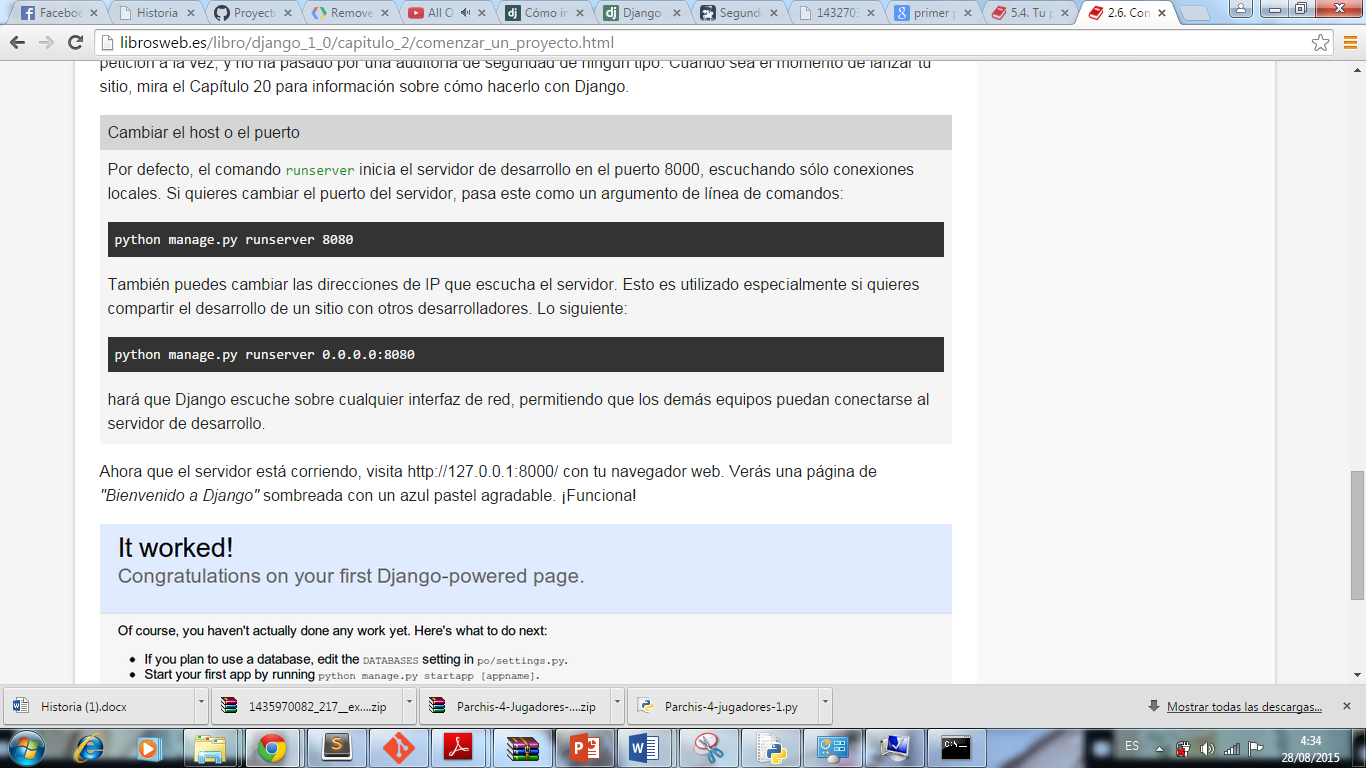
**python manage.py runserver**

Verás algo parecido a esto:

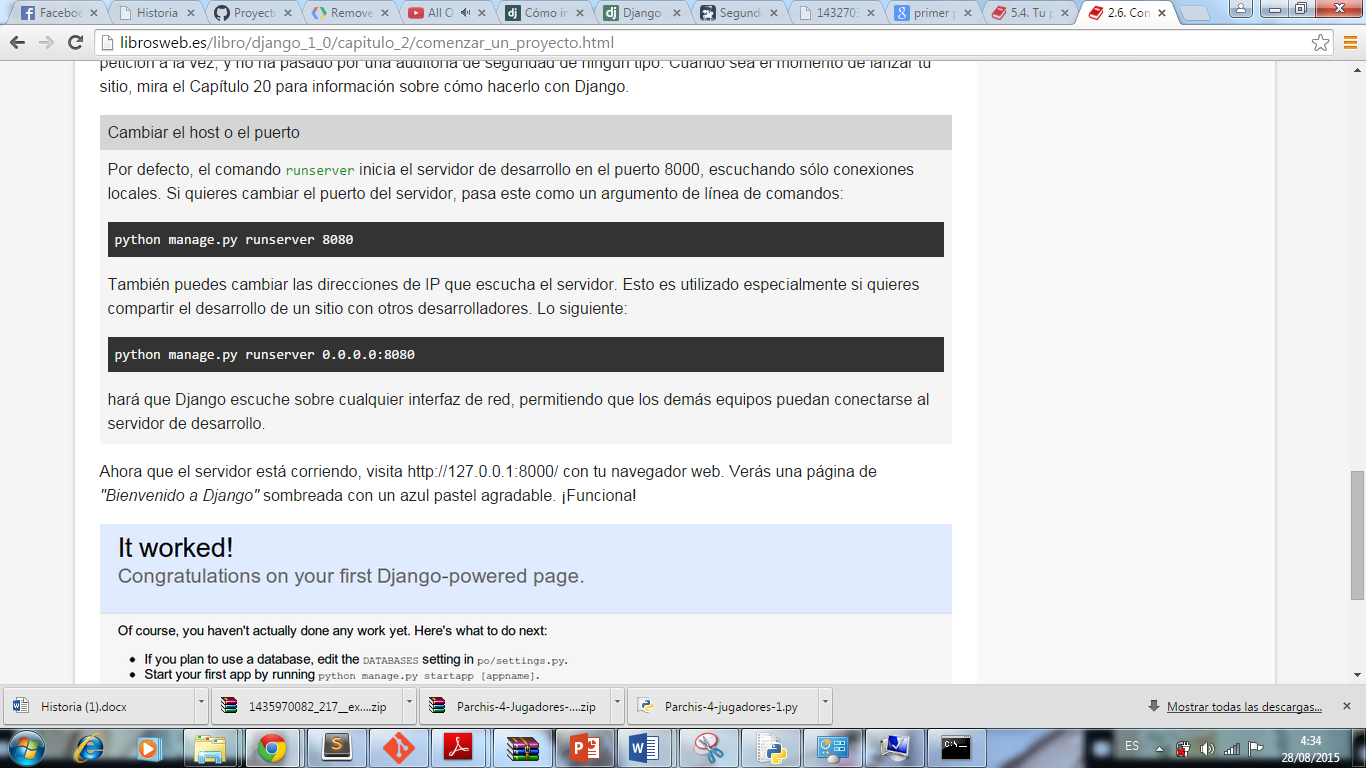


**Cambiar el host o el puerto**

Por defecto, el comando **runserver** inicia el servidor de desarrollo en el puerto 8000, escuchando sólo conexiones locales. Si quieres cambiar el puerto del servidor, pasa este como un argumento de línea de comandos:

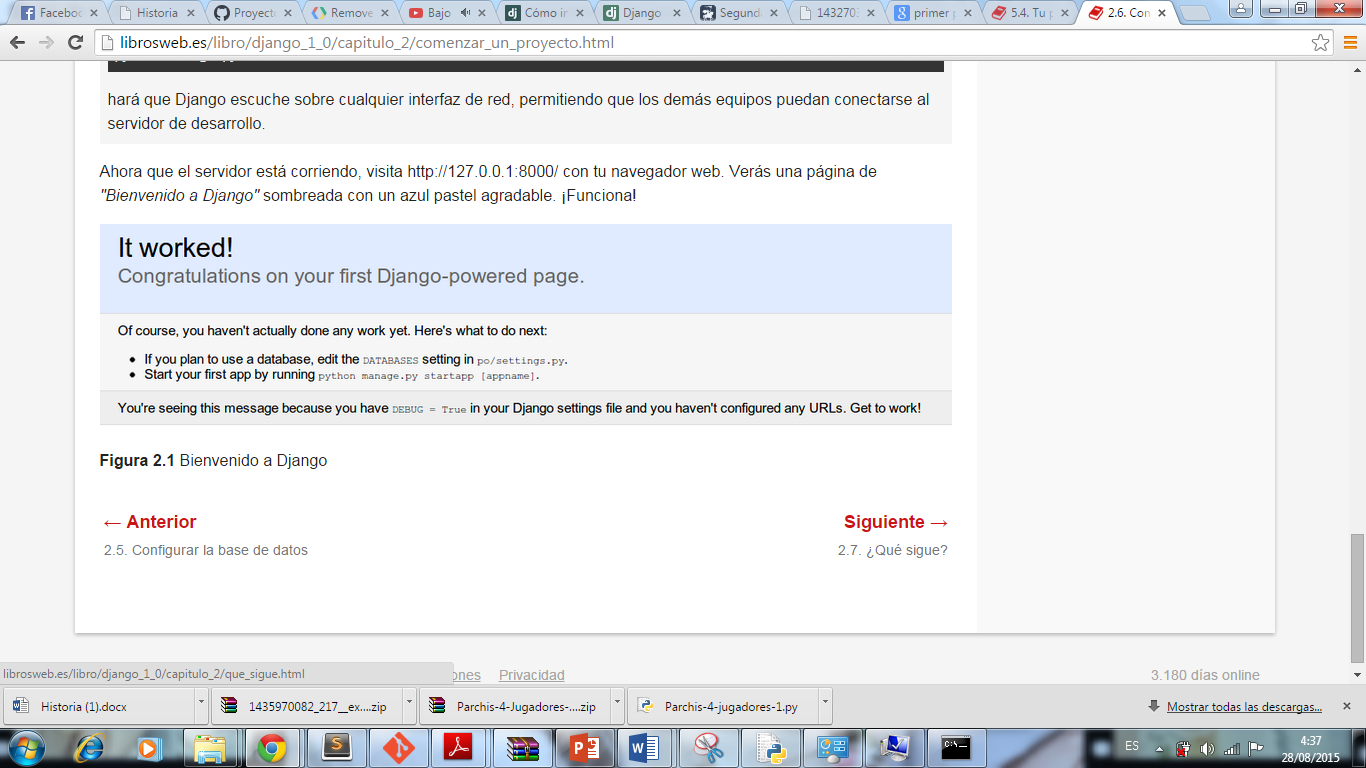


También puedes cambiar las direcciones de IP que escucha el servidor. Esto es utilizado especialmente si quieres compartir el desarrollo de un sitio con otros desarrolladores. Lo siguiente:



hará que Django escuche sobre cualquier interfaz de red, permitiendo que los demás equipos puedan conectarse al servidor de desarrollo.

Ahora que el servidor está corriendo, visita http://127.0.0.1:8000/ con tu navegador web. Verás una página de"Bienvenido a Django" sombreada con un azul pastel agradable. ¡Funciona!



Aunque el servidor de desarrollo es extremadamente útil para, bueno, desarrollar, resiste la tentación de usar este servidor en cualquier entorno parecido a producción. El servidor de desarrollo puede manejar fiablemente una sola petición a la vez, y no ha pasado por una auditoría de seguridad de ningún tipo. Cuando sea el momento de lanzar tu sitio, mira el Capítulo 20 para información sobre cómo hacerlo con Django.

**Comenzar una aplicación**

Ahora que verificamos que la conexión está funcionando, es hora de crear una **Aplicación de Django**— una colección de archivos de código fuente, incluyendo modelos y vistas, que conviven en un solo paquete de Python y representen una aplicación completa de Django.

Vale la pena explicar la terminología aquí, porque esto es algo que suele hacer tropezar a los principiantes. Ya hemos creado un **proyecto**, entonces, ¿cuál es la diferencia entre un **proyecto** y una **aplicación**? La diferencia es la que existe entre la configuración y el código:

* Un **proyecto** es una instancia de un cierto conjunto de aplicaciones de Django, más las configuraciones de esas aplicaciones.

Técnicamente, el único requerimiento de un proyecto es que este suministre un archivo de configuración, el cual define la información hacia la conexión a la base de datos, la lista de las aplicaciones instaladas, la variable TEMPLATE\_DIRS, y así sucesivamente.

* Una **aplicación** es un conjunto portable de una funcionalidad de Django, típicamente incluye modelos y vistas, que conviven en un solo paquete de Python.

Por ejemplo, Django incluye un número de aplicaciones, tales como un sistema de comentarios y una interfaz de administración automática. Una cosa clave para notar sobre estas aplicaciones es que son portables y reusables en múltiples proyectos.

Entonces después de crear un proyecto, se debe crear mínimo un **subproyecto** que es con el que se trabajará. El primero creado es para configuraciones. ¿Cómo crear un subproyecto?

Nos situamos dentro de la carpeta de nuestro proyecto y:

**python manage.py stratapp nombredelsubproyecto**

En este caso recomiendo poner de nombre web, ¿por qué? Para tener una mejor relación de los archivos que lo contienen que son parte de los html, css, js, etc.

Dentro de esta carpeta tendremos algunos archivos como:

**models.py:** Django trabaja con un ORM y en este script se escribirá como clases, tus tablas con sus atributos.

**views.py**: Las views son las funciones con que trabajaremos la interacción del navegador con el servidor, puesto a que una url (revisar el archivo que creamos en el proyecto principal) llama a un view y el view da una respuesta. Un ejemplo básico es un view que retorna una página html.

**admin.py**: Django te da un administrador de contenido, y cuando quieres mostrar cierto modelo para que sea administrado (ver, modifica, eliminar, insertar) debe ser agregado a este archivo.

**migrations**: En esta carpeta se guardan los archivos de migración de base de datos. Las actualizaciones de cambios realizados en el models.

**static**: En esta carpeta deben ir los archivos estaticos (css, js, imagenes).

**templates** : En esta carpeta deben ir los html, xml, json; se recomienda trabajar con Json.