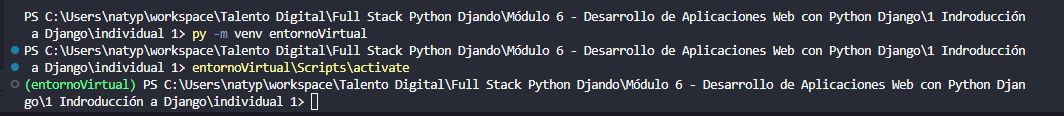
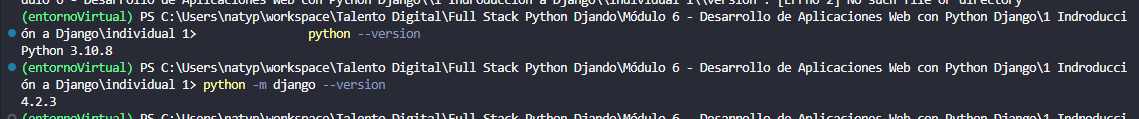
* **Ambiente virtual**
  + python –m venv entornoVirtual
* **Activar el ambiente en termianl:**
  + entornoVirtual \Scripts\activate



* **Instalar Django en el proyecto**
  + pip install django
* Comprobar versiones:
  + Python –version
  + Python –m django –version



**Identifique qué paquetes se descargan automáticamente. Investigue la utilidad que tienen estos paquetes.**

Los paquetes que se instalan automáticamente al ejecutar el comando "pip install django" son:

* **tzdata**: Este paquete proporciona información sobre zonas horarias (timezone data). Django utiliza esta información para manejar fechas y horas en diferentes zonas horarias de forma precisa y coherente. La biblioteca tzdata asegura que Django tenga acceso actualizado a las reglas de las zonas horarias para realizar conversiones adecuadas.
* **typing-extensions**: Es una extensión de la biblioteca typing estándar de Python. Esta extensión ofrece mejoras para ayudar a los desarrolladores a escribir anotaciones de tipo más avanzadas y flexibles. Aunque puede no estar directamente relacionado con el funcionamiento básico de Django, algunos paquetes o herramientas utilizadas junto con Django podrían requerirlo para mejorar el soporte de tipos.
* **sqlparse**: Esta biblioteca se utiliza para analizar y formatear declaraciones SQL (Structured Query Language). Django utiliza sqlparse para formatear las consultas SQL generadas por el ORM (Object-Relational Mapping) de Django, lo que facilita la lectura y depuración de las consultas en entornos de desarrollo.
* **asgiref**: Este paquete proporciona una utilidad para trabajar con aplicaciones asincrónicas en Python, siguiendo la especificación ASGI (Asynchronous Server Gateway Interface). Django 3.0 y versiones posteriores son compatibles con ASGI para admitir aplicaciones asincrónicas, lo que permite utilizar tecnologías como WebSockets y otras funciones asincrónicas avanzadas.
* **django**: Es el paquete principal de Django en sí mismo. Contiene el framework web completo de Django, que incluye el ORM, el sistema de enrutamiento, la gestión de formularios, la administración de sesiones y muchas otras características útiles para desarrollar aplicaciones web de manera rápida y robusta.

En resumen, estos paquetes se instalan automáticamente para proporcionar funcionalidades adicionales o mejorar ciertos aspectos del framework Django. Cada uno desempeña un papel importante para garantizar que Django funcione correctamente y sea capaz de satisfacer una amplia variedad de necesidades en el desarrollo de aplicaciones web.

**¿Qué facilidades nos proporciona Django?**

Django es un popular framework de desarrollo web en Python que proporciona una amplia variedad de facilidades para simplificar el proceso de desarrollo web. Algunas de las principales facilidades que Django ofrece son:

* **Sistema de administración:** Django viene con un poderoso sistema de administración que permite crear interfaces de administración para aplicaciones web de manera rápida y sencilla. Esto facilita la gestión de contenido y datos del sitio sin tener que desarrollar una interfaz de administración desde cero.
* **ORM (Object-Relational Mapping):** Django proporciona un ORM que permite interactuar con la base de datos utilizando objetos Python en lugar de escribir consultas SQL directamente. Esto hace que el manejo de la base de datos sea más sencillo y abstrae la complejidad del SQL.
* **Routing y vistas:** Django proporciona un sistema de enrutamiento (URL dispatcher) que permite definir patrones de URL y vincularlos a vistas que procesan las solicitudes del usuario. Esto facilita el manejo de solicitudes y respuestas HTTP.
* **Plantillas**: Django tiene un sistema de plantillas que permite separar la lógica de presentación del código de la aplicación. Esto facilita el desarrollo y mantenimiento de la interfaz de usuario.
* **Seguridad**: Django incluye características de seguridad integradas, como protección contra ataques CSRF (Cross-Site Request Forgery) y sanitización automática de datos para prevenir ataques de inyección.
* **Autenticación y autorización:** Django proporciona un sistema de autenticación de usuarios listo para usar, lo que facilita la gestión de usuarios y sesiones. También ofrece un sistema de autorización para controlar los permisos de acceso a diferentes partes del sitio.
* **Internacionalización y localización:** Django permite crear aplicaciones multilingües y localizar contenido fácilmente mediante el uso de traducciones y funciones de formateo de fechas y números para diferentes idiomas y regiones.
* **Integración con servicios externos:** Django facilita la integración con servicios web y otras APIs mediante librerías y herramientas disponibles en el ecosistema Django.
* **Testing**: Django proporciona un robusto marco de pruebas que permite escribir y ejecutar pruebas unitarias y de integración para asegurar la calidad del código.
* **Escalabilidad**: Django está diseñado para manejar proyectos de diferentes tamaños y complejidades, permitiendo su escalabilidad mediante el uso de buenas prácticas y técnicas adecuadas.

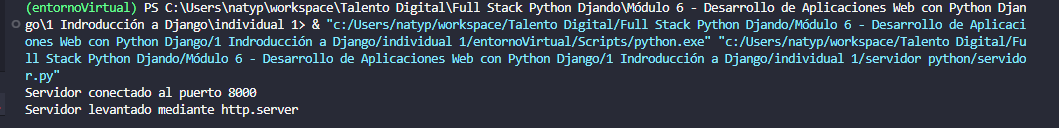
Estas facilidades hacen de Django una opción popular y poderosa para desarrollar aplicaciones web de manera rápida y eficiente.

**Con relación al levantamiento de un servidor. ¿Existe una forma de realizarlo con Python y sin Django? ¿Qué desventajas nos trae este tipo de proyectos sin Django?**

Aunque es posible desarrollar proyectos web sin utilizar Django, existen algunas desventajas a considerar:

* **Complejidad en ciertas tareas:** Sin Django, puede que tengas que implementar manualmente ciertas funcionalidades que Django ya proporciona, como el sistema de administración, autenticación de usuarios, o la capa ORM para interactuar con la base de datos.
* **Reinvención de la rueda:** En lugar de utilizar soluciones probadas y estándares proporcionados por Django, podrías tener que crear tus propias implementaciones para diferentes aspectos del desarrollo web.
* **Mantenimiento**: Django ofrece actualizaciones y mejoras continuas que garantizan la seguridad y el rendimiento de tus aplicaciones. Sin utilizar un framework, te responsabilizas de mantener tu código y asegurarte de que esté actualizado.
* **Ecosistema y comunidad:** Django tiene una comunidad activa y un ecosistema con muchas extensiones y paquetes que facilitan el desarrollo. Al optar por no utilizarlo, te perderías de estas ventajas.
* **Curva de aprendizaje**: Django tiene una curva de aprendizaje, especialmente para principiantes, pero una vez que dominas el framework, puede agilizar significativamente el desarrollo.

**Levante un servidor utilizando Python. Debe basarse en el ejercicio realizado en las capsulas. Muestre un mensaje que indique “Servidor levantado mediante http.server”**



**Por último, debes crear un proyecto en Django. Indague en la utilidad de django-admin startproject**

* Creación nuevo proyecto
  + django-admin startproject nombre\_project
  + Esto crea una carpeta en el directorio donde estemos posicionados



