Python for Web Developers: Task 2.1

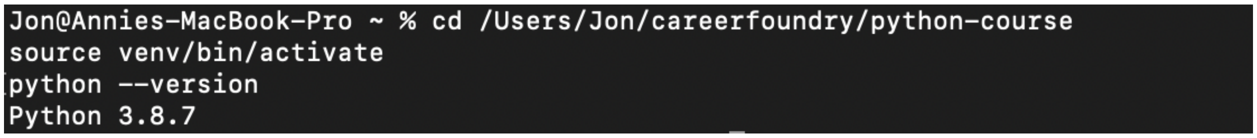
1. *Why is Django so popular among web developers?*
   Django is popular because it follows "batteries included" and rapid development principles, providing built-in features like an admin panel, authentication, ORM, and security measures (CSRF protection, SQL injection prevention) that reduce boilerplate. Its clean architecture, strong documentation, and large community make it suitable for projects from startups to large-scale applications, enabling developers to build complex web applications quickly and maintain them over time.

2. *Five companies that use Django (and their specific services uses) include*:
   Spotify: Music streaming service. Uses Django for various internal tools and web services, taking advantage of Django's admin interface and ORM for managing complex data relationships.
   Instagram: Social media platform. Uses Django for its web backend to handle millions of users, photos, and interactions, leveraging Django's scalability and rapid development capabilities.
   Pinterest: Image-sharing/ moodboard social media platform. Uses Django for its web application backend, benefiting from Django's ability to handle high traffic and complex user interactions, similar to Instagram.
   The Washington Post: News publication. Uses Django for content management and web publishing, utilizing Django's admin panel and flexible content models for managing articles and media.
   Disqus: Commenting platform. Uses Django as its core framework to handle millions of comments and user interactions across websites, relying on Django's scalability and robust database management.

3. *When, and when not to, use Django in various scenarios*:
   You need to develop a web application with multiple users: Yes, use Django. It includes built-in user authentication, user management, permissions, and session management, which simplifies multi-user features and security.
   You need fast deployment and the ability to make changes as you proceed: Yes, use Django. Its rapid development features, built-in admin panel, and ORM allow quick prototyping and iteration, making it easy to deploy and update applications.
   You need to build a very basic application, which doesn't require any database access or file operations: No, Django would be overkill. For a simple static site or minimal app without database needs, a lighter framework (e.g. Flask) or static site generator would be more appropriate and avoid unnecessary overhead.
   You want to build an application from scratch and want a lot of control over how it works: Probably not Django. Django follows "convention over configuration" and has strong opinions about structure. For maximum control, consider alternatives like Flask or FastAPI, which offer more flexibility and fewer built-in assumptions.

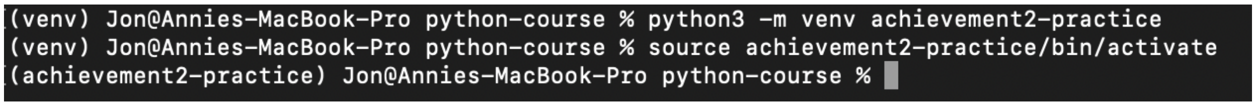
You're about to start working on a big project and are afraid of getting stuck and needing additional support: Yes, use Django. It has extensive documentation, a large community, many third-party packages, and established best practices, providing strong support for large projects and helping prevent common pitfalls.

4. *Screenshot of Python version installed on my computer*:



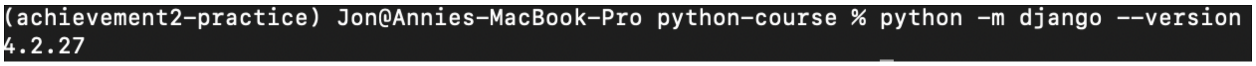
```
Jon@Annies-MacBook-Pro ~ % cd /Users/Jon/careerfoundry/python-course
source venv/bin/activate
python --version
Python 3.8.7
```

5. *Screenshot of activated virtual environment on my computer*:



```
(venv) Jon@Annies-MacBook-Pro python-course % python3 -m venv achievement2-practice
(venv) Jon@Annies-MacBook-Pro python-course % source achievement2-practice/bin/activate
(achievement2-practice) Jon@Annies-MacBook-Pro python-course %
```

6. *Screenshot of Django version installed on my computer*:



```
(achievement2-practice) Jon@Annies-MacBook-Pro python-course % python -m django --version
4.2.27
```