Answers for Exercise 2.1: Pre-Work & Django Setup

Student Name: [Your Name]
Date: [Current Date]

Part 1: Written Answers

1. Why is Django so popular among web developers?

Django is popular because it follows the "batteries-included" philosophy, providing a comprehensive set of built-in features (like an admin panel, authentication, and database ORM) that allow developers to build robust web applications quickly without needing to integrate many separate libraries. Its emphasis on clean, pragmatic design, DRY (Don't Repeat Yourself) principles, and strong security features against common vulnerabilities makes it a reliable and efficient choice for both beginners and large-scale enterprises.

2. Five large companies that use Django:

Company	Product/Service	What they use Django for
Instagram	A social media platform for photo and video sharing.	To handle its massive scale, managing user data, photo feeds, and direct messages efficiently and securely.
Spotify	A digital music streaming service.	For backend services, including managing user libraries, playlist data, and its music recommendation engine.
Pinterest	A visual discovery engine for finding ideas like recipes, home style, etc.	To build its core application logic, handle data, and manage its web interface for millions of users and "pins."

The Washington Post	A major American newspaper and online news source.	To handle high-traffic loads and build its content management system (CMS) for quickly publishing news articles.
Mozilla	The organization behind the Firefox web browser.	For many of its support websites and web services, benefiting from Django's security and ability to handle high traffic.

3. Would you use Django? Explain why or why not.

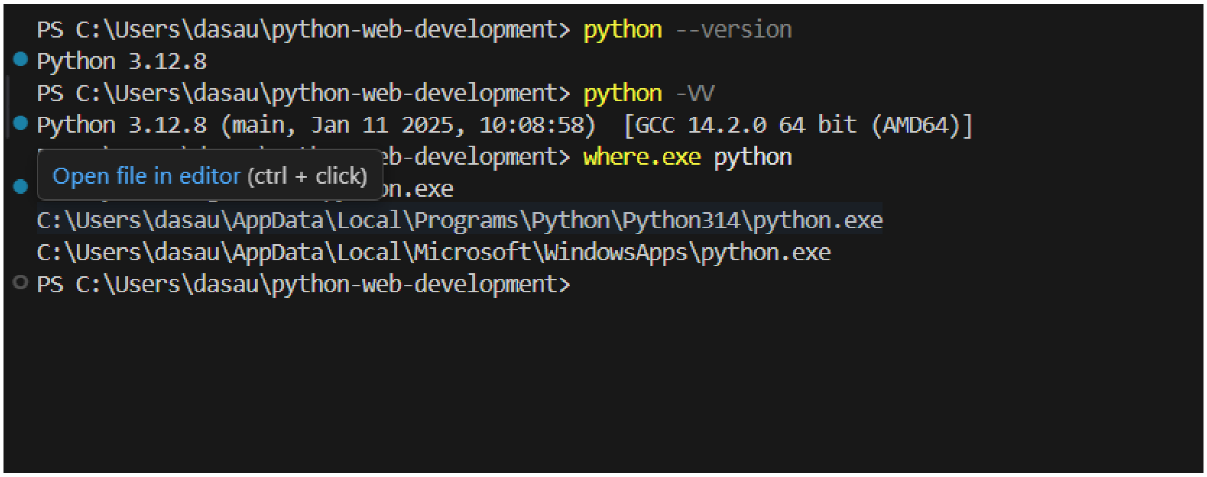
- **Scenario A: You need to develop a web application with multiple users.**
  - **Answer: Yes.** Django is an excellent choice. It comes with a powerful, built-in user authentication system that handles user accounts, groups, permissions, and session management out-of-the-box, making it very straightforward to manage multiple users securely.
- **Scenario B: You need fast deployment and the ability to make changes as you proceed.**
  - **Answer: Yes.** Django's MVT architecture and "batteries-included" nature are designed for rapid development. Its clean structure and ORM allow for quick prototyping and easy iterations, which is ideal for projects that evolve over time.
- **Scenario C: You need to build a very basic application, which doesn’t require any database access or file operations.**
  - **Answer: Probably not.** For a very simple, single-page application with no database needs, a micro-framework like Flask would be more lightweight and appropriate. Using Django would introduce unnecessary complexity and overhead for such a basic task.
- **Scenario D: You want to build an application from scratch and want a lot of control over how it works.**
  - **Answer: It depends, but likely no.** Django has strong opinions on how projects should be structured (the "Django way"). If you require fine-grained control over every component (e.g., you want to use a non-standard database layer or a unique URL routing system), a less opinionated framework like Flask or Pyramid would offer more flexibility.
- **Scenario E: You’re about to start a big project and are afraid of getting stuck and needing additional support.**

- **Answer: Yes.** Django has one of the largest and most active communities in the web development world. It is backed by extensive, well-written documentation, numerous third-party packages, and many tutorials. This makes it an excellent choice for large projects where reliable support is crucial.

Part 2: Setup and Installation Screenshots

1. Python Version Check

- **Command Used:** `python –version 3.14`
- **Screenshot:**



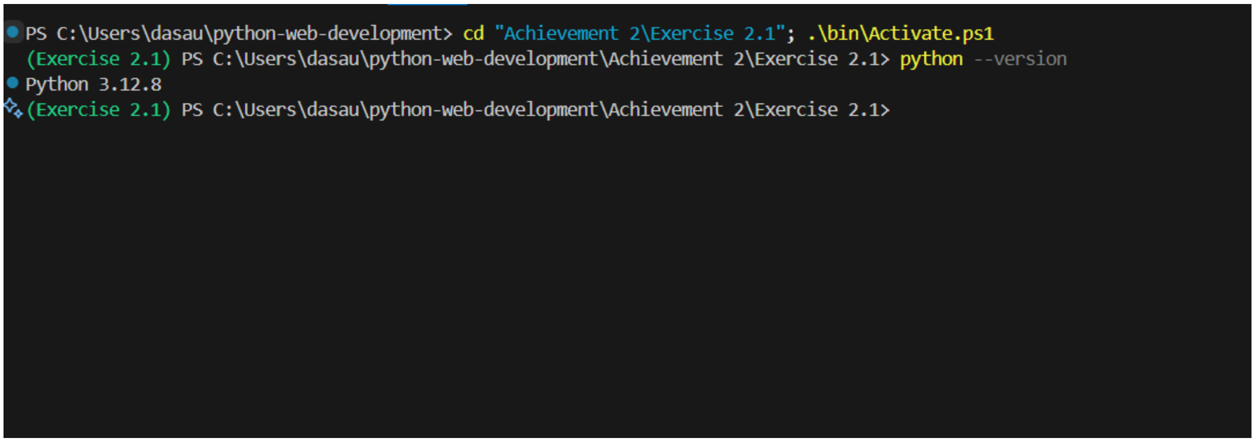
```
  PS C:\Users\dasau\python-web-development> python --version
● Python 3.12.8
  PS C:\Users\dasau\python-web-development> python -VV
● Python 3.12.8 (main, Jan 11 2025, 10:08:58)  [GCC 14.2.0 64 bit (AMD64)]
                                          eb-development> where.exe python
●                                         on.exe
  C:\Users\dasau\AppData\Local\Programs\Python\Python314\python.exe
  C:\Users\dasau\AppData\Local\Microsoft\WindowsApps\python.exe
○ PS C:\Users\dasau\python-web-development>
```

2. Virtual Environment Activation

- **Commands Used:**
  - `mkvirtualenv achievement2-practice` (to create)
  - `workon achievement2-practice` (to activate on Mac/Lin

3. Django Installation and Verification

- **Commands Used:**
  - `pip install django` (to install)
  - `django-admin --version` 5.2.7
- **Screenshot**



```
PS C:\Users\dasau\python-web-development> cd "Achievement 2\Exercise 2.1"; .\bin\Activate.ps1
(Exercise 2.1) PS C:\Users\dasau\python-web-development\Achievement 2\Exercise 2.1> python --version
Python 3.12.8
(Exercise 2.1) PS C:\Users\dasau\python-web-development\Achievement 2\Exercise 2.1>
```

Learning Journal Entry for Achievement 2, Exercise 2.1

Reflection on Achievement 1 and Plan for Achievement 2

Was your study routine effective during Achievement 1? If not, what will you do differently during Achievement 2?

My study routine in Achievement 1 was generally effective, but I sometimes fell into the trap of "tutorial hell," following along without always experimenting enough on my own. For Achievement 2, I will adopt a more proactive approach. After going through the material, I will immediately try to build small features or modify the code without the guide to solidify my understanding before applying it to the main project.

Reflect on your learning and project work for Achievement 1. What were you most proud of? How will you repeat or build on this in Achievement 2?

I was most proud of successfully building the command-line Recipe application and integrating it with a MySQL database. It was rewarding to see the OOP principles and SQL queries come together into a functional program. In Achievement 2, I will build on this by taking that same data model and logic and transforming it into a full-stack web application using Django, focusing on how the backend concepts I've already learned translate into a web framework context.

What difficulties did you encounter in the last Achievement? How did you deal with them? How could this experience prepare you for difficulties in Achievement 2?

The main difficulty was debugging database connection issues and complex SQL queries. I dealt with them by using print statements for debugging, breaking down problems into smaller steps, and extensively searching documentation and Stack Overflow. This experience has prepared me to be more patient and systematic when debugging. I anticipate that learning a large framework like Django will have its own challenges, but I now feel more confident in my ability to research solutions, read error messages carefully, and seek help when truly stuck.