Web Development & Django

Exercise 2.1 Task

1. Why is Django is so popular among web developers?

   Using Django for Python projects allows you to focus on the logic and leave lower-level operations to the framework. The framework takes care of the heavy lifting (like database access, templates, user-session management) and lets you build apps quickly with less code. It's ease of use enables rapid development and clean design. Developers can automate as much as possible and follow the DRY principle, which can help you avoid bugs in the future. As part of it's "batteries included" structure, Django provides a fully functional ORM right out of the box – making for simple and straightforward relational database management.

2. Five large companies that use Django, the company's product or service and what they use Django for:

Company	Product/Service	Django Use
YouTube	Video Sharing	Django enables YouTube to handle high traffic demands and support millions of data requests daily.
Pinterest	Social Media	Django allows for a high-performing, fast and scalable site. It handles thousands of users simultaneously and enables the management of followers, boards, pins, and posts.
Dropbox	Cloud storage	Developers use Django to quickly add features such as user history access, version control option, account synchronization across multiple devices, and file-sharing services.
Washington Post	News/Media	The Django CMS handles large amounts of data generated by the Posts' daily audience and offers a highly scalable, fast application.

Spotify	Music	Django's fast backend and machine learning capabilities are ideal for the app's ability to handle large amounts of data, create personalized playlists and recommendations.

3. For each of the following scenarios, explain if you would use Django (and why or why not):

   a. You need to develop a web application with multiple users. Yes, I'd use Django. It's a good choice for web apps, given the benefits noted in question 1 above. Security and authentication are built into the framework ("batteries included"), so setting up multiple users is a simple process that will require less code than if you were to do it from scratch.

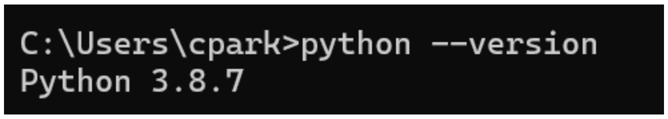
   b. You need fast deployment and the ability to make changes as you proceed. Yes, I'd use Django. It enables fast deployment and apps built with it are scalable, so this would make it a good choice for this scenario. DRY principles followed by Django also ensure that fewer bugs will come from any changes down the line as changes will have to be made in fewer places.

   c. You need to build a very basic application, which doesn't require any database access or file operations. No, Django is likely overkill for this. A micro-framework like Flask might be more suitable.

   d. You want to build an application from scratch and want a lot of control over how it works. No, Django is a strict, highly structured framework and has rules that must be followed. So, if you want a lot of control, Django isn't ideal.

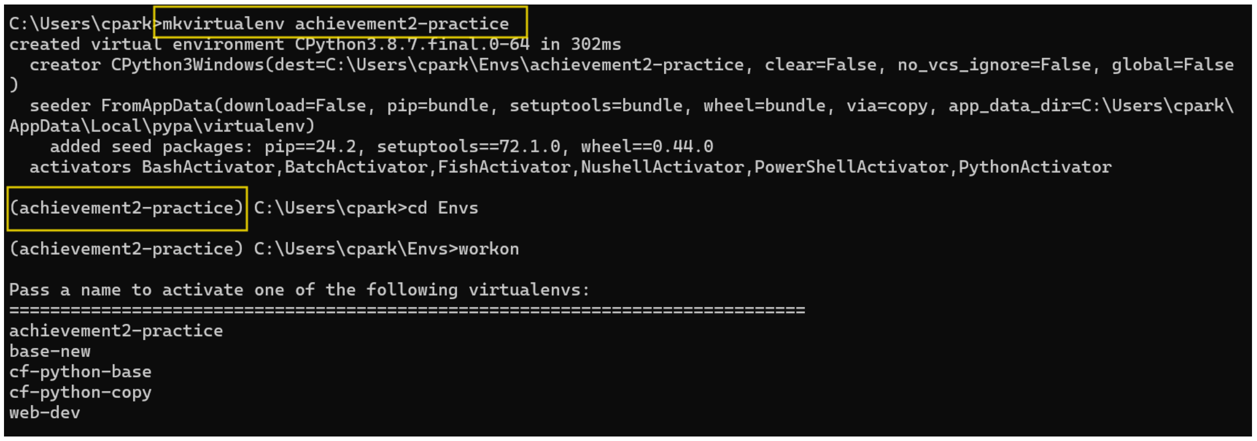
   e. You're about to start working on a big project and are afraid of getting stuck and needing additional support. Yes, I'd use Django. It's open source and increasingly popular. As a result, it has a large community and lots of support, so if you do get stuck, you're more likely to find help quickly either through its official documentation or forums like Stack Overflow.

4. Check Python version:



```
C:\Users\cpark>python --version
Python 3.8.7
```

5. Set up and create a virtual environment:



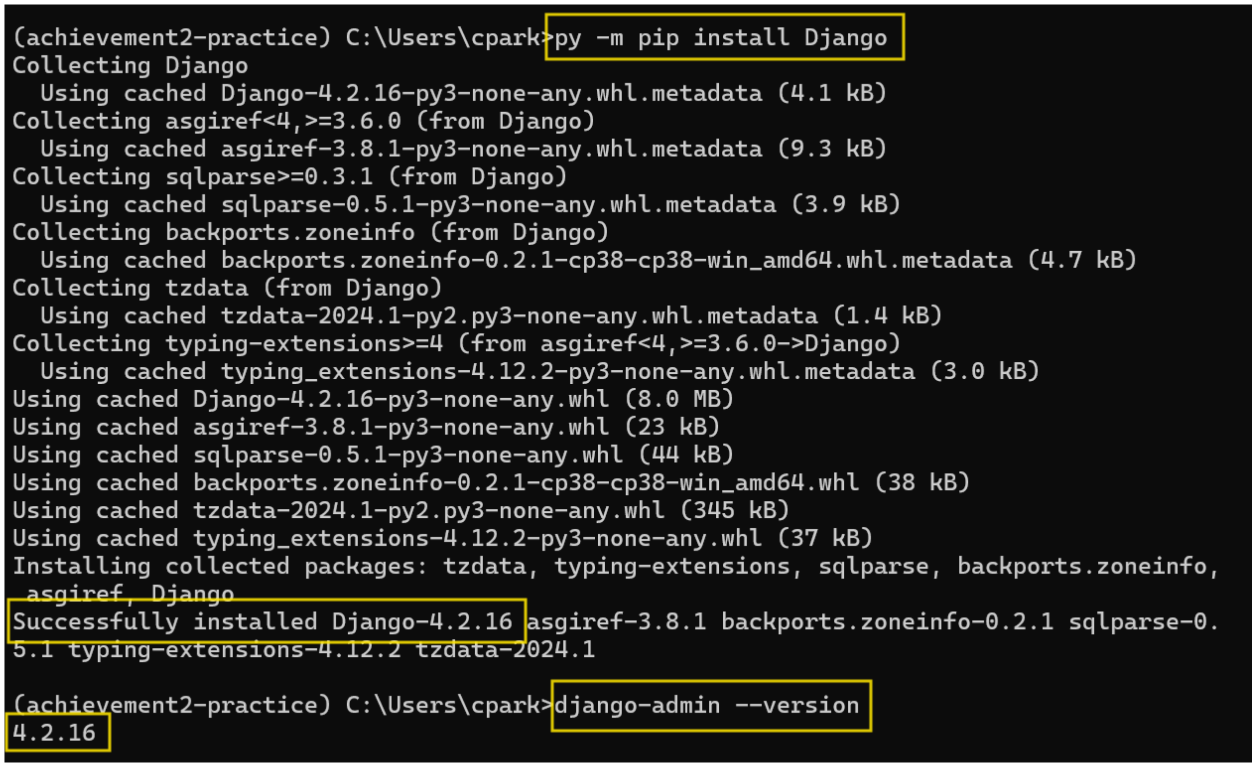
```
C:\Users\cpark>mkvirtualenv achievement2-practice
created virtual environment CPython3.8.7.final.0-64 in 302ms
  creator CPython3Windows(dest=C:\Users\cpark\Envs\achievement2-practice, clear=False, no_vcs_ignore=False, global=False
)
  seeder FromAppData(download=False, pip=bundle, setuptools=bundle, wheel=bundle, via=copy, app_data_dir=C:\Users\cpark\
AppData\Local\pypa\virtualenv)
    added seed packages: pip==24.2, setuptools==72.1.0, wheel==0.44.0
  activators BashActivator,BatchActivator,FishActivator,NushellActivator,PowerShellActivator,PythonActivator

(achievement2-practice) C:\Users\cpark>cd Envs

(achievement2-practice) C:\Users\cpark\Envs>workon

Pass a name to activate one of the following virtualenvs:
==============================================================================
achievement2-practice
base-new
cf-python-base
cf-python-copy
web-dev
```

6. Install Django and verify version:



```
(achievement2-practice) C:\Users\cpark>py -m pip install Django
Collecting Django
  Using cached Django-4.2.16-py3-none-any.whl.metadata (4.1 kB)
Collecting asgiref<4,>=3.6.0 (from Django)
  Using cached asgiref-3.8.1-py3-none-any.whl.metadata (9.3 kB)
Collecting sqlparse>=0.3.1 (from Django)
  Using cached sqlparse-0.5.1-py3-none-any.whl.metadata (3.9 kB)
Collecting backports.zoneinfo (from Django)
  Using cached backports.zoneinfo-0.2.1-cp38-cp38-win_amd64.whl.metadata (4.7 kB)
Collecting tzdata (from Django)
  Using cached tzdata-2024.1-py2.py3-none-any.whl.metadata (1.4 kB)
Collecting typing-extensions>=4 (from asgiref<4,>=3.6.0->Django)
  Using cached typing_extensions-4.12.2-py3-none-any.whl.metadata (3.0 kB)
Using cached Django-4.2.16-py3-none-any.whl (8.0 MB)
Using cached asgiref-3.8.1-py3-none-any.whl (23 kB)
Using cached sqlparse-0.5.1-py3-none-any.whl (44 kB)
Using cached backports.zoneinfo-0.2.1-cp38-cp38-win_amd64.whl (38 kB)
Using cached tzdata-2024.1-py2.py3-none-any.whl (345 kB)
Using cached typing_extensions-4.12.2-py3-none-any.whl (37 kB)
Installing collected packages: tzdata, typing-extensions, sqlparse, backports.zoneinfo,
 asgiref, Django
Successfully installed Django-4.2.16 asgiref-3.8.1 backports.zoneinfo-0.2.1 sqlparse-0.
5.1 typing-extensions-4.12.2 tzdata-2024.1

(achievement2-practice) C:\Users\cpark>django-admin --version
4.2.16
```