Coming with batteries included is a large part of why Django is so popular with web developers. It means that devs don't have to spend any time reinventing the wheel, and can instead use a huge variety of prebuilt functions. This creates a high level of scalability. Partially because of its popularity, it also has a large community and thorough documentation.

Companies that use Django:
1. Instagram is likely the most high-profile site using Django, and it also uses Python. The scalability that Django provides is what allows Instagram to process such enormous amounts of data and user interaction.

2. Mozilla Firefox also uses Django/Python. Typically it uses them more on the MDN/support site side.

3. National Geographic uses Django on its website as the main backend.

4. Indeed uses Django to manage large amounts of changing data such as job inquiries, vacancies, and exchanges between users.

5. Opera rivals Mozilla as browser that uses Django to manage web traffic across millions of active users.

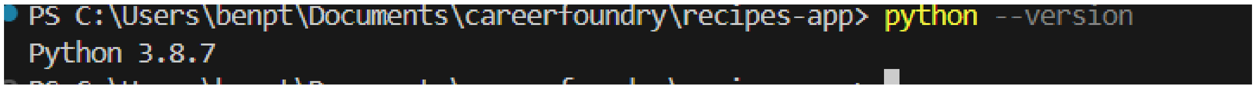
Number 4 questions:
1. Django supports multiple user and multiple user types (i.e. admin, client, etc). This makes it viable for a web app that requires multiple users. Also, a login app can easily be shared between different projects.

2. Django develops and deploys extremely quickly, thanks to being considered batteries-included. It would be perfect for a project like that.

3. Django is considerably too heavy for a very basic application -- it would eat up bandwidth with not enough return on investment.

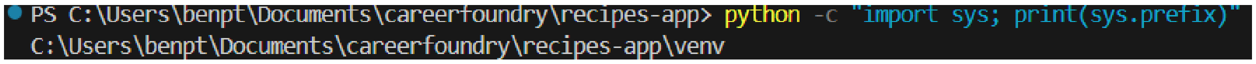
4. Django takes a lot of control out of developers' hands, because it expects things in a very specific way. It would not work for an app whose developer wants tight reins on.

5. Django's large community and thorough documentation would provide sufficient support for a developer who is uncertain of their own skills.



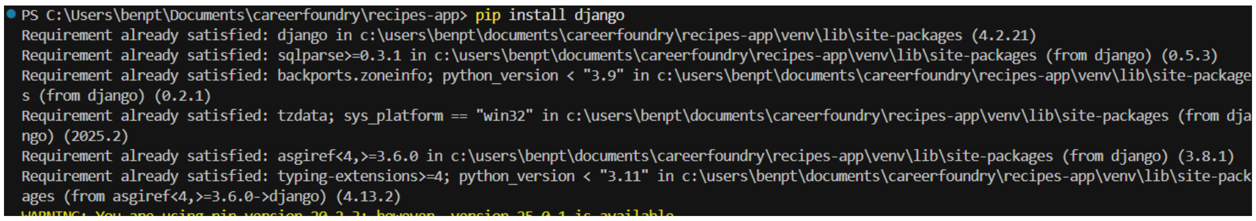
```
PS C:\Users\benpt\Documents\careerfoundry\recipes-app> python --version
Python 3.8.7
```

Python runs at the correct version.



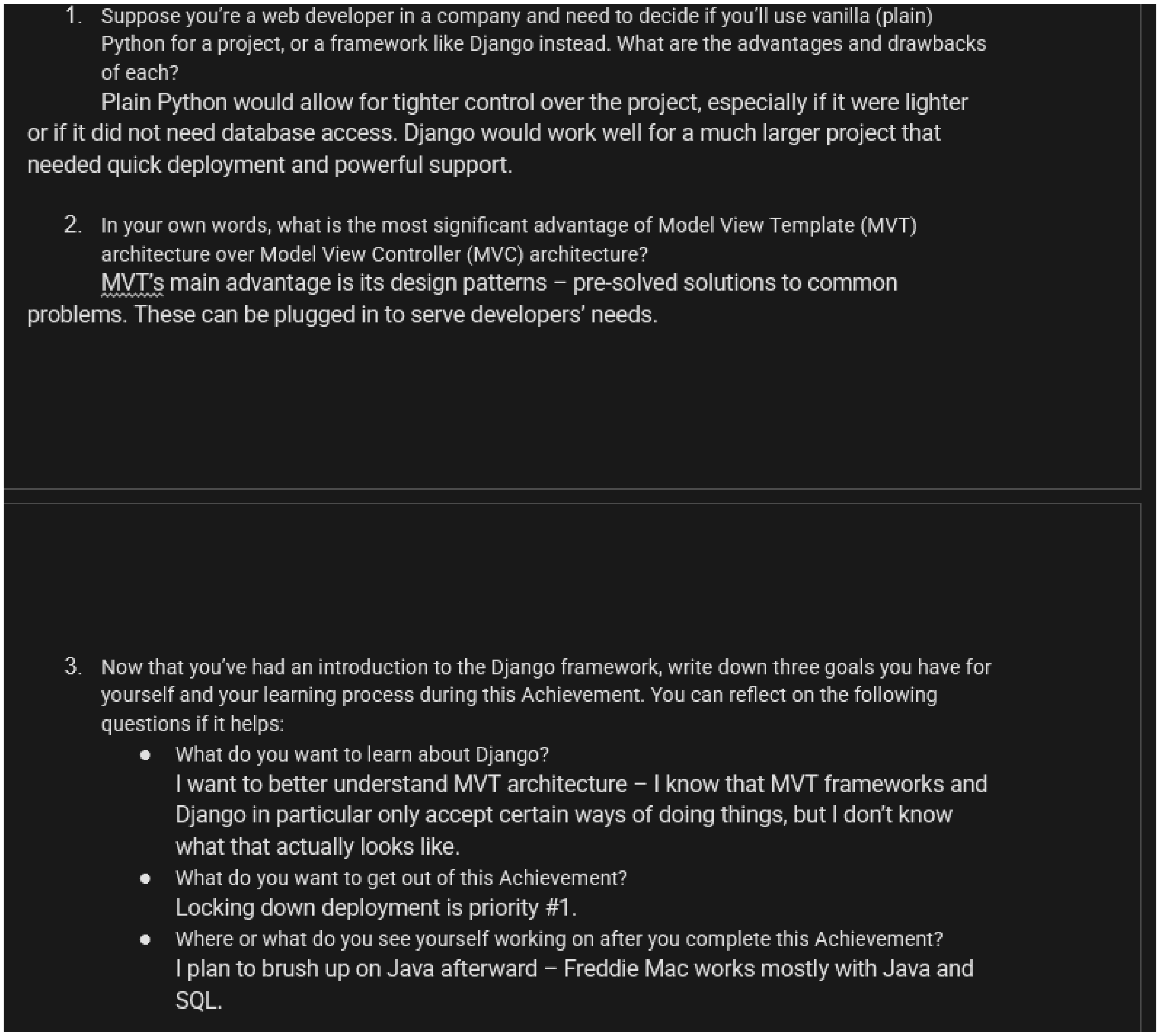
```
PS C:\Users\benpt\Documents\careerfoundry\recipes-app> python -c "import sys; print(sys.prefix)"
C:\Users\benpt\Documents\careerfoundry\recipes-app\venv
```

Note: it gave me a notification saying that although it was running in venv, the (venv) prefix would not pop up. This command demonstrates that the virtual environment is running.



```
PS C:\Users\benpt\Documents\careerfoundry\recipes-app> pip install django
Requirement already satisfied: django in c:\users\benpt\documents\careerfoundry\recipes-app\venv\lib\site-packages (4.2.21)
Requirement already satisfied: sqlparse>=0.3.1 in c:\users\benpt\documents\careerfoundry\recipes-app\venv\lib\site-packages (from django) (0.5.3)
Requirement already satisfied: backports.zoneinfo; python_version < "3.9" in c:\users\benpt\documents\careerfoundry\recipes-app\venv\lib\site-package
s (from django) (0.2.1)
Requirement already satisfied: tzdata; sys_platform == "win32" in c:\users\benpt\documents\careerfoundry\recipes-app\venv\lib\site-packages (from dja
ngo) (2025.2)
Requirement already satisfied: asgiref<4,>=3.6.0 in c:\users\benpt\documents\careerfoundry\recipes-app\venv\lib\site-packages (from django) (3.8.1)
Requirement already satisfied: typing-extensions>=4; python_version < "3.11" in c:\users\benpt\documents\careerfoundry\recipes-app\venv\lib\site-pack
ages (from asgiref<4,>=3.6.0->django) (4.13.2)
WARNING: You are using pip version 20.2.3; however, version 25.0.1 is available.
```

Django is installed.



1. Suppose you're a web developer in a company and need to decide if you'll use vanilla (plain) Python for a project, or a framework like Django instead. What are the advantages and drawbacks of each?

   Plain Python would allow for tighter control over the project, especially if it were lighter or if it did not need database access. Django would work well for a much larger project that needed quick deployment and powerful support.

2. In your own words, what is the most significant advantage of Model View Template (MVT) architecture over Model View Controller (MVC) architecture?

   MVT's main advantage is its design patterns – pre-solved solutions to common problems. These can be plugged in to serve developers' needs.

3. Now that you've had an introduction to the Django framework, write down three goals you have for yourself and your learning process during this Achievement. You can reflect on the following questions if it helps:
   - What do you want to learn about Django?
     I want to better understand MVT architecture – I know that MVT frameworks and Django in particular only accept certain ways of doing things, but I don't know what that actually looks like.
   - What do you want to get out of this Achievement?
     Locking down deployment is priority #1.
   - Where or what do you see yourself working on after you complete this Achievement?
     I plan to brush up on Java afterward – Freddie Mac works mostly with Java and SQL.

Learning journal