CAREERFOUNDRY

Python for Web Developers Learning Journal

Objective

We find that the students who do particularly well in our courses are those who practice metacognition. Metacognition is the art of thinking about thinking; developing a deeper understanding of your own

thought processes. With the help of this Learning Journal, you'll broaden your metacognitive knowledge and skills by reflecting on what you learn in this course.

Thanks to this Learning Journal, when you finish the course you'll have a complete and detailed record of your learning journey and progress over time. We really recommend that you take the time to complete this Journal; students do better in CF courses and in the working world as a result!

Directions

First complete the pre-work section before you start your course. Then, once you've begun learning, take time after each Exercise to return to this Journal and respond to the prompts.

There will be 3 to 5 prompts per Exercise, and we recommend spending about 10 to 15 minutes in total answering them. Don't overthink it—just write whatever comes to mind!

Also make sure that, once you've started filling this document in, you upload it as a deliverable on the platform. This is so that your mentor can also see your Journal and how you're progressing over time. Don't worry though—what you write here won't affect how you're graded for the Exercise tasks. The learning journal is mostly for you and your self-evaluation!

Pre-Work: Before You Start the Course

Reflection questions (to complete before your first mentor call)

1. What experiences have you had with coding and/or programming so far? What other experiences (programming-related or not) have you had that may help you as you progress through this course?
    a. My primary coding experience comes from full-stack development, particularly working with React and Angular on the frontend. I've also worked with GDScript for game development, which has given me exposure to object-oriented programming concepts. These experiences provide a decent foundation for learning Python, as I already understand core programming concepts like functions, loops, and data structures. My frontend experience will be particularly valuable when working on web-related Python projects.
2. What do you know about Python already? What do you want to know?

    a. My current Python knowledge comes from exposure to its syntax through GDScript and general awareness of its popularity in data analysis. I understand Python's basic syntax but want to deepen my knowledge of its data analysis capabilities, specifically libraries like Pandas and NumPy. I'm also interested in learning how Python handles backend web development compared to other technologies I've used.

3. What challenges do you think may come up while you take this course? What will help you face them? Think of specific spaces, people, and times of day of week that might be favorable to your facing challenges and growing. Plan for how to solve challenges that arise.
    a. The main challenges I anticipate are maintaining a consistent learning schedule while balancing other commitments. However, I have a solid plan in place - dedicating morning and afternoon time blocks for focused learning. Having friends in the industry is a valuable resource - they can provide technical guidance and real-world perspectives when I encounter difficulties.

Remember, you can always refer to Exercise 1.4 of the Orientation course if you're not sure whom to reach out to for help and support.

Exercise 1.1: Getting Started with Python

Learning Goals

- Summarize the uses and benefits of Python for web development
- Prepare your developer environment for programming with Python

Reflection Questions

1. In your own words, what is the difference between frontend and backend web development? If you were hired to work on backend programming for a web application, what kinds of operations would you be working on?
    a. Frontend development focuses on what users see and interact with in their browser or apps (like buttons, forms, and layouts), while backend development handles server-side operations hidden from users. As a backend programmer, I would work on tasks like database management, server logic, API development, and processing data between the server and database - essentially handling the "behind-the-scenes" functionality that powers the application.

2. Imagine you're working as a full-stack developer in the near future. Your team is asking for your advice on whether to use JavaScript or Python for a project, and you think Python would be the

better choice. How would you explain the similarities and differences between the two languages to your team? Drawing from what you learned in this Exercise, what reasons would you give to convince your team that Python is the better option?

   a. While both JavaScript and Python are versatile languages, Python stands out for its clear, readable syntax and extensive data libraries. I would explain to my team that Python's simplicity makes it easier to maintain and debug, while its robust ecosystem of tools (like NumPy and Pandas) makes it ideal for complex data operations. Python's "batteries included" philosophy means we'd spend less time configuring tools and more time solving actual problems.

3. Now that you've had an introduction to Python, write down 3 goals you have for yourself and your learning during this Achievement. You can reflect on the following questions if it helps you. What do you want to learn about Python? What do you want to get out of this Achievement? Where or what do you see yourself working on after you complete this Achievement?
   a. My goals for this Achievement are: First, I want to deepen my understanding of Python as a backend technology, particularly how it handles data flow in full-stack applications. Second, I aim to expand my foundational knowledge of Python's data analysis capabilities into practical expertise. Finally, I want to apply these skills by building a production-ready Python application that demonstrates my capabilities to potential employers.

Exercise 1.2: Data Types in Python

Learning Goals

- Explain variables and data types in Python
- Summarize the use of objects in Python
- Create a data structure for your Recipe app

Reflection Questions

1. Imagine you're having a conversation with a future colleague about whether to use the iPython Shell instead of Python's default shell. What reasons would you give to explain the benefits of using the iPython Shell over the default one?
   a. The IPython Shell offers significant advantages over Python's default shell with syntax highlighting that makes code more visually clear and readable, plus automatic indentation that makes writing nested code much easier. It's also more efficient for testing small pieces of code since commands execute immediately and responses print straight away, making it much more practical than creating separate script files for quick tests.

2. Python has a host of different data types that allow you to store and organize information. List 4 examples of data types that Python recognizes, briefly define them, and indicate whether they are scalar or non-scalar.

Data type	Definition	Scalar or Non-Scalar?
Bool	True or false value	Yes
Int	Integer number value	Yes
Tuples	Linear arrays storing any values	No
Dictionaries	List of key value pairs (like JSON)	No

3. A frequent question at job interviews for Python developers is: what is the difference between lists and tuples in Python? Write down how you would respond.
    a. The key difference is that lists are mutable while tuples are immutable - meaning you can modify, add, or remove elements in a list after creation, but a tuple's contents cannot be changed once created. Lists are defined using square brackets [] and tuples use parentheses (), with lists being better for modifiable collections and tuples being more efficient for fixed data since Python knows their size won't change.
4. In the task for this Exercise, you decided what you thought was the most suitable data structure for storing all the information for a recipe. Now, imagine you're creating a language-learning app that helps users memorize vocabulary through flashcards. Users can input vocabulary words, definitions, and their category (noun, verb, etc.) into the flashcards. They can then quiz themselves by flipping through the flashcards. Think about the necessary data types and what would be the most suitable data structure for this language-learning app. Between tuples, lists, and dictionaries, which would you choose?
    a. Dictionaries would be best for the individual "cards" as each vocab word requires several key value pairs. The "deck" of flashcards would be a list of the dictionaries. Like this:
    ```
    flashcard_1 = { "word": "correr", "definition": "to run", "category": "verb" }
    flashcard_2 = { "word": "comer", "definition": "to eat", "category": "verb" }
    flashcard_deck = [flashcard_1, flashcard_2]
    ```

Exercise 1.3: Functions and Other Operations in Python

Learning Goals

- Implement conditional statements in Python to determine program flow
- Use loops to reduce time and effort in Python programming
- Write functions to organize Python code

Reflection Questions

1. In this Exercise, you learned how to use **if-elif-else** statements to run different tasks based on conditions that you define. Now practice that skill by writing a script for a simple travel app using

an **if-elif-else** statement for the following situation:

- The script should ask the user where they want to travel.
- The user's input should be checked for 3 different travel destinations that you define.
- If the user's input is one of those 3 destinations, the following statement should be printed: "Enjoy your stay in _____!"
- If the user's input is something other than the defined destinations, the following statement should be printed: "Oops, that destination is not currently available."

Write your script here. *(Hint: remember what you learned about indents!)*

```
destination_list = ['Morocco', 'France', 'Ireland']
user_destination = input("What country would you like to go to?")

if user_destination in destination_list:
    print("Enjoy your stay in " + user_destination + "!")
else:
    print("Oops, that destination is currently unavailable")
```

2. Imagine you're at a job interview for a Python developer role. The interviewer says "Explain logical operators in Python". Draft how you would respond.
    a. Python has three main Boolean logical operators: 'and' (returns True only if both operands are True), 'or' (returns True if at least one operand is True), and 'not' (inverts the Boolean value). Python has a unique feature where these operators return the actual operand values rather than just True/False.

3. What are functions in Python? When and why are they useful?
    a. Functions in Python are reusable blocks of code that perform specific tasks. They're useful for code organization, reusability, and abstraction - allowing you to write code once and use it multiple times while hiding complex implementation details behind a simple interface.

4. In the section for Exercise 1 in this Learning Journal, you were asked in question 3 to set some goals for yourself while you complete this course. In preparation for your next mentor call, make some notes on how you've progressed towards your goals so far.
    a. After the previous two assignments I feel like I understand the basics of python as a programming language, most of which is directly useful for manipulating data. So I already have a concept of how powerful the language will be for data analysis. Great progress in my opinion so far!

Exercise 1.4: File Handling in Python

Learning Goals

- Use files to store and retrieve data in Python

Reflection Questions

1. Why is file storage important when you're using Python? What would happen if you didn't store local files?
    a. File storage is crucial because it is how data is stored when the script isn't running. If the the data wasn't exported to a bin, then data would be lost upon terminating the script.

2. In this Exercise you learned about the pickling process with the **pickle.dump()** method. What are pickles? In which situations would you choose to use pickles and why?
    a. Pickles are objects that are formatted with a special binary format. Pickles are often used to open and edit files and data. They are also great when working with more complex data structures, since they will retain the exact data relationships and organization when manipulating them between scripts.

3. In Python, what function do you use to find out which directory you're currently in? What if you wanted to change your current working directory?
    a. Os.getcwd() is the function that outputs your current working directory. To change your current working directory, you would use os.chdir(<path to target folder>)

4. Imagine you're working on a Python script and are worried there may be an error in a block of code. How would you approach the situation to prevent the entire script from terminating due to an error?
    a. Using try, except, else, finally blocks is a great way to handle any errors in code that is likely to produce errors (i.e. via user misinput.) There are specific formats built in (like FileNotFound) for catching common errors. The main strengh of try blocks is that it allows you to account for scenarios in which the app would crash.

5. You're now more than halfway through Achievement 1! Take a moment to reflect on your learning in the course so far. How is it going? What's something you're proud of so far? Is there something you're struggling with?
    a. I'm actually feeling really confident. Compared to JavaScript/TypeScript, this code is waaaaay easier to understand. It feels like all the tools I need are built right in, straightforward and easy to implement. Remembering all the different methods and syntax is tough, but that comes with time!

Exercise 1.5: Object-Oriented Programming in Python

Learning Goals

- Apply object-oriented programming concepts to your Recipe app

Reflection Questions

1. In your own words, what is object-oriented programming? What are the benefits of OOP?
    a. OOP is a paradigm for programming that structures/organizes code based on objects. Those objects can be data or functions. Not only in this more intuitive (objects are analogous to real-world logic) this allows for more reusable/organized code (through inheritance.)

2. What are objects and classes in Python? Come up with a real-world example to illustrate how objects and classes work.
    a. A class is essentially an object blueprint that defines the properties and functions of all objects that belong to said class. An object is an instantiation of that class. For example, Reptiles are a class. Snakes and Crocodiles are members of that class and share certain reptilian characteristics.

3. In your own words, write brief explanations of the following OOP concepts; 100 to 200 words per method is fine.

Method	Description
Inheritance	Inheritance simply means certain objects inherit properties/methods from their class. Using the previous example of Reptiles as a class, reptiles have several core characteristics that all reptiles share (scales, cold-blood, egg-laying, etc.) So all individual reptiles (snake, crocodile) also have these qualities. Beyond on that, the individual instances of a class have their own features; snakes for instance have no legs
Polymorphism	Polymorphism allows different classes to have methods of the same name that behave differently. For instance, if we have a snake and crocodile object, we could call the move method for both but with different outcomes. For instance, the snake would slither forward, whereas the crocodile would walk or swim.
Operator Overloading	Operator overloading allows you to define how operators (+, -, *, etc.) behave with objects of your class. For example, when you add two strings with +, you get concatenation. When you add two numbers, you get addition. By overloading operators, you can define what these operations mean for your custom objects. This makes code more intuitive and readable, as you can use natural operators instead of method calls.

Exercise 1.6: Connecting to Databases in Python

Learning Goals

- Create a MySQL database for your Recipe app

Reflection Questions

1. What are databases and what are the advantages of using them?
    a. Databases are organized collections of structured data. The main advantages of using a database is that it can handle (with a server/API) multiple users simultaneously, serves as a centralized source of truth (instead of multiple data sources) and can easily be secured and backed-up.

2. List 3 data types that can be used in MySQL and describe them briefly:

Data type	Definition
INT	A positive or negative whole number
FLOAT	A positive or negative number with a decimal point.
VARCHAR(#)	A string with a limited length.

3. In what situations would SQLite be a better choice than MySQL?
    a. SQLite is ideal for smaller apps, local apps, developing, testing, and shorter term projects.

4. Think back to what you learned in the Immersion course. What do you think about the differences between JavaScript and Python as programming languages?
    a. Instead of using brackets, braces, semicolons, etc, python uses identing, which is much more readable and organized.
    b. Python is also more of a general purpose language that can be used in data science, backend and automation, whereas JavaScript is mostly for front end.
    c. Python is also much more user friendly and therefore easier to learn.

5. Now that you're nearly at the end of Achievement 1, consider what you know about Python so far. What would you say are the limitations of Python as a programming language?
    a. Python's limitations compared to JavaScript are that it can't run directly in web browsers and needs extra tools (IDjango) to build websites, while JavaScript can run right in the browser and is built specifically for web development. JavaScript is also better at handling real-time updates on websites (like chat messages or live updates) and has

more ready-to-use tools for building modern web interfaces, where as Python is mainly used for behind the scenes processing and data analysis.

Exercise 1.7: Finalizing Your Python Program

Learning Goals

- Interact with a database using an object-relational mapper
- Build your final command-line Recipe application

Reflection Questions

1. What is an Object Relational Mapper and what are the advantages of using one?
    a. Object-Relational Mappers allow devs to work with data as objects rather than directly manipulating the database structure. Using an ORM like SQLAlchemy provides benefits such asobject-oriented programming, automatic query generation, and easier data manipulation.

2. By this point, you've finished creating your Recipe app. How did it go? What's something in the app that you did well with? If you were to start over, what's something about your app that you would change or improve?
    a. The app completes all functions correctly, but I think some enhancements would be good. I would add how much is needed for each ingredient (i.e. 1 lemon) so this would require an additional variable to store, perhaps in a tuple. I would also differentiate cooking time and prep time as well as method of adding recipe instructions via user input.

3. Imagine you're at a job interview. You're asked what experience you have creating an app using Python. Taking your work for this Achievement as an example, draft how you would respond to this question.
    a. My experience with Python thus far has been through the development of this Recipe App. I've gained valuable skills in using SQLAlchemy, a powerful Object-Relational Mapping library, to interact with a database. I've learned how to design database models, map them to Python classes, and perform common CRUD operations. Additionally, I've demonstrated proficiency in building robust command-line interfaces with user input validation, modularizing code, and applying object-oriented programming concepts
4. You've finished Achievement 1! Before moving on to Achievement 2, take a moment to reflect on your learning in the course so far:
    a. What went well during this Achievement?
        i. Learning python. It's really straight forward and very easy to understand the syntax.
    b. What's something you're proud of?

    i. Much of the code I coded from memory and my own understanding. I made mistakes often but my initial attempts were fairly accurate.
c. What was the most challenging aspect of this Achievement?
    i. Understanding what “f” mean in the context of say “f"Recipe ID: {self.id}\n.” There were a number of times when my attempts were totally off because I didn’t understand why or how to use this syntax.
d. Did this Achievement meet your expectations? Did it give you the confidence to start working with your new Python skills?
    i. Yeah! I feel like I have a strong foundation of python skills.
e. What’s something you want to keep in mind to help you do your best in Achievement 2?
    i. I’m really curious how this relates to backend and front end and how I might build a UI for the app instead of a CLI.

Well done—you’ve now completed the Learning Journal for Achievement 1. As you’ll have seen, a little metacognition can go a long way!

Pre-Work: Before You Start Achievement 2

In the final part of the learning journal for Achievement 1, you were asked if there’s anything—on reflection—that you’d keep in mind and do similarly or differently during Achievement 2. Think about these questions again:

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

Note down your answers and discuss them with your mentor in a call if you like.

Remember that can always refer to Exercise 1.4 of the Orientation course if you’re not sure whom to reach out to for help and support.

Exercise 2.1: Getting Started with Django

Learning Goals

- Explain MVT architecture and compare it with MVC
- Summarize Django’s benefits and drawbacks
- Install and get started with Django

Reflection Questions

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?
    a. Vanilla Python:
    Advantages: It gives you complete control over how things work, so you can build the project exactly the way you want it. It's also lighter and can be faster for very small projects where you don't need a lot of built-in features.
    Drawbacks: You'll have to build everything from scratch, which can be time-consuming, and you'll need to handle things like security, database management, and user authentication yourself.
    b. Django:
    Advantages: Django provides lots of built-in tools, like authentication, database management, and an admin panel, so it saves a lot of time. It also has great security features and follows best practices.
    Drawbacks: Django can feel restrictive at times because it comes with its own structure and way of doing things, which might not be ideal for projects where you need complete flexibility.

2. In your own words, what is the most significant advantage of Model View Template (MVT) architecture over Model View Controller (MVC) architecture?
    A. The biggest advantage of Model View Template (MVT) over Model View Controller (MVC) architecture is how the template is separated from the views and controllers, making it easier to manage user interfaces and business logic separately. MVT allows the template to handle presentation more clearly, while the controller (called a view in Django) can focus on processing data and responding to requests. This separation can make Django apps easier to maintain and scale.
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 learn the advantages of using Django/Python as a backend vs. javascript/node.
    - What do you want to get out of this Achievement?
        - Enough Django/Python skills to be hired as a dev specializing in them.
    - Where or what do you see yourself working on after you complete this Achievement?
        - I am currently working on my own React Native app with a Node.js backend. I am considering switching over to Django for my backend

Exercise 2.2: Django Project Set Up

Learning Goals

- Describe the basic structure of a Django project
- Summarize the difference between projects and apps
- Create a Django project and run it locally
- Create a superuser for a Django web application

Reflection Questions

1. Suppose you're in an interview. The interviewer gives you their company's website as an example, asking you to convert the website and its different parts into Django terms. How would you proceed? For this question, you can think about your dream company and look at their website for reference.
   (*Hint: In the Exercise, you saw the example of the CareerFoundry website in the Project and Apps section.*)

   If I were in an interview and asked to convert YouTube's website into Django terms, YouTube's website as a whole would be a project. Inside this project, there would be several apps to manage different features of the platform. For example, a Videos app would handle uploading, storing, and displaying videos, with models for video files and details like title and description, and templates to show the video player. A Users app would manage user registration, login, and profiles, tracking user accounts, subscriptions, and personal settings. A Comments app would handle comments and replies on videos, while a Search app would provide search functionality to find videos, playlists, or channels. There could also be a Subscriptions app to manage user subscriptions to channels.

   The project's configuration files, like settings.py, would manage URL routing, global variables, and database settings, which could use PostgreSQL for scalability. The database would store data like video details, user information, comments, and subscriptions, with each app creating its own tables for specific models. Apps like the login app could be reused in other projects, such as a music streaming platform, reflecting Django's DRY principle. This structure makes the code modular, easier to develop, maintain, and reuse.

2. In your own words, describe the steps you would take to deploy a basic Django application locally on your system.
   a. To deploy a basic Django application locally on my system, I would follow these steps:
      i. Create a project folder and run a virtual environment.
      ii. Install Django: I'd make sure I have Python installed, then run pip install django to install Django.
      iii. Create a Project: Run django-admin startproject project_name to set up the project folder.
      iv. Create an App: Inside the project, I'd create an app with python manage.py startapp app_name.
      v. Update Settings: Add the new app to the INSTALLED_APPS list in the settings.py file.

vi. Set Up URLs: Configure the URLs by adding patterns in the urls.py file for the project and the app.
vii. Create Views and Templates: Write views in views.py and create templates in the templates folder to display content.
viii. Run the Server: Finally, run python manage.py runserver and go to http://localhost:8000 to see the app in the browser.

3. Do some research about the Django admin site and write down how you'd use it during your web application development.
    a. The Django admin site is a really useful tool that comes built-in with Django. It gives you a simple interface to manage your database, like adding, editing, or deleting records without having to write code. In the image, you can see options to manage **Users** and **Groups** under "Authentication and Authorization." The **"Add"** and **"Change"** buttons make it easy to create new users or update existing ones. When you're developing a web app, the admin site helps you quickly set up and edit data while you're testing or building features. For example, if you're working on a video-sharing app like YouTube, you could use the admin site to add new user accounts, manage video uploads, or moderate comments. It saves a lot of time and makes handling data super easy, especially in the early stages of development. Plus, you can customize the admin site to show what you need and hide anything you don't.

Exercise 2.3: Django Models

Learning Goals

- Discuss Django models, the "M" part of Django's MVT architecture
- Create apps and models representing different parts of your web application
- Write and run automated tests

Reflection Questions

1. Do some research on Django models. In your own words, write down how Django models work and what their benefits are.
    - Django models are like blueprints for your database tables that you write in Python instead of SQL. Makes it way easier to work with data since you don't have to write complex database queries. For iinstance, instead of dealing with SQL I can just make a Recipe model with fields for title and ingredients, and Django handles all the database stuff behind the scenes. Pretty neat since I can just use Python methods to save and fetch recipes!

2. In your own words, explain why it is crucial to write test cases from the beginning of a project. You can take an example project to explain your answer
    - Writing test cases from the start is crucial because it helps catch bugs early and ensures code works as intended as the project grows. For a recipe sharing app, you'd write tests to verify users can create recipes, ingredients are properly stored, and search functions return correct results. These tests serve as documentation and make it safer to modify code since you can quickly check if changes break existing functionality. Early testing also encourages better code design since testable code tends to be more modular and maintainable.

Exercise 2.4: Django Views and Templates

Learning Goals

- Summarize the process of creating views, templates, and URLs
- Explain how the “V” and “T” parts of MVT architecture work
- Create a frontend page for your web application

Reflection Questions

1. Do some research on Django views. In your own words, use an example to explain how Django views work.
    a. Let me break down Django views with a simple real-world example. Think of a view like a restaurant waiter - when a customer (user) makes a request, the waiter (view) takes that request, goes to the kitchen (database/backend logic) if needed, and brings back whatever was requested. So if someone visits your website's homepage, the view handles that request and decides what to show them. It could show them a list of products, a login page, or whatever makes sense for that particular URL.

2. Imagine you’re working on a Django web development project, and you anticipate that you’ll have to reuse lots of code in various parts of the project. In this scenario, will you use Django function-based views or class-based views, and why?
    a. For a project with lots of reusable code, class-based views are definitely the way to go. Here's why: imagine you're writing a cookbook. With function-based views, it's like writing out the full recipe every single time you want to use it. But with class-based views, it's more like having a basic recipe template that you can easily modify for different dishes. You write the common steps once in a parent class, then any specific view can inherit those steps and just add its own unique ingredients or instructions. This saves tons of time and keeps your code much more organized.

3. Read Django’s documentation on the Django template language and make some notes on its basics.

a. The Django template language is kind of like a smart word processor for web pages. Instead of just static text, you can do things like:
   - Insert dynamic content (like showing a user's name)
   - Create loops (like listing out all items in a shopping cart)
   - Add conditions (like showing different content for logged-in vs logged-out users)
   - Have reusable pieces (like having the same header on every page)
   - Extend base templates (like having a basic page layout that other pages can build on)

   It's powerful because it lets you build dynamic web pages without writing everything from scratch each time. The syntax is straightforward and focused on making it easy to mix HTML with dynamic content from your Python code.

Exercise 2.5: Django MVT Revisited

Learning Goals

- Add images to the model and display them on the frontend of your application
- Create complex views with access to the model
- Display records with views and templates

Reflection Questions

1. In your own words, explain Django static files and how Django handles them.
   a. As I've learned, static files in Django are a way to handle things like CSS, images, and JavaScript. At first it seemed confusing, but it's actually pretty straightforward - you put these files in a 'static' folder in your app, and Django takes care of serving them. I just had to remember to use {% load static %} at the top of my template and then {% static 'path/to/file' %} whenever I needed to link to a static file.
2. Look up the following two Django packages on Django's official documentation and/or other trusted sources. Write a brief description of each.

Package	Description
ListView	You give ListView a model (like Recipe), and it automatically fetches all instances and sends them to your template as object_list. I just had to make my template loop through object_list to display all my recipes in a grid

DetailView	DetailView handles showing a single item from your database. When someone clicks a recipe, it uses the ID from the URL (like /recipes/1/) to fetch that specific recipe and sends it to your template as object. Then you can display all the details about that one recipe.

3. You're now more than halfway through Achievement 2! Take a moment to reflect on your learning in the course so far. How is it going? What's something you're proud of so far? Is there something you're struggling with? What do you need more practice with? You can use these notes to guide your next mentor call.
    a. Compared to a node.js backend, Django is a total breeze. The structure and interlinking of all the views, models, and URLs is much more straightforward and understandable. I am working on my own React Native project and I am considering completely rebuilding the backend in Django/Python it's that much better. As always, I don't feel like I fully understand the logic/syntax, so that will take some time.

Exercise 2.6: User Authentication in Django

Learning Goals

- Create authentication for your web application
- Use GET and POST methods
- Password protect your web application's views

Reflection Questions

1. In your own words, write down the importance of incorporating authentication into an application. You can take an example application to explain your answer.
    a. Authentication is super important because it helps protect user content - like in my recipe app, I want to make sure only logged-in users can add and edit their recipes. It also helps track who added what content, which is especially useful when you have multiple users contributing stuff like recipes or comments. And most importantly, it prevents random people from messing with other users' content, which keeps everything secure and organized.

2. In your own words, explain the steps you should take to create a login for your Django web application.
    a. For creating a login in Django, first I had to create view functions for login and logout in views.py - this handles the actual authentication process. Then I needed to make login

templates that show the form where users input their username and password. Finally, I had to set up the URLs in urls.py to connect everything together so users can actually access the login page and submit their credentials.

3. Look up the following three Django functions on Django's official documentation and/or other trusted sources and write a brief description of each.

Function	Description
authenticate()	Checks if the username and password match what's in the database, used when users login
redirect()	Used to send the user to another page after a certain action, in this case after login/logout, I redirect to main/login respectively.
include()	It allows us to import/use URLs from one URL file to another.

Exercise 2.7: Data Analysis and Visualization in Django

Learning Goals

- Work on elements of two-way communication like creating forms and buttons
- Implement search and visualization (reports/charts) features
- Use QuerySet API, DataFrames (with pandas), and plotting libraries (with matplotlib)

Reflection Questions

1. Consider your favorite website/application (you can also take CareerFoundry). Think about the various data that your favorite website/application collects. Write down how analyzing the collected data could help the website/application.
    - NYT Games data collection must include:
        - Time spent on puzzle (for leader board and sharing)
        - If the puzzle has been attempted/completed
        - Answers submitted for each puzzle
        - Login state

        - A check if the user is a subscriber
        - Account setting
    - Tracking puzzle completion data could be helpful in a myriad of different way. Many of the puzzles have different difficulties, so tracking the time spent/attempts can help the puzzle designers fine tune either the puzzles or the difficulty system itself. It is what also powers the leader board, making it possible to compete other with other players, which boosts user engagement.

2. Read the Django official documentation on QuerySet API. Note down the different ways in which you can evaluate a QuerySet.
    a. Iteration: this is essentially a loop does an action for each item within the QuerySet
    b. Slicing: This essentially pulls only pieces of results from the QuerySet based on a pattern. For instance, you can pull the first 10 results, or every 3rd result, etc.
    c. repr: It returns a string of a specific representation of the data set that can be defined. Typically a string of entry headlines.
    len(): returns the count of items within the query set.
    d. list(): this is used to store the query set as a list object.
    bool(): checks if something is true or false about the query set, i.e. does it contain ____

3. In the Exercise, you converted your QuerySet to DataFrame. Now do some research on the advantages and disadvantages of QuerySet and DataFrame, and explain the ways in which DataFrame is better for data processing.
    a. From what I understand, QuerySet is a great out of the box tool for basic data manipulation within Django, but its not built for completing complex and statistical data processing, both logically and performance-wise. DataFrame is designed for those more advanced methods of data handling, making it ideal for projects in which data is a driving force.

Exercise 2.8: Deploying a Django Project

Learning Goals

- Enhance user experience and look and feel of your web application using CSS and JS
- Deploy your Django web application on a web server
- Curate project deliverables for your portfolio

Reflection Questions

1. Explain how you can use CSS and JavaScript in your Django web application.
    a. I opted to use a static/shared CSS file as consistent styling across the whole app is really important and having inline css makes it really hard to maintain a consistent style. These are loaded in with load static which pulls in the code from the shared css file. For JS, I

opted to keep the scripts inline for a few reasons. The scripts themselves are not necessarily reused, so having a centralized file for the little JS I had did feel unnecessary. To keep it as is just speeds up the development.

2. In your own words, explain the steps you'd need to take to deploy your Django web application.
    a. Freeze dependency list to requirements.txt to ensure Heroku can install them once deployed.
    b. Set up a Procfile that tells Heroku how to run the app: web: gunicorn recipe_app.wsgi:application.
    c. Properly set settings.py file to include Heroku as an allowed Host.
    d. Push the app via the Heroku CLI
        i. heroku login > heroku create > git push heroku main > heroku run python manage.py migrate > heroku run python manage.py createsuperuser

3. (Optional) Connect with a few Django web developers through LinkedIn or any other network. Ask them for their tips on creating a portfolio to showcase Python programming and Django skills. Think about which tips could help you improve your portfolio.

4. You've now finished Achievement 2 and, with it, the whole course! Take a moment to reflect on your learning:
    a. What went well during this Achievement?
        i. I love Django! Learning and building this app was really understandable and fun to troubleshoot.
    b. What's something you're proud of?
        i. While its not perfect, I really like how I set up the UX/UI. It feels good to use the app.
    c. What was the most challenging aspect of this Achievement?
        i. Pushing to Heroku ended up being really difficult. I had it working completely locally but when I pushed to Heroku I would get 404 error. When I attempted to fix it, I would get a 500 error and have issues with HTTPS. I went around in circles trying to troubleshoot. I eventually just reverted to a commit that was working and went through the steps again. I'm still not 100% why it didn't work the first time.
    d. Did this Achievement meet your expectations? Did it give you the confidence to start working with your new Django skills?
        i. Yes! The skills I learned from this achievement will be immediately useful to me, as I am going to re-build the backend of the app I've been working on using Django.

Well done—you've now completed the Learning Journal for the whole course.