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?

My coding and programming experiences so far have mainly been in JavaScript. I've worked on web development projects where I've used JavaScript to create interactive and dynamic web applications. I have good understanding of web technologies like HTML and CSS, and I've also used front-end frameworks like React to build user interfaces. These experiences in web dev and programming-related tasks, such as debugging and problem solving, will be valuable as I transition to learning Python.

1. What do you know about Python already? What do you want to know?

While I have some familiarity with programming concepts from my experience with JavaScript, I am relatively new to Python. I know that Python is widely used programming language known for its simplicity and readability. I'm aware that Python is used in various domains including data analysis, machine learning, and more. In this course, I want gain solid foundation in Python, learn about its syntax, data structures, and core libraries, and explore how Python could be applied in different fields.

1. 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.

Some challenges might include grasping Python's unique syntax and concepts, understanding differences between Python and JavaScript. To overcome these challenges, I plan set aside dedicated time each day or week to practice Python, work through exercises, and build small projects. I'll also seek support from online resources, forums, and my mentors who can provide guidance and answer my questions.

### 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?

The difference between frontend and backend web development is that frontend is more focused on the user interface and user experience, dealing with how the webapp looks and interacts with users, while backend is concerned with server-side operations.If I were hired for backend programming work on a web app, I would be working on operations like creating and maintaining databases, handling user authentication and authorization, implementing server-side logic to process and store data, and ensuring the overall functionality and security of the application.

1. 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?

*(Hint: refer to the Exercise section “The Benefits of Developing with Python”)*

JavaScript and Python are somewhat alike, but they also have differences. Python is easier to read, which is important. Python's neat writing style helps separate code parts. Both let you use variables without saying what kind they are, which can be handy. Python has lots of ready-to-use extra stuff you can add. For example, it has math tools built-in, making math easier. Other bits help with things like passwords, dates, and working with data like JSON. Python already has helpful tools for web work, like handling forms, links, and security. This means you can get going quickly. Python is good for catching mistakes because it's easy to read. It also doesn't need you to say what type of thing a variable is. So, if you start with a number and later want it to be a word, Python is okay with that. To sum up, Python is a good choice for web work. It's easy to read, can handle different kinds of data, and has lots of helpers.

1. 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?
   * + 1. I want to deepen my understanding of Python's core concepts, such as data types, control structures, and functions.
       2. I aim to explore Python's capabilities in web development by creating web apps and understand how frameworks like Django and Flask work.
       3. After completing this Achievement, I envision myself working on projects that involve both frontend and backend development.

### 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?

So, when talking with a future colleague, I'd say that the iPython Shell is better than Python's default shell for a few reasons. Firstly, it's more interactive and user-friendly. You get features like auto-completion and syntax highlighting, which can help you write code more efficiently. Also, it allows you to access the history of your commands, making it easier to see what you've done. Oh, and you can run shell commands directly, which is quite handy.

1. 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?** |
| Integer (int) | It's for whole numbers, like 1 or -5. | It's scalar because it represents a single value. |
| Float | This one is for numbers with decimal points, such as 3.14. | Scalar because it's just one number. |
| String (str) | It's for text, like "hello" or 'Python'. | Non-scalar because it's a sequence of characters. |
| List | It's like an array in JavaScript. You can put multiple values in it. | Non-scalar because it's a collection of values. |

1. 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.

Lists and tuples are both used to store collections of items. However, the main difference is that lists are mutable (can be changed), while tuples are immutable (cannot be changed). So, if one needs a collection where you might need to add, remove, or modify items, use a list. If one wants a fixed collection that cannot be altered after creation, use a tuple.

1. 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? Think about their respective advantages and limitations, and where flexibility might be useful if you were to continue developing the language-learning app beyond vocabulary memorization.

For language-learning app with flashcards to keep words, meanings, and categories, a dictionary is good. It stores pairs of things, like word-meaning. Word is one thing, meaning and category are other things connected to it. Dictionaries are good because they're flexible, easy to get info from, and can grow with more stuff for a growing app.

### 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!)*

|  |
| --- |
| # Define 3 destinations  destinations = ["Hanoi", "Seoul", "Tokyo"]  # Ask the user where they want to travel  user\_input = input("Where do you want to travel? ")  # Check if the user input is one of the defined destinations  if user\_input in destinations:  print("Enjoy your stay in " + user\_input + "!")  else:  print("Sorry that destination is not currently available.") |

1. 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.

Logical operators in Python are used to perform logical operations on boolean values (True and False). There are three main logical operators:

**and**: This operator returns True if both operands are True. Otherwise, it returns False.

**or**: This operator returns True if at least one of the operands is True. Otherwise, it returns False.

**not**: This operator does the opposite of what is true. If something is true, "not" makes it false, and if something is false, "not" makes it true.

1. What are functions in Python? When and why are they useful?

Functions in Python are blocks of reusable code that perform a specific task. They are defined using the **def** keyword, followed by the function name, parameters (if any), a colon, and a block of code. They are useful for several reasons:

* They help you split your big task into smaller, easier-to-handle pieces. Each function does its own job, so your program isn't a mess.
* Once you make a function, you can use it over and over in different parts of your program. No need to write the same stuff again and again.
* They make your code easy to read and manage.

1. 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.

Goal 1: I've been digging into the fundamental concepts of Python, like data types, control structures, and functions. I'd say I've deepened my understanding of these basics.

Goal 2: As for exploring Python's web development capabilities with Django and Flask, I haven't made significant progress in this area just yet. However, I'm looking forward to delving into these topics as the course goes on.

Goal 3: Thinking about working on projects that involve both frontend and backend development, it's still a bit on the horizon for me.

### 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?

File storage in python is crucial for data preservation and retrieval. Without it you ‘d lose your information every time your program ends. File storage not only enables you to maintain data across different program sessions but also facilitates data sharing with other programs.

1. 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?

In python pickles are like special containers that can hold and release objects. They change objects into a format that's easy to save or send. You can use pickles when you want to keep complex objects or big pieces of information for later. They make it simple to store and get back objects, help different python programs talk to each other, and allow data to be shared between processes.

1. 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?

You can use the os.getcwd() function in python to figure out the name of the folder you're currently in. It gives you the folder's name in plain text. If you want to go to a different folder, you can use os.chdir(path), where path is the name of the folder you want to go to.

1. 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?

To prevent a script from crashing due to an error in a particular part, you can use a helpful feature in python called exception handling. It's like having a safety net. All you need to do is enclose the code you're concerned about in a try block and include one or more except blocks to handle any issues that might arise. So, if there's a problem your script won't crash, and instead it will continue running from the except block. This way you can deal with the error and keep the rest of your script running smoothly.

1. 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? What do you need more practice with? Feel free to use these notes to guide your next mentor call.

As someone new to python I've encountered my fair share of challenges. Googling and seeking chatGPT's assistance have become regular parts of my learning process, but I've come to realize that it's perfectly normal. Despite the struggles my optimism and curiosity drive me to continue learning. The prospect of building a real app keeps me motivated, and I'm genuinely excited to see where this course will lead me.

### 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?

OOP is a coding approach that uses objects and classes. Objects are like building blocks, each with unique properties and actions. It helps break complex problems into manageable parts and promotes code reusability. OOP also provides encapsulation, hiding object details for security. It supports abstraction, where you focus on what objects do. Inheritance allows new classes to inherit from existing ones, and polymorphism allows working with different objects similarly.

1. What are objects and classes in Python? Come up with a real-world example to illustrate how objects and classes work.

Objects in python are instances of classes. Classes define the blueprint for objects, specifying their attributes (variables) and methods (functions). Consider a "Molecule" class in chemistry/biochemistry. It has attributes like "chemical\_formula," "molecular\_weight," and "pH\_stability," and methods like "react" and "bind\_to\_substrate." An object of the "Molecule" class could represent glucose with attributes "C6H12O6," "180.16 g/mol," and "pH 7.0." When you call "glucose.react()" or "glucose.bind\_to\_substrate()," it simulates the respective biochemical reactions for glucose. In this biochemistry context, "Molecule" is the class, and "glucose" is an object of that class. Objects allow us to model and simulate complex biochemical processes, making it easier to study and understand the behavior of molecules like glucose in various biochemical reactions.

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

|  |  |
| --- | --- |
| **Method** | **Description** |
| Inheritance | In OOP inheritance lets you create a new class (child) based on an existing class (parent). The child class inherits all attributes and methods from the parent, adding its unique ones. Think of it as creating a class that's much like an existing one but with some differences. |
| Polymorphism | Polymorphism in OOP is a versatile concept that allows objects of different classes to be treated as objects of a common parent class. This enables you to use a single interface, such as a method or function, to work with various types of objects. |
| Operator Overloading | Operator overloading in OOP means we can make operators do special things with our own objects. For example, we can say what the "+" sign should do when we use it with our custom objects. So, if we have two classes for molecules in biochemistry, we can decide how they should combine when we use the "+" sign. It's like making our own rules for how math works with our objects. |

### 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?

Databases are like organized containers for storing, retrieving and managing data. They help us store data in a structured way, thus, making it easier to access and manipulate. Some advantages are: Data **Integrity**: they ensure data consistency and accuracy. **Security**: access control protects data from unauthorized users. **Scalability**: databases can handle large amounts of data.

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

|  |  |
| --- | --- |
| **Data type** | **Definition** |
| VARCHAR(n) | Stores variable length text with a max length of “n”. |
| INT | Stores whole numbers without decimal point. |
| FLOAT | Stores decimal numbers with a decimal point. |

1. In what situations would SQLite be a better choice than MySQL?

SQLite is a good choice for small-scale applications. It is serverless, which means it doesn't require a separate server, making it easier to set up. I would use SQLite when I need a light and self contained database for apps like simple desktop programs or simple mobile apps.

1. Think back to what you learned in the Immersion course. What do you think about the differences between JavaScript and Python as programming languages?

Comparing JavaScript and Python, Python's syntax is simpler and more readable making it beginner friendly. Python has a broader range of applications beyond web development, including data analysis, artificial intelligence, and scientific computing. JavaScript, on the other hand, is mainly used for web development.

1. 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?

So far, I have grasp on – OOP principles, including classes, objects, methods, and attributes.

Python Syntax: I started understanding its syntax, such as variable assignment, function definitions, and conditional statements. Working with Databases covered by this lesson. File handling. However, I am still not sure of its limitation as I still have not pushed its boundaries, hence, and I only can speculate based on my research on the internet – its speed, as it can be slower than other languages for heavy computational tasks. Memory usage - Python might use more memory, which can be an issue for big datasets. However, it still seems that Python is great for bioinformatics, data analysis, and scripting because it's easy to learn, readable, and has useful libraries like NumPy and SciPy.

### 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?

ORM is like a helper tool for talking to a database from a programming language. It is a bit like a translator that helps your program understand the database. It turns database tables into objects that your program can understand. This makes your code simpler and safer because it helps prevent problems with the database. To sum up, it makes talking to databases in Python much easier and safer.

1. 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?

The creation of the Recipe app went relatively smoothly. I hope I was able to design and implement the core functionalities, including adding, viewing, searching, editing, and deleting recipes. In my opinion, the app provides a user-friendly interface for managing recipes, and it calculates the difficulty level of recipes based on cooking time and the number of ingredients. One of the strengths of the app is its modular structure, which makes it easy to add new features or expand existing ones in the future.

1. 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.

I have experience creating a Recipe app using Python. In this project, I designed and implemented a recipe management system that allows users to add, view, search, edit, and delete recipes. Specifically, I used the SQLAlchemy library for high-level object-relational mapping and the mysql.connector library for direct database interaction. This allowed me to define models, manage database connections, and perform operations such as creating, reading, updating, and deleting records.

1. You’ve finished Achievement 1! Before moving on to Achievement 2, take a moment to reflect on your learning in the course so far:
   1. What went well during this Achievement?

I successfully grasped the Python code structure, which was facilitated by my background in JavaScript.

* 1. What’s something you’re proud of?

Programming, to me, is a different way of thinking, and I'm proud of the progress I've made. It has been very challenging…

* 1. What was the most challenging aspect of this Achievement?

The most challenging part has been implementing complex logic.

* 1. Did this Achievement meet your expectations? Did it give you the confidence to start working with your new Python skills?

Achievement 1 met my expectations by providing valuable experience and insights into Python programming. It has encouraged me to further develop my coding skills.

* 1. What’s something you want to keep in mind to help you do your best in Achievement 2?

To do my best in Achievement 2, I want to continue practicing Python, especially writing code independently without relying on templates or examples. Additionally, I'll explore more complex database operations and improve my problem-solving skills.

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](https://careerfoundry.com/en/steps/your-cf-team#receiving-support) 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?

If I must choose between using plain Python (vanilla) or a framework like Django for a project, there are some things to consider. For vanilla Python the advantage is it's simple and flexible, but it may take more time to build a complete web application. Additionally, vanilla Python might not follow best practices for web development. On the other hand, Django provides a lot of tools and pre-built components, which can save time and make development faster.

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

The most significant advantage of MVT over MVC is that it emphasizes a clear separation between the data (Model), how it's presented to the user (View), and the logic that connects them (Template). In MVT, the Template acts as a bridge between the Model and the View, making it easier to design and maintain user interfaces.

1. 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 in depths how to create a web application using Django, including setting up the project, defining models, creating views, and designing templates. I've already started building a social media app with Django, and I hope to enhance its complexity and functionality with the knowledge gained from this Achievement.

* What do you want to get out of this Achievement?

I aim to understand the concept of Django's URL routing and how to handle different URLs and requests.

* Where or what do you see yourself working on after you complete this Achievement?

I see myself working on small web development projects using Django, gaining hands-on experience, and gradually building more complex applications as I become more proficient. My long-term goal is to continue improving my coding skills and pursue a career in web development.

### 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.*)

My dream company would be NASA, so asked on an interview to convert https://www.nasa.gov to convert it into Django terms, here is how I would proceed:

Website Analysis: I will thoroughly examine their website to understand its structure and functionalities. Django Models: I will then create Django models to represent different content types, like articles, images, videos, events. Views and Templates: I will develop Django views and templates to render the website’s various pages and components, such as the homepage, articles, and multimedia. URL Routing: Using Django URL routing, I will map URLs to corresponding views, allowing users to navigate the site seamlessly. User Functionalities: I will implement user authentication and registration using Django built-in authentication system, enabling features like personalized profiles and comments. Admin Panel: I will set up a Django admin panel for content management, making it easy for NASA’s team to update and publish content. Database Configuration: I will configure the database and perform data migration to ensure data consistency. Testing: I will rigorously test the application to guarantee its functionality and responsiveness. Deployment: I will deploy the Django application to a web server. Documentation: I will create comprehensive documentation for future maintenance and provide training for the website administrators.

1. In your own words, describe the steps you would take to deploy a basic Django application locally on your system.

To deploy a simple Django app on my computer, I would first create a folder for the project, set up a virtual environment, install Django, and start a new project. Then, I will configure the project’s settings, create database tables, and run the server. Finally, I will develop my app features and test it on my local machine.

1. Do some research about the Django admin site and write down how you’d use it during your web application development.

The Django admin site is a tool for managing your web app’s data. I would use it to add, edit, or delete data, like user accounts or blog posts without writing code. It is like a control panel for your app’s content, making it easy to keep your website up-to-date and manage user information.

### 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 containers for different aspects of a website, similar to having a well-organized garage for your collection of cars. Imagine you're managing a database for an auto dealership, and you need to keep track of all the cars available for sale. You can use a model to define what information you want to store about each car, such as the make, model, year, color, and price. These models function like compartments in your garage where each car has its designated parking spot. They help keep the data organized and structured, making it easier to manage your car inventory efficiently. Just like you wouldn't park a truck in a space meant for a compact car, Django models ensure that each piece of data is stored in the right place. The benefits of using Django models in this context are significant. They allow you to create new car listings, retrieve information about specific vehicles, update prices, and remove sold cars in a structured manner. For instance, when you acquire a new car for the dealership, you don't need to worry about where to "park" it in your database or how to keep tabs on its details. The model handles all these tasks for you. This structured approach not only simplifies your data management but also helps prevent errors, ensuring consistency in the information you present to potential buyers.

1. 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.

Imagine you're building an online platform for buying and selling cars. One of the crucial features of your platform is that when someone lists a car for sale, it should appear in the search results for potential buyers. If you don't write test cases from the beginning, here's what might happen: You launch your platform, people start listing their cars, and everything seems fine initially. However, after a while, you receive complaints from users that their car listings are not showing up in the search results as expected. Without test cases, you might find yourself searching for the problem in a vast sea of code. Test cases act as a quality control mechanism, helping you identify and fix issues early in the development process. This way, you can maintain a smoothly functioning platform and provide a reliable service to your users, keeping them satisfied with their car-buying and selling experience.

### 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.

Django views are like traffic signs for your website. They guide web pages to show when you type a URL. For example, if you ask for a recipe, Django views tell the website to display the recipe page.

1. 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?

If I expect to recycle a bunch of code in different spots of my project, I would go for class-based views. They are like pre-made blueprints for web pages, making it easy to use and share code. Function-based views are simpler but not as flexible.

1. Read Django’s documentation on the [Django template language](https://docs.djangoproject.com/en/3.2/ref/templates/language/#templates) and make some notes on its basics.

Django's template language serves as a powerful tool for enhancing the presentation of your web pages. It allows you to seamlessly integrate dynamic content, such as names and numbers, into your templates by enclosing them within {{ }} tags. Additionally, it enables you to incorporate conditional logic and control structures using {% %} tags, which ensures that specific content is displayed based on predefined conditions. This template language offers a sophisticated mechanism for elevating the visual appeal and functionality of your web pages.

### Exercise 2.5: Django MVT Revisited

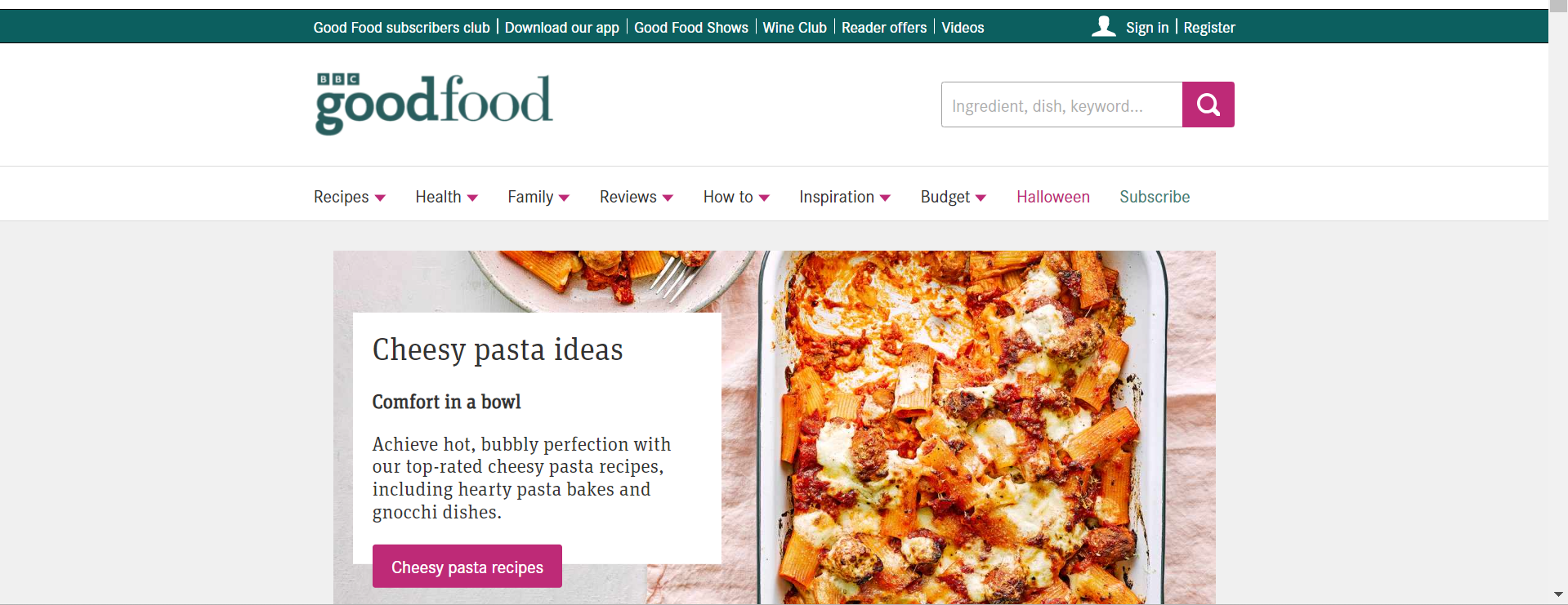
#### Learning Goals

* 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

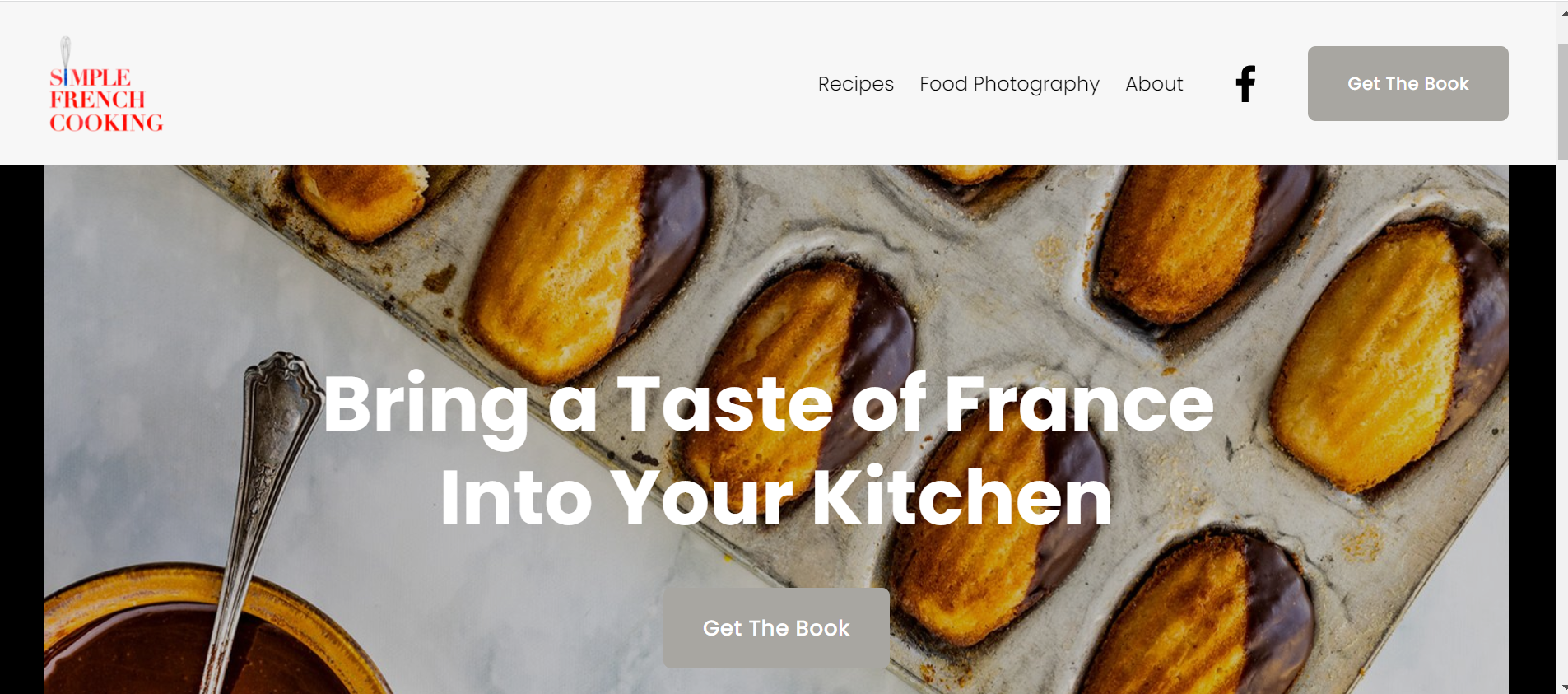
**Inspirations:**  
**1. BBC good food:**

Worldwide Cuisine: BBC Good Food stands out with its extensive collection of recipes from around the world. Users can explore and experiment with cuisines from different countries and cultures.Diverse Categories: The website offers a wide range of categories, catering to various preferences and dietary requirements. From health-conscious recipes to indulgent treats, it has something for everyone. Special Issues: The inclusion of special issues or thematic collections adds an extra layer of content. Users can find recipes and inspiration for specific occasions or events. Easy Navigation: The website's user-friendly interface makes it easy to navigate and locate specific recipes or articles.



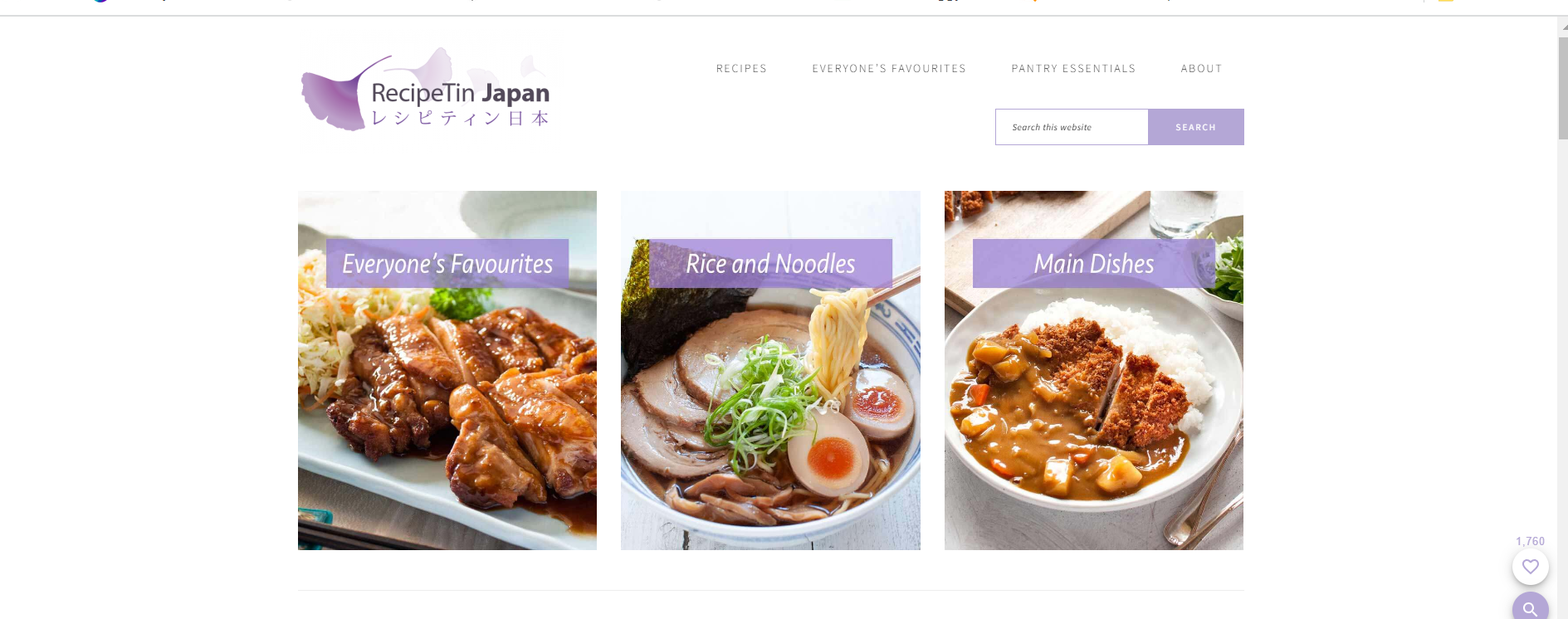
**2. Simple French Cooking:**

Easy French Recipes: The website stands out with its collection of easy French recipes. The simplicity of the recipes is a major plus, making French cooking accessible to a wide audience. Great Photography: The high-quality food photography used throughout the website is both enticing and informative. It allows users to see what the final dish should look like and encourages them to try the recipes. Simple and Easy to Navigate: The site's layout is straightforward and user-friendly. The recipes are neatly organized into categories, making it simple to find a specific dish or explore new ones. Variety of Recipes: The site offers a diverse range of French recipes, catering to different tastes and dietary preferences. Informative Content: Beyond recipes, the website provides informative content about French cuisine, including cooking tips, ingredient explanations, and regional specialties.



**3. RecipeTin Japan:**

Simplicity with Precision: Emulate the Japanese model's simplicity and precision in your Recipe application. Ensure that the design is clean and uncluttered, allowing the focus to remain on the recipes. Detailed Recipe Presentation: Provide detailed presentations of each recipe. Include information such as preparation time, cooking time, serving size, and nutritional facts. Step-by-Step Instructions: For each recipe, offer step-by-step instructions that are clear and easy to follow. Break down the cooking process into concise and well-organized steps. Video Demonstrations: Accompany recipes with video demonstrations. Users can benefit from visual guidance, enhancing their understanding of the cooking techniques.



1. In your own words, explain Django static files and how Django handles them.

Django static files are used for storing things like images, CSS, and JavaScript to make a website look and work correctly. Django handles them by collecting and serving these files when the website is run.

1. 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 | Django’s class-based view that helps display a list of objects from a database. It simplifies the process of retrieving and displaying data. |
| DetailView | Class-based view used to display details of a single object from a database. It is helpful when you need to show in-depth information about a specific item. |

1. 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.

I feel a sense of accomplishment in navigating through tasks, even when the instructions are a bit unclear (particularly this task). This experience is somehow valuable in terms of improving my adaptability and research skills, which I guess are essential for a full-stack developer.

### 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.

It ensures that only authorized users can access sensitive information or perform certain actions within the application. For example, consider a banking application. Without authentication, anyone could access your account details, leading to severe security breaches. Authentication guarantees that only users with valid credentials can view account balances, make transactions, and access financial data. It establishes trust, protects user privacy, and upholds the application's integrity.

1. In your own words, explain the steps you should take to create a login for your Django web application.

Creating a login system in Django involves several steps:

2.1. Login View: Develop a view that authenticates user credentials. Use authenticate() function to verify username and password.

2.2. Login Template: Design a template where users enter their credentials. Include form fields for username and password.

2.3. URL Mapping: Register the login view in Django's urls.py to map the URL. Define a URL pattern like login/ that triggers the login view.

2.4. Clickable Login Link: Integrate a login link or button in relevant pages. This link redirects users to the authentication form.

2.5. Protect Views: Use the @login\_required decorator to protect specific views. It ensures that only authenticated users can access those views.

2.6. Redirect After Login: Define the destination page where users should be redirected after a successful login. This could be the user's profile or a dashboard.

1. 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() | This function verifies the user's credentials. It takes a username and password as parameters, checks them against the stored data (usually in a database), and returns a user object if the credentials are valid. If not, it returns None. |
| redirect() | This is a shortcut function to perform HTTP redirects. It takes a URL as the parameter and redirects the user's browser to that URL. This is often used after form submissions or successful login, where users are redirected to a specific page. |
| include() | It allows you to include URL patterns from other files into the current urls.py file. This promotes modularity and keeps the URL routing organized, especially in large applications with multiple apps. |

### 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.

My favorite website is Reddit. It collects data on the topics I've shown interest in and the discussions I've engaged in. Analyzing this data can help Reddit recommend relevant content, subreddits, and discussions that align with my preferences, enhancing my overall experience on the platform.

1. Read the [Django official documentation on QuerySet API](https://docs.djangoproject.com/en/3.2/ref/models/querysets/). Note down the different ways in which you can evaluate a QuerySet.

The QuerySet API in Django offers methods like filter(), exclude(), get(), and annotate() to evaluate and filter the data in a QuerySet.

1. 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.

Converting a QuerySet into a DataFrame is useful because DataFrames offer more powerful tools for data manipulation and analysis. DataFrames are better for complex data tasks because they work well with libraries like pandas and matplotlib, making it easier to process and visualize data compared to QuerySets. However, QuerySets are simpler and more efficient for basic database operations in Django. DataFrames are better when you need to work with data extensively, especially for tasks like statistical analysis and chart generation.

### 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.

Using Static Files: You can create dedicated CSS and JavaScript files and place them in the 'static' directory of your Django application. Afterward, you can include these static files in your HTML templates by using the '{% load static %}' tag and the '{% static 'path/to/your.css' %}' template tag.

Incorporating Inline Styles and Scripts: One can directly insert CSS styles and JavaScript code into your HTML templates using <style> and <script> tags, respectively. While this approach is more suitable for smaller projects.

You can integrate third-party CSS and JavaScript libraries, such as Bootstrap or Tailwind CSS, into your Django project. This can be achieved by including the necessary library files and stylesheets in your HTML templates and configuring them for use in your web application.

1. In your own words, explain the steps you’d need to take to deploy your Django web application.

Deploying a Django web application typically involves the following steps:

1. Choose a Hosting Service: Select a web hosting service (e.g., Heroku, AWS, DigitalOcean, Render, Railways, PythonEverywhere) for your application. Ensure it supports Python and databases (e.g., PostgreSQL).

2. Version Control: Set up version control (e.g., Git) to track changes and collaborate with a team if needed.

3. Prepare the Environment: Create a virtual environment to isolate dependencies. Install Django and other required packages.

4. Database Setup: Configure your application's database settings. Set up a database (e.g., PostgreSQL, MySQL) and apply database migrations.

5. Static Files: Collect static files using the `collectstatic` management command to prepare them for serving.

6. Debug Mode: Set the DEBUG setting to False for production, and configure allowed hosts.

7. Domain and SSL (Optional): Configure a custom domain and set up SSL/TLS for secure connections.

8. Deploy: Deploy your application code to the server using Git or other deployment methods.

9. Web Server Configuration: Configure the web server (e.g., Gunicorn, uWSGI) to serve your Django application.

10. Proxy Server (Optional): Set up a reverse proxy server (e.g., Nginx or Apache) for load balancing and added security.

11. Automate Deployment (Optional): Use deployment scripts or CI/CD tools to automate deployment workflows.

12. Testing: Thoroughly test your deployed application to ensure it works as expected.

13. Monitoring and Maintenance: Implement logging, error handling, and monitoring tools. Schedule regular backups and updates.

1. (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.
2. You’ve now finished Achievement 2 and, with it, the whole course! Take a moment to reflect on your learning:
   1. What went well during this Achievement?
   2. What’s something you’re proud of?
   3. What was the most challenging aspect of this Achievement?
   4. Did this Achievement meet your expectations? Did it give you the confidence to start working with your new Django skills?

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