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?

   My only coding experience is from Careerfoudry's full-stack immersion course. Beein, an architect, helps in this course because I'm accustomed to doing and learning about new things. There is some parallel between newspapers and the web, so being DTP helps also; it is my current work.

2. What do you know about Python already? What do you want to know?
   I know nothing about Python, and I want to know everything …

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. Based on my previous work, I might have problems with the speed of the course and memorizing things. It will be over when it gets interesting.
I will solve the problem by isolating myself in a friend's beach house in Portugal and by working more.

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?

   Frontend web development involves creating the visual elements and user interface that users interact with on a website or application.

   On the other hand, backend web development involves working on the application's server side.

   If I were hired to work on backend programming for a web application, I would be working on various operations such as designing and developing APIs, writing code to interact with databases, implementing user authentication …

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?
   (Hint: refer to the Exercise section "The Benefits of Developing with Python")

Phyton has a simple and readable syntax, a large and active community, cross-platform compatibility, and versatility, allowing it to be used for various tasks. It promotes productivity and efficiency and has rich libraries and frameworks.

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?

    - I want to learn basic syntax of all necessary packages python backend developers use.
    - I want to have enough python knowledge to get comfortable with senior python developers.
    - In the future I want to be able to use python together with other programming languages to create interactive web apps.

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?

    I would give these reasons to explain why:

    - Enhanced interactive experience
    - Better debugging
    - Rich output
    - Easy data exploration
    - Access to advanced libraries

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?
Integer (int)	a whole number	scalar
Float (float)	a real number with a decimal point	non-scalar
String (str)	string is a sequence of characters	non-scalar
Boolean (bool)	has only two possible values: True or False	scalar

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.

   Once a list is created, you can add, remove, or modify elements in it, while a tuple is fixed and cannot be changed.

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? 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 this app I would use 'Dictionary' to store the information of each vocabulary in cards. Dictionaries have the advantage of storing date in key: value pairs which makes it easy to index and modify a particular attribute using its key name assigned. Also, dictionaries allow much more operations to be performed on the stored data enhancing flexibility that lists and tuples don't have.

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."

```
user_preference = input(
    "Do you want to travel? Enter 'Y' for yes and 'N' for no:   ")

if user_preference.upper() == 'Y':
    destination = input("Where do you wish to travel?   ")

    if destination.capitalize() == 'Dubai':
        print('Enjoy your stay in Dubai!')
    elif destination.capitalize() == 'London':
        print('Enjoy your stay in London!')
    elif destination.capitalize() == 'New York':
        print('Enjoy your stay in New York!')
    else:
        print('Ooops, that destination is not curently avaliable.')

else:
    print('Thanks for checking!')
```

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

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.

    Python has three logical operators: "and", "or", and "not". "and" returns True only if both operands evaluate to True, "or" returns True if at least one operand evaluates to True, and "not" is used to negate a boolean value.

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

   Functions in Python are reusable blocks of code that perform specific tasks. They are useful for breaking down complex tasks into smaller, more manageable parts, reducing code duplication and increasing code efficiency, and enabling code modularity,

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.

   I am happy to share that I have acquired a good understanding of the fundamental syntax of Python.

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?

   If you didn't store local files, you would not be able to store data permanently for later use, and you would have to rely on the program's internal state to keep track of any information. Moreover, if you don't store local files, you may lose important data, configurations, or settings of your program or application when the program is closed or restarted.

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?

   Pickling is a way to serialize and deserialize Python objects into a binary format. It's useful when you want to store or transfer complex Python objects. Pickle can be used for caching data, storing configuration data, and inter-process communication.

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?

   To find out the current working directory, you can use the os.getcwd() function. To change the current working directory, you can use the os.chdir() function.

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?

   I would wrap the block of code in a try block, and handle the exception in an except block. After the except block, the script can continue executing normally.

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

   I am absolutely blown away by the incredible readability of Python! It's truly amazing how easy it is to understand and follow the code. Although I may struggle at times to anticipate what's coming next, I'm so excited to continue learning and growing my skills.

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?

   Object-oriented programming (OOP) is a programming paradigm that uses objects and classes to represent and manipulate data and behavior. OOP provides benefits such as encapsulation, abstraction, inheritance, and polymorphism, which result in modular, reusable, and maintainable code. OOP promotes better code organization, structure, and collaboration among developers, leading to faster development, fewer bugs, and increased flexibility in code design.

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

   In Python, objects are instances of a class, while classes are blueprints for creating objects. A class defines the properties and methods that objects of that class will have.

   ```
   class Car:
       def __init__(self, make, model, year, color):
           self.make = make
           self.model = model
           self.year = year
           self.color = color
   ```

```
    def start(self):
        print("The car has started.")

    def stop(self):
        print("The car has stopped.")

    def accelerate(self):
        print("The car is accelerating.")

my_car = Car("Tesla", "Model S", 2022, "black")
print(my_car.make) # Output: Tesla
print(my_car.model) # Output: Model S
print(my_car.year) # Output: 2022
print(my_car.color) # Output: black
my_car.start() # Output: The car has started.
my_car.accelerate() # Output: The car is accelerating.
my_car.stop() # Output: The car has stopped.
```

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 is a mechanism by which a new class can be created based on an existing class. The new class inherits the properties and methods of the existing class, and can also add its own properties and methods or override existing ones.
Polymorphism	Polymorphism is the ability of an object to take on multiple forms. In object-oriented programming, this means that a single method can be used to perform different actions depending on the context in which it is called.
Operator Overloading	Operator overloading is a feature that allows operators in object-oriented programming to have different meanings or behaviors depending on the context in which they are used. This is achieved by defining special methods with specific names that correspond to the operators, such as add() for the + operator. When you use an operator on objects of a particular class, Python will automatically call the corresponding method to perform the operation.

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?
   MySQL databases are a type of database that are widely used for storing, managing, and retrieving data in various applications and systems. Some advantages of using MySQL databases include efficient data organization, secure data storage, scalability, easy data backup and recovery, and data integrity.

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

Data type	Definition
INTEGER	The INTEGER data type is used to store whole numbers (positive, negative, or zero) that do not contain decimal points. The size of the INTEGER data type can be specified in terms of the maximum number of digits it can store.
VARCHAR	The VARCHAR data type is used to store variable-length strings of characters. The maximum length of the VARCHAR data type can be specified when it is created.
DATE	The DATE data type is used to store dates in the format "YYYY-MM-DD".

3. In what situations would SQLite be a better choice than MySQL?
   SQLite is a lightweight, file-based database that can be embedded directly into an application, making it a good choice for small, single-user applications, as well as for prototyping and testing. It is widely used in mobile and desktop applications.

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?
5. When it comes to comparing Python to other programming languages, the first thing that stands out is its simplicity and readability, which makes it relatively easy to learn. In addition, Python comes with a wide range of built-in methods and functions, eliminating the need for additional packages. However, the main difference between Python and JavaScript lies in their areas of focus - Python is mainly used for backend/server-side programming, while JS is used for frontend programming and user interaction, with full support for DOM manipulation. Both languages have large and supportive communities.

6. 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?

I believe that one limitation of Python is its slow execution due to its line-by-line interpretation. This can be a significant drawback in sectors where efficiency is critical. Additionally, I find that accessing databases through Python is not straightforward and not user-friendly. Despite Python's versatility and simplicity, it is not commonly used for developing mobile and web applications.

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?
   ORMs map objects to tables in a relational database, providing an abstraction layer between an application and the database. They reduce development time, improve code maintainability, and increase security. ORMs allow developers to work with objects instead of raw SQL, and automate repetitive tasks in database interaction. They also provide a consistent interface for database interaction that improves database portability.

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?
   I wouldn't change a thing about the app as a whole.

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.
   I recently developed a cool terminal app that utilizes Python scripts to edit a SQL database via sqlalchemy ORM. The app starts with a table, and users can then choose to add recipes, display the recipe list from the database, edit recipe attributes, and delete recipes from the database.

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?
      It's great that the instructions were easy to follow, especially since I have no prior experience with Python.
   b. What's something you're proud of?
      I feel a sense of accomplishment having made it this far with CF courses!

c. Did this Achievement meet your expectations? Did it give you the confidence to start working with your new Python skills?
   Due to my lack of prior experience, I don't feel very confident about using Python.
d. What's something you want to keep in mind to help you do your best in Achievement 2?
   It's important for me to remember how far I've come and to keep pushing through.

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?
   Vanilla Python offers more control and flexibility but also requires extensive knowledge of web development best practices. Django offers faster development time and built-in functionality but has a steeper learning curve and less control over the application's code and design.

2. In your own words, what is the most significant advantage of Model View Template (MVT) architecture over Model View Controller (MVC) architecture?
   The MVC architecture requires a controller to coordinate between the model (which interacts with the database) and the view (which prepares the presentation of the page), sending HTTPS responses for every request received. In contrast, the MVT architecture places the view in the middle of the model and the template, coordinating them and sending back a response URL. The advantage of MVT is that the view matches the URL with the specific request and returns a response, eliminating the need to write code for the view part, as required in the controller part of MVC.

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?
   - What do you want to get out of this Achievement?
   - Where or what do you see yourself working on after you complete this Achievement?

     Upon completing this achievement, I aim to have a strong command over various features and tools of Django. I believe that if I can successfully develop applications capable of handling high traffic requests or handling large amounts of data with ease while being able to scale up or down with minimal effort, it would be a significant accomplishment for me.

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.*)

   To convert a company's website into Django terms, you need to analyze the website's structure, identify the models, define URL patterns, create views and templates, integrate CSS and JavaScript, and test the website thoroughly.

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

   To deploy a Django application, I would first install Django and then use its built-in tools to start a new project, which will create the file structure. After that, I would run migrations to create the database, which should be configured in the settings.py file. Then, I would create a superuser for the admin and update the database. Next, I would add apps that perform specific functions within the project. Finally, I would run the server on the localhost to test the application.

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

   The django admin panel is an interface where users can create, read, update and delete on the model directly. It is an instant interface where user can manage contents of the site using CRUD operations, which saves a lot of development time.

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 a core part of the Django web framework, used for defining the structure and behavior of data stored in a database. They simplify database management, provide data

validation, abstract database details, support portability, and seamlessly integrate with other Django components.

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 beginning ensures early bug detection, maintains functionality, facilitates refactoring, enhances collaboration, enables CI/CD, and saves time.

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 handle HTTP requests and generate responses. For example, in a blog application, a function-based view named `blog_list` can be created. This function takes a `request` object as a parameter, performs data retrieval, and stores the posts in a `posts` variable. Then, using the `render` function, the `blog/list.html` template is combined with the `context` dictionary containing the `posts`. This generates an HTML response. The view is associated with a URL pattern in the project's `urls.py` file. When a user visits the corresponding URL, Django invokes the `blog_list` view, retrieves the blog posts, and renders them using the specified template. The resulting HTML response is sent back to the user's browser, displaying the list of blog posts. Django views are essential for handling requests, performing logic, and generating responses, enabling the creation of dynamic web applications.

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?
   In this scenario, class-based views are preferable for code reuse in a Django web development project due to their support for inheritance, mixins, and simplified method handling. They provide flexibility, modularity, and built-in functionality that promote cleaner and more reusable code.

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

Django's template language is used for rendering dynamic content. It supports variables, filters, and tags. Variables are enclosed in double curly braces, filters modify variable values, and tags provide control flow. Template inheritance allows creating base templates, and loops iterate over collections. Object attributes are accessed using dot notation, and custom tags and filters can be created.

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.

   Django static files are files served directly to the client without any processing by the Django application. They include CSS stylesheets, JavaScript files, images, and other assets. Django provides a mechanism to handle them efficiently. By separating static files from dynamic content, Django ensures quick and reliable delivery, while features like automatic versioning and compression optimize their performance. Overall, Django simplifies the management and serving of static files, improving the efficiency and maintainability of web applications.

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	The ListView is a class-based view in Django that simplifies displaying lists of objects. By inheriting from it and specifying the model, template, and other attributes, you can fetch objects from the database and render them in a template. ListView automatically handles tasks like querying the database, pagination, and rendering.

DetailView	DetailView simplifies displaying details of a single object. It fetches the object from the database and renders it using a template.

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.

   During Achievement 2, my learning has been going well. I have gained a solid understanding of Django and its various components. I am proud of my ability to explain concepts like models, views, and template rendering in Django. I have also grasped the usage of class-based views and the handling of static files.

   While I feel confident in my understanding of Django, I would like to further improve my skills in database integration and working with more complex queries. Additionally, I would appreciate more practice in handling authentication and user management within Django applications.

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.

   Incorporating authentication is crucial for user identification, security, and personalization. Authentication ensures that only registered users can access the app, safeguarding their personal information and preventing unauthorized access. It enables personalized features like saving favorite recipes.

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

First you need to create a view for your login functionality of your app. The view should first check that when the user hits the 'login' button, a 'POST' request with required data is sent using AuthenticatioinForm(), a django built-in function, then it should check if the user is valid - during which it sends an error message if it's not – using another django built-in function, authenticate() . Next a successful user is logged in using yet another django built-in function login() and taken/redirected to the requested page.
In the next step, you create a template for the login form and register the URL for the new login view you created. Finally you protect the views you want only authenticated users to access by passing LoginRequiredMixin as parameter in class based views and on function based views add @login_required decorator before the definition.

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()	is a django function which takes user credentials as parameter and checks if user is valid against the backend data. If user is valid it returns a user object if user is not valid it returns None.
redirect()	is a function from django module djnago.shortcuts that takes URL of the page you want a user to be directed to and it returns the view of that page which in turn the view displays the corresponding template.
include()	is django function used add urls from apps directory to the main urls.py file in the project directory.

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.

2. Read the Django official documentation on QuerySet API. Note down the different ways in which you can evaluate a QuerySet.

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.

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.

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

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?
    b. What's something you're proud of?
    c. What was the most challenging aspect of this Achievement?
    d. 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.