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 recent experience with programming so far is in the Career Foundry bootcamp for JavaScript. I believe that my previous certification in web design from Design Labs might be beneficial for me to work through this course when it comes down to designing.**
2. What do you know about Python already? What do you want to know?
    a. **I honestly don't know too much about Python other than a lot of people say it's easier to understand once you've understood another language. Based on some opinions from my colleagues, they claim it's easier to comprehend compared to JavaScript. What I would like to know is how much of the world is running Python compared to JavaScript.**
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. **I think a big challenge for me is that since I'm using Python 3.12 and the achievement is using 3.8. I'm concerned about things that might be obsolete or may have changed. I believe a good way to combat this challenge is to probably read up on the documentation/release notes to see if there are any changes, and using Stack Overflow to see if there are any workarounds if something might not work. Similar to how I was working in the JavaScript course; I ran into an issue where certain operations in Angular 10 had changed in Angular 18, and I had to use Stack Overflow as a resource.**

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. **Front-end development focuses on the visible aspects of a site, including design and interactivity, using technologies like HTML, CSS, and JavaScript. Conversely, backend development handles the server-side processes, managing databases, servers, and application logic. Backend programmers engage in tasks such as database management, server-side coding, API development, security implementation, performance optimization, external service integration, and error handling.**

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")
    a. **Python stands out over JavaScript for web development due to its readability, dynamic typing, extensive package ecosystem, built-in essentials, and strong community support. With Python, developers find code easier to understand, variables more flexible, and access to a wealth of pre-built solutions for various tasks like mathematical calculations and database connections. Additionally, Python frameworks streamline common web operations, reducing development time, while its supportive community offers resources for troubleshooting and continuous improvement.**

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. **I want to build some real world projects with Python and compare it to JavaScript to see what might be my preferred language.**
    b. **I want to gain a solid understanding of the fundamentals concepts of Python so that I can write clean, efficient and maintainable code.**
    c. **I also want to explore possible specialization in the language itself.**

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. **IPython improves code readability with syntax highlighting, allowing for easier distinction between keywords and lines of code. It also automatically manages indentation for nested statements, which helps keep the code well-organized. In general, IPython offers a more user-friendly environment compared to Python's default shell.**

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?
Dictionary	**An unordered set of items that stores values and objects within itself indexed by identifiers, or keys.**	**Non-Scalar**
Tuple	**Linear arrays that can store multiple values of any type.**	**Non-Scalar**
List	**Similar to Tuple, but they are mutable; internal elements of a list can be modified or deleted.**	**Non-Scalar**
Bool	**Data type that represents one of two 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.
    a. **Tuple and lists both store multiple values, however tuples are immutable while lists are mutable. This means that when tuples are created, they cannot be changed, whereas lists can still be modified by adding, removing, or updating after creating them.**
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.
    a. **For the language-learning app, dictionaries would be the best choice. They allow organizing vocabulary words with their definitions and categories efficiently, making it easy to access information when flipping through flashcards during quizzes. Dictionaries also offer flexibility for future enhancements like adding example sentences or user notes without needing to reorganize existing data. This adaptability**

and efficient data retrieval make dictionaries the optimal data structure for both current needs and potential app expansions.

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!)*

```
travel_destinations = ["california", "utah", "washington"]
state = input("Enter where you would like to travel: ")
if state.lower() in travel_destinations:
   print("Enjoy your stay in", state + "!")
else:
   print("Oops, that destination is not currently available.")
```

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.

    Logical operators are used when evaluating conditional statements. The `and` operator returns `True` if both operands are `True`. The `or` operator returns `True` if at least one operand is `True`. The `not` operator reverses the boolean value of its operand.

3. What are functions in Python? When and why are they useful?
    Functions in Python are defined using def, followed by a name and optional parameters in

parentheses. They encapsulate specific tasks, promoting code organization and reusability. They help facilitate by dividing complex tasks into manageable parts.

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. **Currently still working on understanding Python more before building a real-world app.**
    b. **Currently in progress with this course by doing the assignments and reading.**
    c. **Again, a work in progress from above. Also looking at examples of others that made Python apps.**

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. **It's crucial because without file storage, any data assigned to variables during script execution would vanish once the script concludes, rendering it inaccessible for future use. Employing file handling methods in Python allows this data to be saved into files, ensuring it remains available beyond the script's runtime.**

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 serialized byte streams used to store data. Through the pickle.dump() method, data is serialized into these pickles and written into binary files. They are particularly useful for handling complex data structures like dictionaries, allowing their storage in files. Pickles facilitate the retrieval and access of this data while maintaining its original structure.**

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. **I would utilize functions provided by the os module. To determine the current directory, os.getcwd() can be employed, while os.chdir() can be utilized to switch to a different directory.**

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. **I would attempt using try-except blocks. Within the try block, I'd place code that might encounter an error, while in the except block, I'd handle what to do if an error occurs. If no error arises in the try block, the remaining code continues execution normally without triggering the except block. However, if an error occurs, the code within the except block executes, preventing the entire script from terminating.**

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.
    a. **The course is going smoothly so far. I feel like I understand the functions a lot more compared to Javascript, but still struggle with syntax a bit. However, I believe a bit more practice on my end when it comes to actually programming will make it more clear to myself.**

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. **Object-oriented programming organizes code into objects, each with its own attributes and methods. The benefits of OOP include making code reusable and easier to maintain and debug.**
2. What are objects and classes in Python? Come up with a real-world example to illustrate how objects and classes work.
    a. **In Python, a class is a blueprint for creating objects. It defines the attributes and methods of objects. A real-world example would be a car, where a "car" class can be defined with attributes such as the `make`, `model`, `year`, and `color`, and methods would be `start`, `stop`, `accelerate`, and `brake`. The objects would be specifying if it is my car, or someone else's car.**

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

Method	Description
Inheritance	**A feature that enables a new class, known as a subclass or inherited class, to inherit attributes and methods from an existing class, referred to as a parent class or base class.**

	This allows the subclass to reuse code from the parent class and expand upon its functionality. By inheriting all attributes and methods of the parent class, the subclass can also introduce its own additional attributes and methods. This concept encourages the reuse of code and facilitates the extension of software capabilities.
Polymorphism	**Refers to the ability of methods or attributes to share the same name across different classes while performing different actions based on where they are implemented. Because these definitions are independent and distinct, there is no conflict between them.**
Operator Overloading	**Allows you to redefine how operators such as +, -, and others behave for custom classes. This means you can customize these operators to work with user-defined objects, in addition to their default functionality.**

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 structured repositories of information. They provide benefits such as maintaining data consistency through standardized storage formats, which streamline the process of storing and retrieving data. Additionally, databases include security features like password protection to ensure data safety. Moreover, databases enable data accessibility across different applications, including Python.**

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

Data type	Definition
VARCHAR(n)	**VARCHAR is a data type for variable-length character strings. It can store a flexible number of characters up to a maximum length specified by (n).**
INT	**INT is a data type used for storing integer values, which are whole**

	numbers without decimal points.
DATETIME	**DATETIME is a data type used to store values that represent both date and time information.**

3. In what situations would SQLite be a better choice than MySQL?
    a. **SQLite is preferable over MySQL when you require a lightweight and portable database solution. This is particularly useful for managing uncomplicated databases or for rapid testing needs. SQLite offers straightforward setup and portability by storing data in simple .db files, which enhances convenience compared to MySQL.**

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. **JavaScript is predominantly employed in web development, emphasizing interactive features and dynamic content. It is frequently paired with HTML and CSS to build web applications. In contrast, Python is renowned for its readability and versatility. Python finds extensive use across multiple domains, including web development, data analysis, and machine learning. While JavaScript is typically linked with frontend development, Python is often favored for backend development, scientific computing, and other applications.**

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. **To my knowledge, Python is generally not utilized for frontend development. This implies that Python might have certain limitations in that domain when compared to languages more commonly employed for frontend tasks.**

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. **An object-relational mapper (ORM) facilitates the conversion of database contents and structures into classes and objects, enabling easier interaction. It simplifies database operations, speeds up development, and improves code readability and maintainability.**

Developers can work with familiar programming concepts such as classes and objects, rather than directly managing SQL queries and database tables.

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. **I think overall it went pretty smoothly. If I were to start over, I would focus more on error handling and maybe touching up on writing more optimal code.**

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. **I recently developed a Recipe app in Python, which included building a user-friendly menu interface for adding, viewing, searching, updating, and deleting recipes. This project allowed me to practice object-oriented programming concepts and work with SQLAlchemy for interacting with a relational database. I also gained experience in implementing error handling and managing database operations efficiently.**

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. **I felt like everything I was reading was applied when writing the app.**
    b. What's something you're proud of?
        i. **I was proud that I could write most of the code off the top of my head after reading and digesting it, then be able to confirm it was written correctly when looking back at the examples.**
    c. What was the most challenging aspect of this Achievement?
        i. **Most challenging is trying to solve the traceback errors when it could be rooted back to indentation.**
    d. Did this Achievement meet your expectations? Did it give you the confidence to start working with your new Python skills?
        i. **I feel like this achievement did meet my expectations and I feel a lot more comfortable with Python.**
    e. What's something you want to keep in mind to help you do your best in Achievement 2?
        i. **I need to be more mindful of indentation because that could make or break a code, which could be at its specific section of the code, or at the forefront.**

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?
    - **I believe my study routine was effective.**
- 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?
    - **I felt the most proud when I was able to write code from memory after practicing it a few times. I hope to do the same thing with Achievement 2 by repeating and taking my time to understand what I'm reading/writing.**
- 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?
    - **I found that sometimes digesting the information can be longer than anticipated, especially when the information could be dense to read and not as exciting. How I plan to combat this is possibly using the Pomodoro method of studying to take breaks after reading something and going back to it.**

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 is straightforward and lightweight, allowing full control over code and organization. However, it involves more manual effort for web tasks such as client request handling and database interactions. Django accelerates development with pre-built components

for these tasks, including features for database access and file management. Yet, its structured approach must be adhered to, potentially adding complexity to smaller applications that don't require Django's built-in capabilities.

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

   In MVC architecture, developers need to write code for fetching data, displaying it, and mapping it to URLs. In contrast, MVT architecture streamlines this process by allowing developers to specify the presentation logic, leaving the framework to manage other aspects automatically.

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 more of what features are available in Django and how helpful the documentation will be towards building my app.
   - What do you want to get out of this Achievement?
     Practical use of Django and how it synergizes with Python.
   - Where or what do you see yourself working on after you complete this Achievement?
     I want to try and see if I can make a mock social media app for further my understanding of Python and Django.

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.*)

Twitch's setup in Django terms would involve the entire platform as a Django project. The homepage and featured streams translate into a Django app for dynamic content. Streams themselves are managed by another app with models for live content and views for displaying streams. User profiles utilize a Django app for user data and editing. Real-time chat operates through a Django app with WebSocket support. Recorded streams (VODs) are handled by a Django app for video storage and playback. Search functionality is facilitated by a Django app for managing search results and filters, while notifications are managed by a Django app for real-time user alerts.

2. In your own words, describe the steps you would take to deploy a basic Django application locally on your system.
   To set up a basic Django application on my computer, I'd begin by executing migrations to generate essential database tables and configure the backend (using `python manage.py migrate`). After that, I'd launch the server to deploy the application locally, enabling access through a web browser (using `python manage.py runserver`).

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 site is an integrated tool enabling developers to manage a web application's data models using an intuitive interface. Throughout web development, I would utilize the Django admin site to execute CRUD (Create, Read, Update, Delete) operations on the database efficiently.

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.

   In Django models, data structure is defined using Python classes, which are converted into database tables. This approach simplifies database interaction by making it easier to query and manipulate data.

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.

   Testing early allows you to identify and address bugs before they become major issues. For instance, in a web application, testing user authentication from the start can reveal security vulnerabilities early, reducing the risk of breaches or login problems. Additionally, it saves time by eliminating the need for manual testing every time changes are made.

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 incoming requests from users' browsers and send back web responses. For instance, when a user wants to access the "about us" page of a company's website, the corresponding view generates an HTML template with information about the company. The view then returns this response to the user's browser, displaying the "about us" page to the user.

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, I would use Django class-based views (CBVs). CBVs provide a more reusable and modular approach to building views, allowing for code reuse through inheritance and mixins. This makes it easier to maintain and extend the functionality across different parts of the project.

3. Read Django’s documentation on the Django template language and make some notes on its basics.
   - **Variables:** Enclosed in double curly braces `{{ variable }}`, they are replaced with their evaluated values. Variable names must follow specific naming rules, avoiding spaces, punctuation, and cannot start with an underscore.
   - **Filters:** Modify variables and are structured as `{{ value|modifying_func }}`.

- **Tags:** Can generate output, control flow, or load external information for use by variables, structured as `{% tag %}`. Some tags need both opening and closing tags, like `{% tag %} ... tag contents ... {% endtag %}`.
- **Comments:** Use `{# comment #}` to add comments in a template.
- **Template Inheritance:** Allows creating a base template with common elements and defining blocks that child templates can override.

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 not altered or dynamically created by the server during the web application's operation. They remain unchanged and include assets such as CSS, JavaScript files, or images.

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	**List View presents data from a database table in a defined sequence.**
DetailView	**Detail View in Django provides comprehensive information about a particular item from the database.**

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.

So far I'm very proud of my capabilities of url mapping/pathing. Although there are some struggles with not noticing typos, I tend to go back and look over the entire code instead of looking at my naming conventions. I believe I could cut down on time if I did simplistic debugging first before diving into the code. An example was that I could not get the "recipes_list.html" file to be located as I didn't notice that i had typed in "recipes_list..html". But, I was browsing throughout all my other files to see if I was calling on "recipes_list.html".

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.

   Authentication is crucial for guaranteeing that only authorized individuals can access an application's resources and functionalities. For example, in a banking application, authentication confirms the user's identity before allowing access to sensitive financial data, thereby safeguarding against unauthorized access.

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

   To implement a login feature in a Django web application, begin by creating a login view and template, and map the URL for access. Add a clickable login link or button on the homepage to direct users to the authentication form. Identify the pages that need protection through authentication and set up a redirect for users upon successful login. To secure views, incorporate `LoginRequiredMixin` or `login_required` into the relevant views to restrict access for unauthenticated users. Next, create a logout view and map its URL. Additionally, provide a logout button on the protected pages.

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 a user's credentials against the data in the authentication backend, returning a User object if the credentials are valid, or None if they are not.**
redirect()	**Accepts a URL as an argument and returns a response that redirects to that URL. This is used in views to redirect users to another page after handling a request.**
include()	**Imports and renders a template, allowing other templates to be included within a template.**

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.

   Analyzing the data Twitch collects offers multiple advantages for both the platform and its users. It enhances viewer engagement by identifying peak activity times and content preferences, enabling better content recommendations and viewing experiences. Monetization

efforts benefit from targeted ads, customized subscription offerings, and insights into donation behavior. Data informs the development and improvement of platform features, facilitates bug detection, and supports new functionalities. Streamers gain from detailed analytics on their audience and growth, while community management is strengthened through effective moderation of toxic behavior. Technical improvements like optimized bandwidth and reduced latency improve streaming quality. Personalized recommendations and notifications help users find relevant content. Market research and competitive analysis keep Twitch ahead of trends and enhance strategic planning. In essence, data analysis fuels growth, user satisfaction, and platform optimization.

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

   Iteration: Loop through a QuerySet to access each object individually.
   Slicing: Slicing an unevaluated QuerySet returns another unevaluated QuerySet, but using the "step" parameter triggers query execution.
   Pickling/Caching: Involves reading results from the database.
   repr(): Calling `repr()` on a QuerySet causes it to be evaluated.
   len(): Retrieve the length of the resulting list.
   list(): Converting a QuerySet to a list forces its evaluation.
   bool(): Evaluating a QuerySet in a boolean context executes the query.

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.

   DataFrames provide greater flexibility and functionality for data manipulation than QuerySets, offering faster in-memory processing and seamless integration with other Python libraries for analysis and visualization. However, they consume more memory and may not always reflect real-time data changes. QuerySets, on the other hand, are more suitable for accessing large data sets directly from databases.

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