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?

I completed the Career Foundry Intro and Immersion course in Full Stack Development. During the course, I gained experience with HTML, CSS, JavaScript, React, Angular, NoSQL (specifically mongoose), and SQL. Before that, I studied business administration, which included an introductory module in business informatics where I learned basic Python. Additionally, I attended a short university course focused on PHP and SQL.

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

I'm familiar with basic Python concepts like if-then statements and loops. I want to learn how to apply Python in real-world projects and how to integrate it with databases.

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.

I anticipate challenges with exercises that involve more math. To tackle them, I plan to work on them during late morning hours (9-12), when I feel most energized and focused.

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 development focuses on creating the user interface that users interact with directly, while backend development handles the hidden operations that support the frontend, like interacting with servers and databases, processing data, and managing requests. If hired for backend programming, I'd focus on tasks such as database management, server-side logic, API integration, and ensuring smooth data processing and communication.

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")

Python and JavaScript both emphasize readability with clear syntax and keywords. Both have various frameworks with ready-to-use features for web development, like setting up web pages and databases. Python, however, offers an extensive library of built-in packages, simplifying complex tasks like math operations and data manipulation. Plus, Python is highly advanced in areas such as Machine Learning, Robotics, and AI.

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 plan to apply Python to real-world projects, improving my web development abilities*
- *I'm interested in learning frameworks like Django to build dynamic web applications.*
- *Master Python essential concepts like variables, data types, loops, and functions to build a strong foundation.*

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?

   Using the iPython Shell offers clearer code readability with syntax highlighting, making it easier to distinguish keywords and lines of code. Unlike Python's default shell, iPython automatically handles indentation for nested statements, enhancing code organization. Overall, iPython provides a more user-friendly environment.

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

Data type	Definition	Scalar or Non-Scalar?
bool	Bools representing True or False values.	Scalar
dictionary	A dictionary in Python is a collection of key-value pairs, where each key maps to a corresponding value.	Non-Scalar
list	A list in Python is a mutable and stores multiple values	Non-Scalar
tuple	A tuple is an immutable and stores multiple values	Non-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.

Lists and tuples both used to store multiple values, but there is a key difference. Lists are mutable, meaning you can add, remove, or modify elements after creation, while tuples are immutable and cannot be changed once created.

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.

I would choose the dictionary data type for the flashcards because it allows me to store the information in key-value pairs. This means I can store the word, definition, and category as keys, making it easy to organize and retrieve data. So, the advantage compared to lists or tuples would be the efficient and fast access to the information needed. Additionally, dictionaries are mutable, so I can modify the flashcards as needed, adding or updating data. This could also be useful, because I might want to add more information to the flashcards later on. However, a limitation is that to retrieve the desired information, I need to know the specific key associated with the information first.

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 = ["japan", "taiwan", "vietnam"]
country = str(input("Enter where you want to travel: "))
if country.lower() in travel_destinations:
   print("Enjoy your stay in", country + "!")
else
   print("Oops, 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 commonly used in conditional statements to make decisions based on if the condition is True or False. The **and** operator returns True only if both conditions are True. The **or** operator returns True if at least one of the conditions is True. The **not** operator returns the opposite boolean value of its condition boolean value.*

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

Functions in Python are reusable blocks of code created using the def keyword. Functions are used when some operation in the code keeps repeating. In that case you can encapsulate the specific task in a function and then just use the function when you need to perform that task. Functions help organize code, avoid repetition, and improve readability.

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 plan to apply Python to real-world projects, improving my web development abilities. → not yet*
- *I'm interested in learning frameworks like Django to build dynamic web applications. → not yet, will happen in achievement 2*
- *Master Python essential concepts like variables, data types, loops, and functions to build a strong foundation. → I have practiced all of these in the last three exercises and start to feel more confident working with them.*

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?

   It's important, because without file storage, data assigned to variables when running a script would be lost once the script stops running, making it unavailable for later use. By using file handling techniques in Python, this data can be stored in files, ensuring it remains accessible even after the scripts stops running.

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?

   Pickles are packaged streams of bytes. The pickling process converts data into these pickles by using the pickle.dump() method and then writes them into binary files. Pickles are used for more complex data structures, such as when working with the dictionary data type and wanting to save that data in a file. Pickles make it possible to retrieve and access this data in 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?

   I will use functions from the os module. To find which directory I am currently in I can use ***os.getcwd()*** *and to change my current working directory I can use* ***os.chdir()****.*

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 use try-except blocks. Inside the try block, I would write the code I expect might have an error, and then in the except block, I write code in case of an error. If no error occurs in the try block, the rest of the code will be executed as normal without executing the code in the except block. However, if an error occurs, the code in the except block will be executed and prevent 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.

   I'm happy with my progress in understanding the for- and while loops, which were previously challenging for me. However, I'm struggling with the "tree" command introduced in this exercise. I couldn't really figure out how to use it.

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 is a kind of coding that organizes code into objects. Each object has its own attributes and methods. The benefits of OOP include code reusability and easier maintenance and debugging.

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

   In Python, a class is a blueprint for creating objects. It defines the attributes and methods of objects. For example, the class "Movie". This class could have the attributes title, release year, and director, along with methods like play() and pause(). An object of this class would be a specific movie, such as "Fargo" released "1996" and directed by "Joel Coen". Another object could represent "Nobody" released "2021" and directed by "Ilya Naishuller". Each object represents a different movie with its own details, but all following the structure defined by the "Movie" class.

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 in OOP is a mechanism where a new class, called a subclass or inherited class, can inherit attributes and methods from an existing class, known as a parent class or base class. This allows the subclass to reuse code and extend the functionality of the parent class. The subclass inherits all the attributes and methods of the parent class, and it can also define its own additional attributes and methods. This promotes code reusability.*
Polymorphism	*In OOP, polymorphism means that a method or attribute can have the same name in different classes but do different things depending on where it's used. Since they're defined separately and exclusively, there's no conflict.*
Operator Overloading	*Operator overloading in OOP lets you redefine how operators work for custom classes. This means that operators like +, -, and others can be customized to work with user-defined objects in addition to their built-in 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?

   Databases are organized collections of data. Their advantages are, that they can ensure data consistency by storing it in a standardized format, simplifying storage and retrieval processes. Also, they offer security measures, such as password protection to safeguard data. Furthermore, databases enable data access through various applications, like Python.

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

Data type	Definition
VARCHAR(n)	*VARCHAR is a variable-length character string data type. It can hold a variable number of characters up to a maximum specified length (n).*
INT	*INT is a data type used to store integer values (whole numbers).*
DATETIME	*DATETIME is a data type used to store date and time values.*

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

   SQLite would be a better choice than MySQL in situations where you need a lightweight and portable database solution. For instance, when working with simple databases or for quick testing purposes. SQLite's ease of setup and portability, with data stored in simple .db files, makes it more convenient than 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?

   JavaScript is primarily used for web development, focusing on interactivity and dynamic content. It's commonly used alongside HTML and CSS to create web applications. Python, on the other hand, is a language known for its readability. It's widely used for various purposes, including web development, data analysis, and machine learning. While JavaScript is more commonly associated with frontend development, Python is more commonly associated with backend development, scientific computing, and more.

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?

   As far as I know, Python is not typically used for frontend development. This suggests that it may have limitations in that area compared to other languages commonly used for frontend development.

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?

   An object-relational mapper makes database conversion easier, by converting the contents and structure of databases into classes and objects that can be interacted with. ORM simplifies database interactions and accelerates the development process. Moreover, it enhances code readability and maintainability by enabling developers to work with familiar programming constructs like classes and objects, rather than dealing directly with 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?

   If I were to start over, I would focus on improving error handling and adding more input validations to ensure a smooth user experience and prevent unexpected crashes.

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've recently developed a Recipe app using Python, which involved creating a menu-driven interface for users to add, view, search, update, and delete recipes. This project allowed me to apply object-oriented programming concepts and interact with a relational database using SQLAlchemy. I gained experience in implementing error handling and managing database operations.

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 had the opportunity to apply all the concepts learned in the exercises.

   b. What's something you're proud of?
      I'm proud of being able to build this application for this exercise by implementing what I learned in the previous exercises.

   c. What was the most challenging aspect of this Achievement?
      The most challenging aspect of this Achievement was trying to catch every possible error and thorough input validation.

d. Did this Achievement meet your expectations? Did it give you the confidence to start working with your new Python skills?
This Achievement met my expectations, giving me confidence in my newfound Python skills. However, there is still much more to learn.

e. 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 keep in mind everything I learned so far.

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 think my study routine was effective, so I will stick to it.

- 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'm proud that I was able to build this application for Achievement 1 by implementing what I learned in the previous exercises. I will build on this in Achievement 2 by keep implementing what I learned.

- 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?
The most difficult aspect of Achievement 1 was trying to catch every possible error and thorough input validations. I deal with that by keep testing my code by running the script. This experience has prepared me for difficulties in Achievement 2 by reinforcing the importance of thorough testing.

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 simple and lightweight, and it offers complete control over the code and structure. But it requires more manual work for web development tasks like handling requests from the client browser and database interactions.
   Django provides pre-built components for common web development tasks, speeding up development. It offers features such as database access and file management. But it has a firm structure, which one must follow and if it's and small application, which doesn't need Djangos build-in features, using Django might add unnecessary complexity.

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

   MVC architecture requires writing code to fetch data, display it, and map it to the URL, while MVT architecture simplifies this process by allowing developers to specify what to present, letting the framework handle the rest.
   In MVC, all components are typically coded in the same programming language, whereas in MVT, the Template is written in a frontend language like HTML or CSS.

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 about Djangos features, such as file management and authentication, and how to use them.

   - What do you want to get out of this Achievement?
     Gain practical experience with Django by building a project using it.

   - Where or what do you see yourself working on after you complete this Achievement?
     After this Achievement I would like to try to build a notes application with 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.*)

   To convert IMDb's website into Django terms, I'd create modules (apps) for movie listings, single movie pages, user profiles, and ratings. Each view would handle specific functionalities like for example displaying movie details, managing user accounts, and handling user interactions with ratings and reviews.

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

   To deploy a basic Django application locally on my system, first, I would run migrations to create the necessary database tables and prepare the backend. (**py manage.py migrate**) *Then, I would start the server to deploy my application to localhost, allowing me to access it through a web browser.* (**py 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 a built-in feature that allows developers to manage the application's data models through a user-friendly interface. During web application development, I would use the Django admin site to perform CRUD operations on the database.

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 the structure of the data is defined in Python classes, which are then translated into database tables. Their benefits are simplifying database interaction, by enabling easy querying and manipulation of 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 helps to catch and fix bugs before they cause serious problems. For example, in a web application, testing user authentication right away can find security problems early on, preventing potential security breaches or login issues. It also saves time, because the code doesn’t have to be manually tested every time changes were 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 receive requests from users' browsers and return a web response. For example, a user wants to view the about us page of a company's website. The view for the about us page renders a predefined HTML template containing information about the company. The view sends the response back to the user's browser, showing them the about us page.

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 class-based views because they are based on Python classes, making them easier to reuse across several apps of the project.

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

- ***Variables**: are enclosed in double curly braces {{ variable }}, and when encountered, they are replaced with their evaluated values. The variable names must adhere to specific naming rules, prohibiting spaces, punctuation characters, and can not start with an underscore.*
- ***Filters**: used to modify variables and structured like this {{ value|modifying_func }}*
- ***Tags**: can create output, control flow, or load external information for later use by variables, structured like this {% tag %}. Some tags require both opening and closing tags, such as {% tag %} ... tag contents ... {% endtag %}.*
- ***Comments**: To add comments in a template, use the syntax {# comment #}*
- ***Template inheritance**: lets you to create a base template with common elements and define 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 aren't dynamically generated or modified by the server while the web app runs. They remain constant. Static files can be 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 in Django displays information from a database table in a specific order.*
DetailView	*Detail View shows all the important info about a specific item from a 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.

   I'm happy that I'm starting to understand the "Django way," but there's still a lot more to learn, and I need to practice more, to really get used to it.

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 important for ensuring that only authorized users have access to an application's resources and functionalities. For instance, in a banking app, authentication verifies the identity of the user before granting access to sensitive financial information, protecting against unauthorized access.

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

   To create a login for a Django web application, first start 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 pages to protect via authentication and redirect users after successful login to one of them. To protect views, add LoginRequiredMixin or login_required to relevant views to prevent access if the user isn't logged in. Then create the logout view and map the URL. Also provide a logout button on 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()	*Verifies a user's credentials against stored data in the authentication backend, returning a User object if valid, and None if not.*
redirect()	*Takes a URL argument and returns a response redirecting to that URL. Used in views to redirect users to another page after processing a request.*
include()	*Imports and renders a template, allowing inclusion of other templates 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 data collected by Thalia, a bookstore website, could help improve user experience by understanding customer preferences, popular genres, and buying patterns. This data could inform personalized recommendations, targeted promotions, and inventory management.

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

- *Iteration: iterate over a QuerySet using a loop, accessing each object individually*
- *slicing: slicing an unevaluated QuerySet returns another unevaluated QuerySet, but Django executes the query if you use the "step" parameter*
- *pickling/caching: reading results from the database*
- *repr(): calling repr() on a QuerySet evaluates it*
- *len(): get the length of the result list*
- *list(): calling list() on a QuerySet forces evaluation*
- *bool():evaluate 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 offer more flexibility and functionality for data manipulation compared to QuerySets. They're faster for in-memory processing and integrate well with other Python libraries for analysis and visualization. On the other side, DataFrames consume more memory and may not always reflect real-time data changes. QuerySets are better for accessing large datasets 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

Reflection Questions

1. Explain how you can use CSS and JavaScript in your Django web application.

    a. ***Inline****: Directly within HTML tags using the style and script attributes.*
    b. ***Embedded****: Within the <style> and <script> tags in the <head> section of your HTML templates.*
    c. ***External Files:*** *By placing CSS files in the static/css folder and JavaScript files in the static/js folder, then linking them to HTML templates using <link> and <script> tags.*

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

    To deploy a Django web app, first, upload the code to GitHub for version control. Next, prepare the application for hosting by updating its settings. Then, configure the hosting service client and server to meet the app's needs. Then update the Django application with tools like dj-database-url for database configuration and psycopg2 for PostgreSQL. Use collectstatic to gather static files and install whitenoise for efficient static file serving.

3. 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?
    Integrating CSS and JavaScript into the project went smoothly.

    b. What's something you're proud of?
    I'm proud that the application is now fully functional and deployed.

    c. What was the most challenging aspect of this Achievement?
    The most challenging part was ensuring the proper serving of static and media files during deployment.

    d. Did this Achievement meet your expectations? Did it give you the confidence to start working with your new Django skills?
    Yes, this Achievement met my expectations and gave me confidence.