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)

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

Remember, you can always refer to [Exercise 1.4](https://careerfoundry.com/en/steps/your-cf-team#receiving-support) of the Orientation course if you’re not sure whom to reach out to for help and support.

Exercise 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

* 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 involves elements such as User Interfaces, HTML and CSS styling, interactivity and all other front-facing functionality that an end user encounters. Backend web dev, on the other hand, handles all the business logic and database centered functionality, as well as sensitive data such as is requested via authentication and authorization.**

**If hired to work on backend programming, I would be working on handling data integrity of the database, or configuring servers and API interactions.**

* 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?

**I would explain to the team that Python offers cleaner, more explicit syntax making for clearer collaboration, as well the fact that Python has a more versatile scope of functionality that makes it better for a cutting edge team. While Javascript and Python both have a deep archive of libraries and frameworks, Python has more strength and applicability in the current programming paradigm. This is mostly because of its applications in data science and automation.**

* 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?

**Goal 1: Apply what I know from Javascript and its popular frameworks to Python (REST APIs, database integration, etc.)**

**Goal 2: Master 2 popular Python libraries/Frameworks**

**Goal 3: Learn how to apply AI and Python to solve a real world problem**

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

* 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?

**iPython Shell has color coding which allows the dev to see different data types represented by color.**

* 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?** |
| Int | Integers | Scalar |
| Float | decimals | Scalar |
| Tuples | immutable list | Non-scalar |
| bool | true or false value | Scalar |

* 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.

**Tuples, unlile lists, are immutable. This means that the order can't be played with, otherwise things like indexing and slicing would not work. Lists, on the other hand, simply keep an array of data grouped in any order.**

* 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.

**The most suitable data structure here would once again be the dictionary type. This allows you to assign one vocabulary flashcard to one definition, quite literally like a dictionary. Since each flash card needs to have a specific term and definition, we don't need a data type that iterates once over many objects, but instead one that contains unique keys to represent each unique card.**

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

* 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!)*

|  |
| --- |
| print("Welcome to my travel app!")  travel\_destination = input("\nWhere do you want to travel?")  if travel\_destination.lower() == "paris":  print("Enjoy your stay in Paris!")  elif travel\_destination.lower() == "london":  print("Enjoy your stay in London!")  elif travel\_destination.lower() == "dubai":  print("Enjoy your stay in Dubai!")  else:  print("Oops, that destination is not currently available.") |
|  |

* Imagine you’re at a job interview for a Python developer role. The interviewer says “Explain logical operators in Python”. Draft how you would respond.

**Logical operators in Python assess whether a statement is true or false in any given condition so that a function can act upon the object or element in question. This includes and, or, not, if, elif, else, and so on.**

* What are functions in Python? When and why are they useful?

**Functions in python are code that performs a defined task and is reusable for that task. They are useful for iterating over large amounts of code, or executing a task that a user requests via an input.**

* 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.

**So far I have made only rudimentary progress in Python, and have yet to delve into libraries and frameworks as I did with Javascript in the full stack immersion course. I would like to discuss with my mentor what I can expect going forward in the course and how I can best prepare to take on my own project, as one of my goals is to make an innovative app on my own.**

Exercise 1.4: File Handling in Python

Learning Goals

* Use files to store and retrieve data in Python

Reflection Questions

* Why is file storage important when you’re using Python? What would happen if you didn’t store local files?

**File storage is important because Python deletes data from assigned to variables after the script is finished executing. Without proper file storage there could be no persistent sessions or usage of the same data for the same script.**

* 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 allow you to store more complex data than you would normally put in a .txt file, because they use binary files. When you're dealing with dictionaries for example, especially on a large scale, pickles can store that structured data in a .bin file.**

* 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 my current directory I can use the os.getcwd() command. To change my directory I can use os.chdir()**

* 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?

**This is done with the use to try-except blocks which handle errors based on expected conditions. The try block allows the code to be run without crashing the whole program and it uses except to give a response when a particular error occurs. This allow the script to keep running if it encounters such errors.**

* 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 finding the course technically interesting and I am glad to be getting some rudimentary basics down. Other than that I am quite disappointed as I have been with the .js portion of this course, due to outdated information/versions, obsolete references, typos, sloppy instructions and and overall frustrating user experience. I have taken beginner courses on python before and I can see the areas in these lessons that are needlessly ambiguous or confusing. I am, however, looking forward to working with my tutor and mentor to make this a valuable learning experience.**

Exercise 1.5: Object-Oriented Programming in Python

Learning Goals

* Apply object-oriented programming concepts to your Recipe app

Reflection Questions

* In your own words, what is object-oriented programming? What are the benefits of OOP?

**Object-oriented programming involves storing data in class structures instead of lone variables. This offers a lot more versatility when calling upon objects and methods, and can more effectively store data that would otherwise be cumbersome to recreate each time a script is executed.**

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

**Classes are used to define the attributes and methods that apply to objects that are assigned to the class. In the real world, you could use the example of Car classes (sport, offroad, exotic, etc.) which each have attributes (horsepower, top speed, handling, suspension type, etc.) which pertain to their relevant objects, or specific cars (Ford Bronco, Lamborghini Diablo, Chevy Malibu)**

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

|  |  |
| --- | --- |
| **Method** | **Description** |
| Inheritance | The concept of subclasses assuming the methods or attributes of their "parent" class by passing the parent as an argument in the subclass |
| Polymorphism | The concept of a single variable or method name that can perform a different action depending on the class it is used in |
| Operator Overloading | Specifically defining functions that determine how operators should execute across classes |

Exercise 1.6: Connecting to Databases in Python

Learning Goals

* Create a MySQL database for your Recipe app

Reflection Questions

* What are databases and what are the advantages of using them?

**Databases are server-based storage systems that allow developers to read, create, update and delete data and utilize that data via a remote or local server.**

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

|  |  |
| --- | --- |
| **Data type** | **Definition** |
| INT | Integers: Whole numbers |
| TEXT | Strings of text |
| FLOAT | Floating numbers: numbers with decimal |

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

**SQLite would be better if I just wanted to quickly test a database without completeing a whole setup process like mySQL. It would also be a better choice if I just wanted to store the data as .db files.**

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

**Python is definitely more readable, while Javascript follows a more traditional syntax. I also note that Python seems more conducive to Object-oriented programming, because of dynamic typing and built in OOP features. I prefer Python at this point due to its versatility and range of functionality.**

* 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 can tell, Python is limited by its speed and performance potential. It's really easy to make bloated, repetitive and memory-intensive scripts that are easy to read and write, but quickly become slow. This can make python executions a potential performance bottleneck in a development stack. I can imagine a full-scale Python project would involve a lot of shortening syntax and off-loading memory intensive functions to other systems whenb possible.**

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

* What is an Object Relational Mapper and what are the advantages of using one?

**An ORM converts the content of a database into classes or objects that can be used in Python. This means you don't need to make adjustments to the SQL for every different database.**

* 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 did well understanding and using basic Python structures such as classes, objects, loops, operators, etc. I feel comfortable getting my code to operate as expected on an internal level. What I have difficulty with and want to improve is accomplishing the same thing with less code and fewer workarounds for problems I encountered. For example, using a different SQL connector than the one I was instructed on in the exercise (this had conflicts).**

* Imagine you’re at a job interview. You’re asked what experience you have creating an app using Python. Taking your work for this Achievement as an example, draft how you would respond to this question.

**I have created a Command Line Python app that works seamlessly and carries a solid, clear foundation for what could easily become a much more complicated recipe app. It has the bones to build a scalable and very user friendly app that allows users to create, update, view, and delete their recipes with no hassle and no required end-user interaction with databases or third parties.**

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

**My coding syntax went well, as I got comfortable using the fundamentals of python structures and was able to use by foundational Javascript knowledge.**

* What’s something you’re proud of?

**I am proud of sticking with this course and not giving up despite numerous issues and the tendency of this course to discourage me, especially given how out of date and sloppily put together it is.**

* What was the most challenging aspect of this Achievement?

**The most challenging aspect of this achievement was connecting what I learned on the syntax and structure level, with the other technology I was working with (mySQL, sqlconnector, etc). I would often get confused how the script relates to things happening behind the scenes with mySQL and the databases.**

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

**No, this course highlighted to me the importance of upkeeping online information about web development clear and up to date, lest you lead your reader into the pits of hell trying to resolve conflicts and sort out this course's phantom limbs from obsolete versions of different technology.**

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

**I want to focus more on developing lean code and avoid my tendency to allow python script to get bloated and unnecessarily lengthy.**

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

Pre-Work: Before You Start Achievement 2

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

* Was your study routine effective during Achievement 1? If not, what will you do differently during Achievement 2?
* Reflect on your learning and project work for Achievement 1. What were you most proud of? How will you repeat or build on this in Achievement 2?
* What difficulties did you encounter in the last Achievement? How did you deal with them? How could this experience prepare you for difficulties in Achievement 2?

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

Remember that can always refer to [Exercise 1.4](https://careerfoundry.com/en/steps/your-cf-team#receiving-support) of the Orientation course if you’re not sure whom to reach out to for help and support.

Exercise 2.1: Getting Started with Django

Learning Goals

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

Reflection Questions

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

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

* 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.*)

* In your own words, describe the steps you would take to deploy a basic Django application locally on your system.
* Do some research about the Django admin site and write down how you’d use it during your web application development.

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

* Do some research on Django models. In your own words, write down how Django models work and what their benefits are.
* 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.

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

* Do some research on Django views. In your own words, use an example to explain how Django views work.
* 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?
* Read Django’s documentation on the [Django template language](https://docs.djangoproject.com/en/3.2/ref/templates/language/#templates) and make some notes on its basics.

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

* In your own words, explain Django static files and how Django handles them.
* 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 |  |
| DetailView |  |

* 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.

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

* In your own words, write down the importance of incorporating authentication into an application. You can take an example application to explain your answer.
* In your own words, explain the steps you should take to create a login for your Django web application.
* 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() |  |
| redirect() |  |
| include() |  |

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

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

* Explain how you can use CSS and JavaScript in your Django web application.
* In your own words, explain the steps you’d need to take to deploy your Django web application.
* (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.
* You’ve now finished Achievement 2 and, with it, the whole course! Take a moment to reflect on your learning:
* What went well during this Achievement?
* What’s something you’re proud of?
* What was the most challenging aspect of this Achievement?
* 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.