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 web development is when you work on the client side of an application. You will be working on the elements of the app that users will see and interact with. Frontend developers use languages like HTML, CSS, and Javascript. Backend web development is when you work on the server side of an application. You will be working with databases and creating APIs. Backend developers use Node.js, Express, MongoDB, etc. If I was hired as a backend developer, I would be working on creating databases and using API routing to use CRUD operations to manipulate the data inside the database.
* Imagine you’re working as a full-stack developer in the near future. Your team is asking for your advice on whether to use JavaScript or Python for a project, and you think Python would be the better choice. How would you explain the similarities and differences between the two languages to your team? Drawing from what you learned in this Exercise, what reasons would you give to convince your team that Python is the better option?

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

* Javascript and Python are both high-level scripting languages. They both use dynamic typing. What makes Python great is that they have many open-source packages that can be installed. Python also comes pre-installed with common web operators like URL routing, form validation, etc. Python is very readable and easy to debug. Python also has one of the largest communities.
* Now that you’ve had an introduction to Python, write down 3 goals you have for yourself and your learning during this Achievement. You can reflect on the following questions if it helps you. What do you want to learn about Python? What do you want to get out of this Achievement? Where or what do you see yourself working on after you complete this Achievement?
* I want to learn how to use Python to create AI logic and algorithms. In this Achievement, I want to get a strong base understanding on how I can use Python for my development goals. I see myself doing future research on how I can use Python to create programs that use AI and other ways I can use Python in my future projects.

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?
* Code in iPython's Shell is clearer to see because of the syntax highlighting. iPython also automatically indents nested statements for you which makes it quicker to type code. It is easier to test small batches of code because you get a result as soon as you enter a command.
* 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 | Data type used for whole numbers | Scalar |
| list | Mutable and ordered sequence | Non-Scalar |
| float | Data type used for numbers with decimals | Scalar |
| str | Immutable array of characters | Non-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 are immutable, but they are faster to read and access. This would be useful if you had a large amount of data that didn't need to be changed. Lists are mutable and more flexible. Elements can be modified and deleted. It is easier to add elements into a list and you can sort the data in a list.
* In the task for this Exercise, you decided what you thought was the most suitable data structure for storing all the information for a recipe. Now, imagine you’re creating a language-learning app that helps users memorize vocabulary through flashcards. Users can input vocabulary words, definitions, and their category (noun, verb, etc.) into the flashcards. They can then quiz themselves by flipping through the flashcards. Think about the necessary data types and what would be the most suitable data structure for this language-learning app. Between tuples, lists, and dictionaries, which would you choose? Think about their respective advantages and limitations, and where flexibility might be useful if you were to continue developing the language-learning app beyond vocabulary memorization.
* For each card I would use a dictionary because I can use the keys to easily access, modify, and print the value. I would use a list to store all of the cards because you can add, modify, and delete cards from the list. The cards would also be in sequential order so you can flip through the cards easily.

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!)*

|  |
| --- |
| destination = input("Enter one of these destinations you would like to go to (Tokyo, New York, London): ")  if destination == "Tokyo" or destination == "New York" or destination == "London":  print("Enjoy your stay in " + destination + "!")  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.
* If you have two or more conditions in an expression that use the 'and' operator, both conditions have to be True for the expression to return True. Otherwise, it will return False. If the conditions use the 'or' operator, only one of the conditions has to be True for the expression to return True. The only way an expression using the 'or' operator will return False is when all of the conditions are False. The 'not' operator will return the opposite value of the expression. If the expression originally returns True, if you add the 'not' operator, it will return False instead and vice versa.
* What are functions in Python? When and why are they useful?
* A function is a set of instructions that process or manipulate your code. It is useful to use functions if you know that that block of code will be reused many times. Instead of writing the same code over and over, you can just call the function. This makes your code easier to read. You can make the functions more flexible by adding arguments. You can plug in different values into the same function instead of rewriting the code for each new value.
* 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.
* Conditional statements are an integral part of creating AI. I feel like I'm ready to try to script more complex programs now that I have a good understanding of the building blocks of Python scripting. I'm excited to learn more about how Python is applied in real-world projects.

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?
* It is important to use file storage because you can use the saved data in the future. If we didn't store local files, the data will be deleted when the program ends.
* 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 pickle is a stream of bytes. We should use pickles when we have to save or load complex data like lists, tuples, or dictionaries. When you load data using the pickle method, the data is parsed for you, which will save you a lot of time and effort.
* 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 what directory you're in, you would use os.getcwd(). If you wanted to change your current working directory, you would use os.chdir("<dir path>").
* 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 place my code in a try block and handle any errors in an except block. If there is anycode I would like to execute if there is no error, I would use an else block. If there is code I would like to execute after the try-except-else blocks, I would use a finally block.
* 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 about learning about file writing and reading. I feel like this will be extremely important to know for my future tasks/jobs. I would like practice using everything I learned so far to try to make some more complex programs.

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?
* OOP is a programming mindset that is focused on creating objects that contain methods and attributes to handle the app's logic. One benefit of OOP is that you can create a class to define what attributes and methods the objects will have. You can then use this class to create many objects with the same attributes and methods. You can use the class methods to modify the data inside of a specific object. You can also use inheritance to have a subclass that adds additional attributes and methods to the original class.
* What are objects and classes in Python? Come up with a real-world example to illustrate how objects and classes work.
* Classes are the blueprints used to define what data attributes and methods the object will have. Objects are the data that is created by using a class to initialize it. So if I make a class like Recipe:

class Recipe(object):

def \_\_init\_\_(self, name, ingredients, cooking\_time):

self.name = name

...

I can create many recipes using this blueprint.

tea = Recipe("Tea", ["Tea Leaves", "Sugar", "Water"], 5)

cake = Recipe( ...

...

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

|  |  |
| --- | --- |
| **Method** | **Description** |
| Inheritance | This is when you create a subclass that inherits the data attributes and methods from the parent class. You can add additional attributes and methods to the subclass that the parent class will not be able to use. A parent can have many subclasses. If you add any additional attributes you have to call the parent class's \_\_init\_\_ inside the subclasses \_\_init\_\_ method. |
| Polymorphism | This is when many classes use the same method name but have different logic. The method chosen is based on what class is calling the method name. You can call the method from the class itself or from the object. |
| Operator Overloading | This is when your class uses custom logic when the user uses one of the operators like + or -. These operators correspond with a specific method name. For example, + is \_\_add\_\_() and - is \_\_sub\_\_(). There are also overloads for comparison operators. For example, > is \_\_gt\_\_(). You can also make your class easily printable using \_\_str\_\_(). If you define operator overloads for the comparison operators, you can make your class sortable in a list. |

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 a group of tables that contain columns that are used for data attributes and rows that are used to add new entries into the table. Some advantages of using databases is that it is an quick and easy way to add, delete, modify, read, and delete large sets of data. All changes will be saved to a database server so you can access the same data the next time you open your program. Some other benefits is that your data is secured because you can only access the server with a password and you can use a database in many programming languages because of the standardized format of the data.
* List 3 data types that can be used in MySQL and describe them briefly:

|  |  |
| --- | --- |
| **Data type** | **Definition** |
| VARCHAR(n) | A string of characters where 'n' is the max number of characters the string can have. |
| INT | Used for storing whole numbers. |
| FLOAT | Used for storing decimal numbers. |

* In what situations would SQLite be a better choice than MySQL?
* If you have a small and simple database, SQLite would be a better choice.
* 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 seems more flexible because the syntax is simpler to implement. It is also easier to implement OOP in Python because it is simpler to create classes and implement OOP principles like inheritance and polymorphism. With TypeScript it is easier to keep data consistent by assigning them types. Overall I enjoy using both.
* Now that you’re nearly at the end of Achievement 1, consider what you know about Python so far. What would you say are the limitations of Python as a programming language?
* I think one limitation is that you can't assign data types to a variable. So it is harder to check if the data is valid and can lead to more runtime errors.

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?
* ORM is a tool that simplifies database conversion. It converts your database content into classes that you can easily read, update, delete, etc. This allows database operations to be more object-oriented and saves you time because you don't have to type out full SQL commands.
* 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?
* Something I think I did well while creating the app was identifying parts of the code that could be encapsulated and reused in many functions. One thing I would change is instead of exiting the function when the user inputs invalid data, I would loop to restart the function. If the user wanted to cancel I would let them enter a command like "cancel."
* 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.
* One app that I worked on using Python is a recipe app that lets you create a list of recipes. To hold the recipe information I used a MySQL server. Each recipe entry had an autoincremented id that was used as the primary key, a name, a cooking time, and a list of ingredients that was put into a string. The user has the option to create, view, edit, or delete the entries on this table. I used an ORM to make it easier to let the user do these CRUD operations.

* 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?
* I think learning how to utilize all of the various tools, types, databases, etc. available in Python went well.
* What’s something you’re proud of?
* I'm proud of making my first app in Python using MySql and an ORM. I think it went very well and I can use these skills to make even larger apps.
* What was the most challenging aspect of this Achievement?
* The most challenging part was getting used to the new syntax and functionalities of Python.
* Did this Achievement meet your expectations? Did it give you the confidence to start working with your new Python skills?
* I'm very happy with what I learned from this achievement. I'm very confident about using the skills I learned to develop future apps.
* What’s something you want to keep in mind to help you do your best in Achievement 2?
* I want to revisit the topics covered in this achievement to make sure I don't forget anything that can help in the second achievement.

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 somewhat effective. I think studying in a better environment to focus would help.
* 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 was most proud of using MySQL and an ORM to develop a fully functional app. I will try to focus on retaining the info I learned from the first achievement and try to identify how I can use the skills in Achievement 2 to build on those previous skills.
* 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 main difficulty was getting used to the new syntax and tools that are available in Python. In this experience, I did some extra research to help me with any issues. I will do the same for any 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.