CAREERFOUNDRY

Python for Web Developers Learning Journal

Objective

We find that the students who do particularly well in our courses are those who practice metacognition. Metacognition is the art of thinking about thinking; developing a deeper understanding of your own thought processes. With the help of this Learning Journal, you'll broaden your metacognitive knowledge and skills by reflecting on what you learn in this course.

Thanks to this Learning Journal, when you finish the course you'll have a complete and detailed record of your learning journey and progress over time. We really recommend that you take the time to complete this Journal; students do better in CF courses and in the working world as a result!

Directions

First complete the pre-work section before you start your course. Then, once you've begun learning, take time after each Exercise to return to this Journal and respond to the prompts.

There will be 3 to 5 prompts per Exercise, and we recommend spending about 10 to 15 minutes in total answering them. Don't overthink it—just write whatever comes to mind!

Also make sure that, once you've started filling this document in, you upload it as a deliverable on the platform. This is so that your mentor can also see your Journal and how you're progressing over time. Don't worry though—what you write here won't affect how you're graded for the Exercise tasks. The learning journal is mostly for you and your self-evaluation!

Pre-Work: Before You Start the Course

Reflection questions (to complete before your first mentor call)

1. What experiences have you had with coding and/or programming so far? What other experiences (programming-related or not) have you had that may help you as you progress through this course?
    a. The only coding experience that I've had to this point is the CareerFoundry Intro to Frontend Development and Immersion courses. Prior to that, I have zero experience whatsoever with it–not even a college course. Aside from programming, I have earned a Bachelors Degree and worked in a few different jobs to this point as a career changer, so having gone through adversity and coming out on the other side is always helpful not just for the course, but as a person. Additionally, my dad works in IT as well, so that is helpful to have conversations and look over things together.
2. What do you know about Python already? What do you want to know?
    a. I don't know anything about Python yet. I've heard of it many times since beginning this course and I'm aware that it is a critical skill for many jobs in the industry today. Other than knowing it is a backend language, I know nothing.

3. What challenges do you think may come up while you take this course? What will help you face them? Think of specific spaces, people, and times of day of week that might be favorable to your facing challenges and growing. Plan for how to solve challenges that arise.
    a. Imposter syndrome, which has been extremely present throughout the length of this entire program so far. It's not a feeling I am new to, so it just has to be overcome by continuing on and making sure I take care of myself outside of the program as well.
    b. Another challenge I face regularly is the feeling that I am racing against the clock. I learn more slowly than I had anticipated and have a tendency to take too long trying to make something perfect before I move on. I am taking much longer than I planned on taking for this program as a whole, and it is a negative feeling/reality that I face daily.

Remember, you can always refer to Exercise 1.4 of the Orientation course if you're not sure whom to reach out to for help and support.

Exercise 1.1: Getting Started with Python

Learning Goals

- Summarize the uses and benefits of Python for web development
- Prepare your developer environment for programming with Python

Reflection Questions

1. In your own words, what is the difference between frontend and backend web development? If you were hired to work on backend programming for a web application, what kinds of operations would you be working on?
    a. Put simply, the frontend (client-side) is what the user interacts with directly (the user interface). The frontend is built with languages like HTML to build the skeleton, CSS for styling, and JavaScript to bring it to life if you so choose. It is what displays the contents of the site or app to the user. The backend (server-side) is the brains of the operations that contains the logic. The backend decides what to do when users make an interaction. For example, if a user types their name and email in form fields and clicks a "Submit" button, the backend figures out what to do with that information (sending and storing in a database, receiving a response, and sending that response back to the frontend).
    b. Picture a pizza shop example: The customer is the user, the order ticket is the frontend, the cashier that inputs the order is the backend, the computer and food operation system is the logic. The cashier takes the order ticket from the customer, inputs it, and returns the response to the user.

2. Imagine you're working as a full-stack developer in the near future. Your team is asking for your advice on whether to use JavaScript or Python for a project, and you think Python would be the better choice. How would you explain the similarities and differences between the two languages to your team? Drawing from what you learned in this Exercise, what reasons would you give to convince your team that Python is the better option? *(Hint: refer to the Exercise section "The Benefits of Developing with Python")*
    a. Python and JavaScript are both scripting languages where commands are executed line by line through a standardized syntax as opposed to all at once. Python was designed with quick deployment and easy readability in mind so that someone with a baseline understanding of programming would be able to grasp the concepts. This readability is thanks to the indentation of separate blocks of code. Similar to JavaScript, Python uses keywords that are clear and easy to understand to make commands (*import, return, continue, etc.*). A big similarity between JavaScript and Python is the use of dynamic typing, which essentially allows variables to have any data type as their value without producing errors. This is opposed to TypeScript that uses static typing. For example, with Python and JavaScript's use of dynamic typing, a variable called "z" can be a number at first and be reassigned to a string later on without producing errors. Both languages are open-source and can be used by anyone. With Python, it is almost always a backend language whereas JavaScript is most common for both, or more so frontend. Also, Python is certainly more readable, as it was designed with that in mind. JavaScript can be quite readable too, but you are likely to have an easier time with Python. Also, Python packages that come with many of its frameworks often integrate more smoothly for specific tasks than those of JavaScript which uses npm for packages. Development and deployment speed is quicker with Python and it also comes with built-in tools for common tasks where you might need to manually integrate many of these types of tools with JavaScript.

3. Now that you've had an introduction to Python, write down 3 goals you have for yourself and your learning during this Achievement. You can reflect on the following questions if it helps you. What do you want to learn about Python? What do you want to get out of this Achievement? Where or what do you see yourself working on after you complete this Achievement?
    a. **Goal 1**: To have an intermediate understanding of Python and in what situations / projects it would be best to use.
    b. **Goal 2**: By the end of this Achievement project, I want to be able to have a fluent conversation about Python and be able to discuss terminology, use cases, and display technical skills that I learned in an interview setting.
    c. **Goal 3**: Have a better understanding of what jobs I want to apply for. I want my Python learning journey to give that final push I need to decide what I want my first job to entail and whether it will be frontend, backend, or full-stack.
    d. **Goal 4**: Be able to move through Achievement 2 more quickly and efficiently by the time I finish Achievement 1, having just gained enough of a Python foundation to work more smoothly.

Exercise 1.2: Data Types in Python

Learning Goals

- Explain variables and data types in Python
- Summarize the use of objects in Python
- Create a data structure for your Recipe app

Reflection Questions

1. Imagine you're having a conversation with a future colleague about whether to use the iPython Shell instead of Python's default shell. What reasons would you give to explain the benefits of using the iPython Shell over the default one?
    a. I would tell a future colleague that the iPython shell offers features like auto-indentation, auto-completion, and color-coding/syntax highlighting for readability and easier testing/ debugging. All of these features make it a clear choice as a preferred shell in most cases.

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?
int	Integers. Can be positive or negative whole numbers	Scalar
bool	Value of either true or false. Good for conditions	Scalar
tuple	Arrays that can store multiple values of any data type	Non-Scalar
dictionary	Stores values & objects within itself indexed by keys	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.
    a. In Python, a list is a type of ordered sequence, similar to a tuple. However, the main difference between a list and a tuple in Python is that lists are mutable, meaning that the internal elements of a list can be modified, rearranged, deleted, added, etc, without affecting the rest of the sequence as opposed to the immutability of tuples, which are static. Lists would generally be the most recommended data structure when there is or will be a need for reordering or modification. While lists are more useful for messing with elements inside of them, tuples are faster to read and access when dealing with large amounts of data. As far as operations, the ones that you can perform on tuples can be performed on lists as well.

4. In the task for this Exercise, you decided what you thought was the most suitable data structure for storing all the information for a recipe. Now, imagine you're creating a language-learning app that helps users memorize vocabulary through flashcards. Users can input vocabulary words, definitions, and their category (noun, verb, etc.) into the flashcards. They can then quiz themselves by flipping through the flashcards. Think about the necessary data types and what would be the most suitable data structure for this language-learning app. Between tuples, lists, and dictionaries, which would you choose? Think about their respective advantages and limitations, and where flexibility might be useful if you were to continue developing the language-learning app beyond vocabulary memorization.
    a. For creating individual flashcards, I've chosen dictionaries as the data structure. Dictionaries are ideal for organizing labeled data, which is great for something like flashcards, where each piece of info (word, definition, category) can be labeled as a key. Values can be any data type, giving flexibility to expand on each flashcard's info if needed (difficulty level, synonyms, etc.). Also, lists are mutable, so key-value pairs can be added or deleted at any time (great for growth and expansion). For the outer structure containing all flashcards, a list would be best for its flexibility and mutability. A list can store and organize multiple flashcards as individual items, so users can add, delete, and rearrange flashcards. Lists are ordered, so users can have a set order or shuffle the flashcards for quizzing. Because lists are mutable, changes to individual flashcards within the list don't affect the overall sequence, which makes them great for managing multiple flashcards as a flexible and easily accessible collection.
    b. One reason I had previously considered a list for the individual flashcards is because each piece of info on the flashcards (word, definition, category) would all be strings, making it simple and easy. However, this choice would make for little flexibility and that can be tough if the app were to eventually grow or add keys to the flashcards requiring different data types.

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!)*

```
destination = input('Where would you like to travel? ')

dest_1 = 'London'
dest_2 = 'Japan'
dest_3 = 'India'

if destination == dest_1:
    print('Enjoy your stay in London!')
elif destination == dest_2:
    print('Enjoy your stay in Japan!')
elif destination == dest_3:
    print('Enjoy your stay in India!')
else:
    print('Oops, that destination is not currently available.')
```

2. Imagine you're at a job interview for a Python developer role. The interviewer says "Explain logical operators in Python". Draft how you would respond.
    a. Logic operators are best used when you need to check for more than one condition at the same time (e.g., if a car is blue and less than 10 years old). You would use the boolean logic operators "and" and "or" to combine the checks. Truth tables are sometimes used to describe what an output would look like for a logical statement given its inputs.
        i. If you need to check if both conditions are met, use the "and" operator. If both conditions are met, it will return a boolean value of True. If one or both condition are not met, it will return a boolean value of False. The first and second conditions must be met in order to return True with the "and" operator.
        ii. The "or" operator is different in that it check whether either (not both) of the conditions are met. If both are met, it's True. If both are False, it's False. If one of the two is met, it's True.
        iii. There is also the "not" operator. The "not" operator doesn't require a condition on either side of it because it isn't looking for how the two conditions may or may not relate to each other. It is actually used to reverse the result of what follows it and, essentially, flips whether or not a conditional statement was True or False. For example, if the conditional statement `print(5 > 3)` printed True, the conditional statement with the "not" operator `print(not 5 > 3)` would print False. It just does the opposite, or reverse.

3. What are functions in Python? When and why are they useful?
    a. Functions in python are blocks of code that process and work with data in order to perform tasks and actually make things happen. Python comes with some built-in

functions like print() and append(), but you can create your own functions as well. If you need something to happen in your code, you can make custom functions to condense the amount of code and save on time as well. Functions can be called on throughout your code so that you don't have to write something new each time you need something to happen (like filtering lists). In Python, you use the `def` keyword to before the name of the function to define it. Function names have to be all lowercase and/or underscores, and are followed by a set of parentheses (which you can include parameters in if need be). You can call functions in your code by typing the function name and the set of parentheses.

4. In the section for Exercise 1 in this Learning Journal, you were asked in question 3 to set some goals for yourself while you complete this course. In preparation for your next mentor call, make some notes on how you've progressed towards your goals so far.
    a. I've gained a ton more knowledge on Python since starting the course (had none before).
    b. I've gained a better appreciation for the backend so far. Not as stale/boring as I thought.
    c. I could have a conversation about Python now even through three Exercises. Don't know much yet, but I can at least talk about it as it seems easy to relate to in the way it that it reads.

Exercise 1.4: File Handling in Python

Learning Goals

- Use files to store and retrieve data in Python

Reflection Questions

1. Why is file storage important when you're using Python? What would happen if you didn't store local files?
    a. File storage is so important with Python because if you are only running commands on the CLI, your data is not getting saved anywhere and you would essentially be left with standalone sessions in your Terminal, and that doesn't serve much of a purpose beyond testing and learning for yourself. You can store data in files in different ways and read and write to them, so it is critical that you learn about the different options and methods.

2. In this Exercise you learned about the pickling process with the **pickle.dump()** method. What are pickles? In which situations would you choose to use pickles and why?
    a. Pickles are a way to store complex Python data structures, like dictionaries and lists, in a binary format that preserves their structure. This is useful when text files can't handle the complexity of the data, such as saving a list of recipes with nested attributes. The **pickle.dump()** method serializes the data into a binary file, and **pickle.load()** restores it, allowing easy reuse of structured data across programs.

3. In Python, what function do you use to find out which directory you're currently in? What if you wanted to change your current working directory?
    a. For accessing files across different directories, Python uses the **os** module that includes features for file management. To figure out which directory you are currently in: **os.getcwd()** command. To change the current / working directory: **os.chdir()** command.

4. Imagine you're working on a Python script and are worried there may be an error in a block of code. How would you approach the situation to prevent the entire script from terminating due to an error?
    a. I would add a try-except block. You try a block of code where you expect that an error can occur. If no errors are found, the code is executed as normal. If an error does occur, the except block will notify the user of the specific error (if you have an except block for that error) and guide them in how to fix it.

5. You're now more than halfway through Achievement 1! Take a moment to reflect on your learning in the course so far. How is it going? What's something you're proud of so far? Is there something you're struggling with? What do you need more practice with? Feel free to use these notes to guide your next mentor call.
    a. I'm most proud of making it this far. There have been moments of confusion, mostly having to do with the course structure causing some confusion. I've found so far that Python is easy to understand as far as the actual language itself and how it relates a lot to the English language in how it reads. Obviously it all takes practice and repetition to fully understand on a consistent basis, but it seems like the most readable language that I've learned so far, and I know it was designed with that in mind. I need a ton more practice with all of it, but I like it more than I thought I would so far.

Exercise 1.5: Object-Oriented Programming in Python

Learning Goals

- Apply object-oriented programming concepts to your Recipe app

Reflection Questions

1. In your own words, what is object-oriented programming? What are the benefits of OOP?
    a. Object-oriented programming (OOP) is a way of writing code that organizes data (called attributes) and actions (called behaviors) into reusable classes, which act like templates for real-world objects. It follows the DRY principle (Don't Repeat Yourself) by keeping code non-repetitive and non-redundant, making it more efficient and easier to manage. By grouping related data and methods into classes, OOP allows you to build clear, scalable, and reusable code, making it simpler to work on complex projects while keeping everything organized and easy to expand. Basically, OOP is the process of abstracting data and methods into classes.

2. What are objects and classes in Python? Come up with a real-world example to illustrate how objects and classes work.
    a. In Python, everything is an object. Most objects you work with in Python can be broken down into the data they contain and into the methods you can use to interact with the data. One class can contain multiple objects. A class is like a blueprint or template for creating objects. It defines attributes and methods that the objects created from the class will have. An object is an instance of a class, meaning that it is specific instance created based on the class.
    b. Example: A class called Reptile. The class Reptile could have attributes like “species”, “color”, and “type”. The class could also have methods like “walk”, “fly”, and “slither”. Each specific reptile (like Dragon or Gecko) would be an object created from the Reptile class. So here, Reptile is the class and “dragon” and “gecko” are objects created from the class, each with its own specific attributes and methods.

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

Method	Description
Inheritance	OOP feature that allows a subclass to inherit attributes and methods from a parent class. Good for reusing code without rewriting it. Makes code more organized and reduces duplication.
Polymorphism	Means “many forms”. Allows objects of different classes to be treated like they belong to the same class. Done by using the methods with same name but behaving differently depending on the object calling them. Makes it easy to write flexible and reusable code, especially when working with multiple related classes.
Operator Overloading	Allows you to customize how standard Python operators like (+, -, and *) work with your objects. e.g., if you have a class “Height”, you can define what happens when you add two “Height” objects using the “+” operator by creating a special method called “__add__”. This makes it easy to perform ops on custom objects without writing extra code. It keeps your code cleaner and lets operators like “+” behave in ways that make sense for your program.

Exercise 1.6: Connecting to Databases in Python

Learning Goals

- Create a MySQL database for your Recipe app

Reflection Questions

1. What are databases and what are the advantages of using them?
    a. Databases are a way to store data in a structured and electronic way. Operations like creating, reading, updating, deleting, and more can be performed on databases. Some advantages to databases over regular local storage include keeping data in a standardized format, which makes it easier to store and access. They can be made more secure through password access. You can access databases using apps other than Python. Database Management Systems (DBMS's) offer interfaces allowing users to read, store. And modify data easily.

2. List 3 data types that can be used in MySQL and describe them briefly:
    a. MySQL data types are different from Python. They don't include complex data structures such as dictionaries and lists.

Data type	Definition
VARCHAR(n)	String of variable length, with n representing the max number of
INT	Standard integers (1, 2, 3)
FLOAT	Floating-point decimal numbers (1.2, 5.6, 7.532)

3. In what situations would SQLite be a better choice than MySQL?
    a. SQLite is the portable version of MySQL and doesn't require setup or installation. Allows you to store data in simple .db files which you can access and modify directly from apps like Python. SQLite would be recommended when working with simple databases or if you were testing a database and didn't feel the need to set up an entire database engine.

4. Think back to what you learned in the Immersion course. What do you think about the differences between JavaScript and Python as programming languages?
    a. Python is more readable and simple with indentation. JavaScript is a bit more complex (use of {} and ; etc.), but it is also a bit more flexible with its syntax.
    b. Python is mainly used on the server-side of apps while JavaScript is the primary language of the web and runs in browsers. JavaScript is popular and well-known for full-stack development with a single language for all aspects of it (frontend and backend).

5. Now that you're nearly at the end of Achievement 1, consider what you know about Python so far. What would you say are the limitations of Python as a programming language?
    a. It is much more of a server-side language, so creating a full-stack app with just Python isn't a good choice as opposed to JavaScript. Some may not consider it a limitation, but the indentation that Python uses is very strict, so it is important to be extra careful. Additionally, due to Python's dynamic typing as well as it being OOP in nature, it is more memory-intensive than other languages. Might not be ideal for apps requiring a ton of data processing. Lastly, while not experiencing it yet personally, Python code runs slower as an interpreted language as opposed to compiled languages. Not great for real-time apps, etc.

Exercise 1.7: Finalizing Your Python Program

Learning Goals

- Interact with a database using an object-relational mapper
- Build your final command-line Recipe application

Reflection Questions

1. What is an Object Relational Mapper and what are the advantages of using one?
    a. An ORM makes database conversion much easier for you, especially if you're moving from one DBMS to another. An ORM converts the contents and structure of your database into classes and objects that you can interact with directly. This allows you to not have to worry about SQL syntax.
    b. This saves you a lot of time. The simplification that ORMs offer is part of "Pythonic style", which is a style particularly well suited to Python's features

2. By this point, you've finished creating your Recipe app. How did it go? What's something in the app that you did well with? If you were to start over, what's something about your app that you would change or improve?
    a. I did well with formatting the terminal output. I have a good eye for optimizing readability for the user and making sure everything is easy to see / find / access / understand.
    b. I was able to use and improve on pieces of code from the previous Exercise that was able to be reused/refactored for this Exercise. I was able to save myself time and also improve parts of my code through this process.
    c. I would like to get better and more consistent with how I name variables if I were to create this again.
    d. If I were to do this again, I would find ways to simplify parts of my code. There were parts of my file that I opted to make longer because it was easier for me to read and understand as a beginner. For example, a couple parts could be one-liners that I opted to make three-liners (creating an empty list, then adding to it after instead of just adding a condition inside the empty list).
    e. I would become better at writing comments and be more structured about it to maintain readability a bit better.
    f. I would make more frequent commits as I go instead of doing big chunks at the end. I was better at this with other projects. I think some of this came from a fear of committing things that were wrong in my code, even though I could have just committed the right answer again later if that was the case.

3. Imagine you're at a job interview. You're asked what experience you have creating an app using Python. Taking your work for this Achievement as an example, draft how you would respond to this question.
    a. I have experience in creating a command-line version of an interactive Recipe application with Python. The application allows users to follow along and add inputs to the app's prompts that will allow them create, view, edit, search for and delete recipes all in the command-line. I worked with running servers, creating functions, the IPython shell, testing and debugging, for and while loops, MySQL, Python objects, object-relational-mapping, and much more.

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. Functions and operations are all working. I found, like many others, that Python is more readable to the average human. It is easier to understand as you write it.
        ii. I got better at writing for and while loops as well as try-except blocks and if-elif-else conditions.
    b. What's something you're proud of?
        i. I am proud of overall creating my first Python project from scratch. I am proud of being able to retain a solid amount of Python fundamentals and look forward to building off of that more in the next Achievement.
    c. What was the most challenging aspect of this Achievement?
        i. The most challenging aspect of this Achievement was simply learning a new language immediately after learning a bunch of other languages. Understanding more about exactly what Python is and what it is best used for would be helpful. I understand that it is primarily a backend language, but I wish we would dive more into that (what Python jobs would look like, most common real-world job activities, best use cases, etc.). Implementing new things one after the other is tough (learning MySQL queries and then switching to ORM). Some instructions were unclear and left to the imagination of myself. Learning a new language for the first time should provoke many thoughts and ways of thinking about problems, but more guidance as a first-timer would have been great.
    d. Did this Achievement meet your expectations? Did it give you the confidence to start working with your new Python skills?
        i. It certainly gave me the confidence to continue building on my new Python skills, but I certainly do not feel job ready yet. I need more practice, just like I imagine many others would at this point. I wish that more things kept building on the last thing more often. For example, we spect time on MySQL and as soon as I was starting to get the hang of it (conn, cursor, execute, SQL queries etc.) we started moving on to ORM (session, engine, objects, etc.) Like many other Achievements, there wasn't much time to "perfect" one thing. It is constantly moving.
    e. What's something you want to keep in mind to help you do your best in Achievement 2?
        i. Make more frequent commits. Ask questions for frequently instead of wasting too much time trying to figure everything out solely by myself and the internet. Test all functions thoroughly as I go instead of waiting to do a bunch at once at the end. Unit testing AND Integration testing. Write out design flows before I begin a project to imagine the concepts of the app from a user perspective.

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?
    - My study routine for Achievement 1 was pretty good and identical to that of the Intro and Immersion courses. One thing I hope to change now is to not burn myself out and try to stick to set hours for myself as if it were a real job. I have a habit of going for way too long which throws me off for the following day, etc.
    - I'd like to get even better at my note taking skills so that they are more readable, I understand my own notes better, and I waste less time.
- 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 creating a functioning, error-free Python project from scratch.
    - I was proud of my resilience and drive to continue on when things were extremely tough.
    - I will continue building off of my previous successes by reflecting on what worked and what did not work. I will continue taking time for myself and taking a break when needed.
- 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 difficulties I encountered was learning all new info in such a short amount of time, though that was unique to Python or Django.
    - Some of the program material is outdated with technologies being deprecated or new versions coming out for things, etc. Having to learn the concepts as well as research the correct way to do something (since they were now incorrect in the reading material) was difficult, as it was in previous Achievements as well.
    - Not having enough opportunity to practice. I would learn how to do something new, implement it, and move on. For example, learning SQL queries and then immediately moving onto ORM and never truly going back to MySQL.
    - All of these difficulties and experiences helped me evolve my approach and mindset naturally as I progressed through them.

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?
    a. Advantages of Django:
        i. It is a web framework built with Python, so it is implemented in Python
        ii. Fast development
        iii. Fast processing
        iv. Follows DRY (Don't Repeat Yourself) principles, which keeps your code non-redundant, non-repetitive, and efficient
        v. The Django project structure is modular and reusable
        vi. Django supports Content Delivery Networks (CDNs) and content management
        vii. Scalability
        viii. Security
        ix. Support community and documentation
    b. Drawbacks of Django:
        i. Django requires that you follow strict rules. A benefit for some, but you will lose some control over how things are done.
        ii. If an app doesn't require certain features like database access or file management, you may not need Django. It's prewritten code means it is more server intensive, which makes it heavy on low-bandwidth systems.
    c. Advantages and Drawbacks for plain vanilla Python:
        i. Good: Full control over how the app is built, No extra tools or frameworks to learn, Lightweight, making it good for small projects
        ii. Bad: Slower development since everything is built from scratch, more code for common tasks (routing, databases, security, leading to less time for working on features for the user), harder to scale and maintain for larger apps.

2. In your own words, what is the most significant advantage of Model View Template (MVT) architecture over Model View Controller (MVC) architecture?
    a. The most significant advantage of MVT over MVC is that with MVT you don't have to write as much code. With MVC, you must write controller code for how to get data, code for how to display that data, then map data to the URL of the app, then send the data to the user. With MVT, you simply need to specify what data you want to present to the user, and the framework (Template) prepares and sends it. Essentially, in MVC, you need

to write the controller code yourself, whereas in MVT, the Template handles the controller output.

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?
        - How it can help me be a more effective code writer in Python
        - Understand at some point if there are any overlaps with other frameworks
        - Physically practice writing code with it
        - Be able to discuss it in an interview or networking setting
        - Why certain apps and websites use it. And their thought process on choosing it.
    - What do you want to get out of this Achievement?
        - Become a more effective programmer
        - Learn more about the backend of programming in general
        - Know enough to land a job
    - Where or what do you see yourself working on after you complete this Achievement?
        - No idea yet. I have to start this current project first before considering another one. My main focus will be improving all of my projects, my portfolio, my interview skills, etc., to land a job.
        - During that time, I can think about some personal projects with Python and Django that would be helpful for my growth.
        - I will polish and brush up on my knowledge of some of the things I enjoyed most and found most valuable throughout this entire course.
        - I will look to gain some more basic knowledge on alternative frameworks as well in case that happens to be what a company is looking for as I get deeper into the interview process.

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.*)

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

3. 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

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

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.

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.

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?

3. Read Django's documentation on the Django template language 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

1. In your own words, explain Django static files and how Django handles them.

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	
DetailView	

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.

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.

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

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()	
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

1. Consider your favorite website/application (you can also take CareerFoundry). Think about the various data that your favorite website/application collects. Write down how analyzing the collected data could help the website/application.

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

3. In the Exercise, you converted your QuerySet to DataFrame. Now do some research on the advantages and disadvantages of QuerySet and DataFrame, and explain the ways in which DataFrame is better for data processing.

Exercise 2.8: Deploying a Django Project

Learning Goals

- Enhance user experience and look and feel of your web application using CSS and JS
- Deploy your Django web application on a web server
- Curate project deliverables for your portfolio

Reflection Questions

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

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

3. (Optional) Connect with a few Django web developers through LinkedIn or any other network. Ask them for their tips on creating a portfolio to showcase Python programming and Django skills. Think about which tips could help you improve your portfolio.

4. You've now finished Achievement 2 and, with it, the whole course! Take a moment to reflect on your learning:
    a. What went well during this Achievement?
    b. What's something you're proud of?
    c. What was the most challenging aspect of this Achievement?
    d. Did this Achievement meet your expectations? Did it give you the confidence to start working with your new Django skills?

Well done—you've now completed the Learning Journal for the whole course.