Python for Web Developers 

Learning Journal

For: Jordan Nelson

# 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?

The majority of my coding knowledge and practice has come from the courses up to this point with CareerFoundry. I had only ever messed around with some of the free apps on my phone up prior. Those seemed to give a very basic understanding of the skills, and didn’t have any real-world application, so it was all just theoretical. Besides this I also have been into Super Mario World hacking for the Super Nintendo, so I have a very basic understanding of 65c816 Assembly. I would not say I have any proficiency with Assembly, but enough to understand some code and adjust some addresses when needed. I think the biggest help to this course will be what I have learned from CareerFoundry so far, and seeing the parallels between JavaScript and Python I believe will be my biggest help in understanding.

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

Aside from knowing it is a very popular programming language, I wouldn’t say I know much at all. I was told that the syntax is similar-ish to JavaScript, and that it is on the easier side to learn. I have heard about its versatility and seems to be used across a wide range of different applications. To that degree, I would be interested in knowing more about what makes the language so versatile, and how can the skills I learn from this course be used to look into other realms within Python for other potential opportunities.

1. What challenges do you think may come up while you take this course? What will help you face them? Think of specific spaces, people, and times of day of week that might be favorable to your facing challenges and growing. Plan for how to solve challenges that arise.

I honestly think that there will be many challenges as I work through this course. Based on the Full Stack Immersion portion of the course, there were some areas that were a major struggle for me to make sense of. These are mostly present in areas where the material gets very abstracted, and I am not able to follow the logic. Reflecting on how I overcame those challenges, mostly revolved around looking up alternative solutions on the internet. So, seeing how others approached an idea or concept, or looked for other forms of explanation, as sometimes the CareerFoundry materials were not helpful in this regard. I think knowing this will likely be challenging again, and having my prior knowledge of course flow, should prove helpful in overcoming some of these challenges.

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

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?

Based on what I have learned so far, frontend development pertains to everything that a user will visually see in their browser. While backend development has to do with everything that happens behind the scenes, or the things the user does not see. An analogy that comes to mind of a house, where the frontend has to do with the interior design that you would see, and the backend is the foundation and structure of the house. If I were to be hired to work on the backend, I believe there is a wide range of potential operations I might encounter. The first that comes to mind are database management and maintenance, and some other areas that I have worked on including; working with an API, form validations, security measures having to do with authentication and authorization, and integrating with third-party services. These are the first ones that came to mind; however, I know that there are a whole host of more things that backend developers can do, I might just not be as familiar with them yet.

1. 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”)*

To best figure out what language we should use, let’s start by comparing the two and seeing which might be the best option. To start, some of the similarities between Python and JavaScript are; they are both high-level languages, so the code is executed line-by-line, they are both dynamically typed allowing our variables to assume various types, and an ease of learning so using either won’t be a strain on the team. Highlighting some key difference between the two include; Pythons ease of readability making code easier to read and maintain through indentation, Python has out-of-the-box essentials that come pre-installed for common web operations, Python’s clear syntax and dynamic typing together make for less error prone code, and Python has an interactive shell making testing and debugging more efficient.

In summary, for why we should use Python, we need to consider that due to the simplicity and readability, we would be able to rapidly prototype and develop. The wide range of packages available will allow us to easily integrate needed functionality such as simple math operations and complex data processing. Looking down the line, using Python will allow us to have a much easier time with future maintenance through its clear and readable syntax. Additional built-in features, such as those specifically for web development, will help us reduce our development time and needed effort. Lastly, the community support is very strong and will be of benefit to us when we run into potential issues that are holding us up.

1. 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?
   1. I want to gain a deeper understanding of the Python language core concepts. Learning more about how to properly use the basics (functions, loops, etc.) to write code from scratch.
   2. I am excited to get more in the weeds with Django and learn how homing in on the area’s specifics will bolster my web development knowledge.
   3. Mostly, I am wanting to see how I can use the above knowledge to build real-world projects and what all that looks like in real code and how it applies to the full stack.

### 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?

The iPython shell offers a host of benefits to improve the efficiency of our coding compared to the default Python shell. A significant aspect is our ability to read through our code; in the iPython shell, we have syntax highlighting, which helps make our code much more readable. Another quality-of-life improvement is the shell's tab completion and history features. These allow for speeding up coding by offering suggestions for code completion, and the history feature keeps all our past code at quick reference if it needs to be accessed or modified. The shell also offers in-line documentation, which aids in case we need any quick reference material for a particular use case. Lastly, there are built-in debugging and profiling tools that can aid us in our development over the default shell. Overall, the iPython shell offers a host of quality-of-life benefits over the default shell and would make a logical choice for us to use moving forward.

1. 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?** |
| Integer | Represents whole numbers | Scalar |
| Float | Represent decimal numbers | Scalar |
| String | Represents text | Scalar |
| List | A mutable, ordered collection of items | Non-scalar |

1. 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.

There are four key areas that come to mind when thinking about the difference between tuples and lists in Python. The first and most significant one, in my opinion, is mutability, where lists are mutable (can be modified) and tuples are immutable (cannot be modified once created). Secondly, although tuples are immutable, they offer faster performance and a smaller memory footprint, which can be beneficial for large-scale projects. Another way I help to discern the difference in my head is through their use cases. A list would be suitable for collections that might need to be modified, like a shopping list, whereas a tuple would be better for fixed data, like the coordinates of a location on a map. Lastly, the syntax is, of course, different between the two; lists are surrounded by square brackets [ ], and tuples are surrounded by parentheses ( ).

1. 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.

This seems like the most suitable data structure would be a dictionary. Being mutable, the dictionary would allow for modifications to the app, such as adding new flashcards or updating existing ones. An important feature of the dictionary is its ability to use key-value pairs for structuring the data. This allows each word to have a direct association with its definition and category, making the data easy to organize and retrieve. Lastly, the dictionary data structure can accommodate expandability over time. For instance, if a new feature such as synonyms or example sentences were added, the dictionary could easily incorporate this new data.

While lists and tuples are useful, they lack the direct association between a word and its details that dictionaries offer. Lists could be used for storing flashcards sequentially, but they don’t provide the same level of organization as dictionaries. Tuples, being immutable, are less practical for an app where users are expected to add and modify content frequently.

### 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 do you want to travel? “)  available\_destinations = [“Paris”, “Tokyo”, “New York”]  if destination in available\_destinations:  print(f”Enjoy your stay in {destination}!”)  else:  print(“Oops, that destination is not currently available.”) |
|  |

1. 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 are quite straightforward. I think of them as the bridge between different conditions in my code. There are three main ones: and, or, and not.

* **and** is like a strict gatekeeper. It only lets a situation pass as True if both conditions on either side are True. It's perfect when I need everything to line up just right.
* **or** is more of a friendly neighbor. It's satisfied if either condition is True. So, it's my go-to when I’m okay with multiple scenarios.
* **not** is the contrarian. It flips things around. If something is True, not says it's False, and vice versa. It's great for reversing a condition's logic.

I think of them as the building blocks for making decisions in my code. They're good for directing the flow based on various scenarios.

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

Functions in Python are blocks of code that are designed to perform a specific task. They are defined using the def keyword and can receive input in the form of parameters and return a value. Functions are useful for several reasons:

* **Reusability**: They allow code to be written once and used multiple times, reducing redundancy.
* **Organization**: Functions help in organizing code into manageable blocks, making it more readable and maintainable.
* **Modularity**: They allow for dividing a complex problem into smaller, more manageable parts.

Functions are essential for efficient and effective programming, especially as the complexity of the programs increases.

1. 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.

**Goal 1: Understanding Python Core Concepts**

Progress: I have been working to try to solidify the basics into my brain. There are many parallels between what I know from JavaScript and their counterparts in Python, so I’m trying to bolster those connections. I am trying to have my recall be quick with these concepts, as sometimes my brain feels overloaded, and I don’t properly recall some basics quickly. I still need to improve in this regard (about coding in general) to boost my confidence in the field. To that degree, I have been going through the lessons and spending time on each of the practice exercises within.

**Goal 2: Learning Django for Web Development:**

There is not much progress to speak of in this area, as I have not gotten far enough into the lessons to have had any experience with Django. This will change once we get more lessons under our belt and get into that content.

**Goal 3: Building Real-World Projects:**

I have not made much progress in this regard, as I am still getting some of the basic understandings under my belt. I am not sure if I could make something or real-world use in Python just yet. I have been tinkering around with the provided code examples and practice exercises to gain better understanding and confidence. Once I think of a cool project idea that I can work on concurrently, I will start to outline the project, but nothing solid has come to mind yet.

### 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?

File storage is crucial in Python for persisting data beyond the lifecycle of a program. If data isn't stored in files, it would be lost when the program terminates, leading to a lack of data persistence. This is particularly important for applications that require data retrieval after the program has been closed, such as databases, user settings, or any form of data analysis.

1. 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 in Python refer to the process of converting a Python object into a byte stream (serialization) and vice versa (deserialization). This is done using the pickle module, where pickle.dump() serializes an object. Pickling is particularly useful when one needs to store complex Python objects, like lists, dictionaries, or custom classes, and retrieve them later in their original state. However, it's important to note that pickling is Python-specific and should be used carefully due to security concerns when loading pickled data from untrusted sources.

1. 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?

In Python, os.getcwd() is used to find out the current working directory. To change the current working directory, os.chdir(path) is used, where path is the target directory path.

1. 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?

To prevent a script from terminating due to an error, I would use try-except blocks. This allows the script to "try" to execute a block of code and "catch" any exceptions or errors in the "except" block. By handling exceptions, the script can continue running even if an error occurs, or it can gracefully exit or log the error for further investigation.

1. 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 feel that my learning progress is going reasonably well so far. The holiday season has brought a bit of a slowdown, which felt like a necessary break, though I did find myself needing to refresh a few concepts upon returning. What I take pride in most is the quality of my work. I've approached each lesson as if it were a task assigned by an employer, and I believe this mindset has elevated the standard of my output.

However, my challenges aren't exclusive to Python. I sometimes struggle with retaining and recalling certain concepts. It's as if some details don't fully stick, leading me to occasionally draw a blank at the start of a task. This requires a bit of time for me to get the gears turning again. This issue isn't new; I experienced it with previous materials too. Embarking on this career change is indeed daunting!

In essence, my goal is to build confidence in my abilities, ensuring I can bring value to a future employer. To achieve this, I recognize the importance of staying updated on earlier materials and concepts, especially as I encounter more complex topics. Keeping everything fresh in my mind is crucial for my ongoing development in this field.

### Exercise 1.5: Object-Oriented Programming in Python

#### Learning Goals

* Apply object-oriented programming concepts to your Recipe app

#### Reflection Questions

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

**Object-Oriented Programming (OOP)** is a programming approach that uses "objects" to represent both data and the procedures or operations that can be performed on that data. In simpler terms, it's a way of organizing code so that it models real-world entities (like a person, a car, a bank account, etc.) in a more natural and manageable way. In OOP, **objects** are instances of **classes**. A class is like a blueprint for an object, defining the properties (like color, size, or name) and behaviors (like walking, talking, or calculating) that the objects created from it will have.

**The benefits of OOP** include:

* **Modularity:** The source code for a class can be written and maintained independently of the source code for other classes. This makes the system easier to understand and maintain.
* **Reusability:** Once a class is written, it can be reused in other parts of a program or in other programs, reducing the amount of code that needs to be written from scratch.
* **Encapsulation:** This is the bundling of data (attributes) and methods (functions or procedures) that operate on the data into a single unit, or class. This encapsulation helps to prevent accidental interference with the data, making the code more robust and less prone to errors.
* **Inheritance:** This feature allows a new class to inherit properties and methods from an existing class. It facilitates code reuse and can make code more organized and manageable.
* **Polymorphism:** This allows objects of different classes to be treated as objects of a common superclass. For example, if you have a function that expects an object of a superclass, you can pass an object of a subclass to it, and the function will still work.

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

Classes: In Python, a **class** is like a blueprint for creating objects. It defines a set of attributes and methods that the created objects (instances of the class) will have.

Objects: An **object** is an instance of a class. It contains real values instead of the placeholder values defined in the class.

Real-world example: Library

|  |
| --- |
| class Book:  def \_\_init\_\_(self, title, author, isbn):  self.title = title  self.author = author  self.isbn = isbn  def display\_info(self):  return f"Title: {self.title}, Author: {self.author}, ISBN: {self.isbn}"  class Borrower:  def \_\_init\_\_(self, name, member\_id):  self.name = name  self.member\_id = member\_id  self.borrowed\_books = []  def borrow\_book(self, book):  self.borrowed\_books.append(book)  def display\_borrowed\_books(self):  return [book.display\_info() for book in self.borrowed\_books]  # Creating two book objects  book1 = Book("1984", "George Orwell", "123456789")  book2 = Book("To Kill a Mockingbird", "Harper Lee", "987654321")  # Creating a borrower object  borrower1 = Borrower("Alice Smith", "001")  # Borrowing books  borrower1.borrow\_book(book1)  borrower1.borrow\_book(book2)  # Display borrowed books  print(borrower1.display\_borrowed\_books()) |
|  |

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

|  |  |
| --- | --- |
| **Method** | **Description** |
| Inheritance | Inheritance allows a new class to extend an existing class. The new class, known as a subclass, inherits the attributes and methods of the superclass, allowing for reuse of existing code and enhancing the relationship between different classes.  For example, a basic class Animal, and a subclass Dog. Dog inherits characteristics from Animal (like age, weight) and might add specific attributes (like breed). |
| Polymorphism | Polymorphism in OOP refers to the ability of different classes to be treated as instances of the same class through inheritance. It allows the same interface (like a method or a property) to be used for different underlying data types.  For example, if Animal has a method speak(), Dog and Cat subclasses can implement speak() differently, but the method call remains consistent in syntax. |
| Operator Overloading | Operator Overloading allows operators to have different meanings depending on their context. In Python, you can define a method in your class where you can give a custom definition to an operator beyond its predefined operational meaning.  For instance, if you have a Vector class, you can define how the + operator works with vectors, such as adding corresponding elements of two vectors. |

### Exercise 1.6: Connecting to Databases in Python

#### Learning Goals

* Create a MySQL database for your Recipe app

#### Reflection Questions

1. What are databases and what are the advantages of using them?

**Databases** are organized collections of data that are stored and accessed electronically. Database management systems (DBMS) like MySQL manage the data in these databases.

**Advantages of using databases:**

* **Efficient Data Management**: Databases allow for efficient storage, retrieval, and manipulation of data.
* **Data Integrity and Security**: They provide mechanisms to ensure data accuracy and protection from unauthorized access.
* **Data Consistency**: Ensures all users see a consistent view of the data, even when it is being concurrently accessed and modified.
* **Scalability and Performance**: Databases can handle large amounts of data and support multiple users.
* **Advanced Querying and Analysis**: They allow complex queries and analytics, enabling insights and decision-making based on data.

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

|  |  |
| --- | --- |
| **Data type** | **Definition** |
| INT | Represents an integer, a whole number without a fractional part. It's used for numerical data where decimal numbers are not required. |
| VARCHAR | Stands for Variable Character. It's used to store strings (textual data) with a variable length up to a specified limit. It's ideal for storing names, addresses, or any text whose length varies. |
| DATETIME | Used for storing dates and times. This data type is useful for records requiring a date and time stamp, like logs or user activities. |

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

**SQLite** might be a better choice than **MySQL** in the following situations:

* **Embedded Applications**: SQLite is often used in embedded systems and applications like mobile apps, as it's lightweight and requires minimal setup.
* **Small Scale Projects**: For small, local applications with limited concurrent access requirements.
* **Prototyping and Testing**: Due to its simplicity and ease of configuration, it's great for prototyping or for educational purposes.
* **Local Storage for Desktop Applications**: SQLite is a good choice for applications that need a local database without the complexity of a server setup.

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

* **Usage**: JavaScript is primarily used for web development, especially on the client-side, while Python is a general-purpose language used in various fields like web development, data analysis, machine learning, and scripting.
* **Syntax**: Python is more readable and simpler, whereas JavaScript's syntax, while also user-friendly, can be more complex especially for asynchronous operations.
* **Performance**: JavaScript, particularly when running on the Node.js environment, tends to be faster for web-based and network applications due to its non-blocking nature. Python, though versatile, can be slower in execution.
* **Community and Ecosystem**: Both have strong communities, but JavaScript has a larger ecosystem in web development, while Python leads in data science and machine learning.

1. 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?

Regarding the potential limitations of Python, several key areas stand out, including speed, mobile development, memory usage, and runtime errors. Python generally exhibits slower performance compared to languages like C++ and Java. It's also less commonly used in mobile app development, where native languages such as Swift for iOS and Kotlin for Android dominate. Despite Python's flexibility and user-friendliness, it is notable for its higher memory consumption, an important consideration during development. Additionally, as a dynamically typed language, Python may encounter more runtime errors that are only identified during execution.

### Exercise 1.7: Finalizing Your Python Program

#### Learning Goals

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

#### Reflection Questions

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

An Object Relational Mapper (ORM) is a tool that bridges the gap between object-oriented programming languages and relational databases. It translates the language of objects in a programming language, like Python, into SQL queries for the database. Using an ORM offers several advantages:

* **Simplification:** It abstracts the complexity of SQL commands, making database interactions more intuitive and aligned with the programming language's syntax.
* **Efficiency:** It speeds up development time because I don't need to write repetitive SQL code.
* **Code Maintenance:** It makes the code cleaner and easier to maintain, as changes in database structure can often be managed with minimal changes in code.
* **Database Agnosticism:** ORMs often work with different types of databases, so I can switch databases without extensively rewriting my application.

1. 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?

Creating the Recipe app was a rewarding experience. One aspect I did particularly well was implementing the user interface; it was intuitive and user-friendly, which I believe is crucial for any application. If I were to start over, I would focus more on error handling and edge cases. I realized that ensuring robustness against unexpected user inputs or database errors is essential for a reliable application. Improving this aspect would make the app more resilient and professional.

1. 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.

In my recent project, I developed a Recipe app using Python, which leveraged SQLAlchemy for database management. This experience allowed me to integrate Python's powerful programming capabilities with a relational database, using ORM for efficient data handling. I focused on creating a user-friendly command-line interface, ensuring the app was intuitive and accessible. The project honed my skills in Python, particularly in managing data, error handling, and creating interactive applications. It's a testament to my ability to develop functional and user-friendly software solutions.

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

During this Achievement, I found that my understanding of Python's fundamental concepts strengthened significantly. I was able to successfully apply these concepts in a practical project, which was a great confidence booster. The integration of Python with a database using SQLAlchemy went smoother than expected, showing that I could grasp and apply new tools effectively.

* 1. What’s something you’re proud of?

I'm particularly proud of how I managed to translate the requirements into a working application. The Recipe app not only functioned as intended but was also user-friendly. Seeing my code come to life in a functional application was truly rewarding and a testament to the skills I've gained.

* 1. What was the most challenging aspect of this Achievement?

The most challenging part was dealing with unexpected scenarios and errors. Ensuring that the app could handle different kinds of user inputs without crashing required careful planning and testing. It was a valuable learning experience in the importance of robust software design.

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

This Achievement definitely met, if not exceeded, my expectations. It was a practical and hands-on way to apply what I've learned, giving me a much clearer understanding of Python's capabilities. Completing the project has definitely boosted my confidence in my Python skills, and I feel more prepared to tackle more complex projects.

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

For Achievement 2, I want to focus on improving my error handling and testing processes. Learning from the challenges I faced in the first Achievement, I realize the importance of anticipating and planning for potential issues. I also want to experiment more with Python's advanced features to enhance my applications' efficiency and functionality.

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?

Overall, my study routine in Achievement 1 was quite effective. I dedicated regular hours each day to studying, ensuring consistency and gradual progress. However, I realized the importance of integrating more practical exercises alongside theoretical learning. For Achievement 2, I plan to allocate specific time slots for hands-on practice and experimentation with code. This should enhance my understanding and retention of new concepts, keeping in line with my continuous desire to learn and improve.

* 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?

In Achievement 1, I was most proud of developing a user-friendly interface and robust error handling in my Recipe App. These aspects significantly improved the usability and reliability of the application. Moving on to Achievement 2, I aim to further refine these skills. I plan to focus on enhancing user interaction elements and ensuring even more comprehensive error management. Building on this foundation will be crucial for developing more complex applications, ensuring they are both user-centric and resilient.

* 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?

During the last achievement, I faced challenges in optimizing code efficiency and debugging some intricate issues. To overcome these, I actively sought out resources like online forums, coding communities, and documentation. I also spent time reviewing and refactoring my code, which greatly improved my problem-solving skills and understanding of Python. This experience has prepared me for future challenges by teaching me the value of persistence and resourcefulness. In Achievement 2, I feel more equipped to tackle complex problems and am ready to delve deeper into more advanced programming concepts.

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

1. Suppose you’re a web developer in a company and need to decide if you’ll use vanilla (plain) Python for a project, or a framework like Django instead. What are the advantages and drawbacks of each?

**Vanilla Python:**

* + **Advantages**: Complete control over the architecture and design of the application. Flexibility to implement features exactly as needed. Potentially lighter than using a full framework.
  + **Drawbacks**: More time-consuming, as you need to write more code from scratch. Higher chance of bugs and security issues if not carefully managed. Requires more effort to implement features that are standard in frameworks (like user authentication, database management).

**Django:**

* + **Advantages**: Rapid development due to built-in features (admin panel, ORM, authentication). Enforces good coding practices and a clean project structure. Strong community and extensive documentation.
  + **Drawbacks**: Less flexibility in how to implement certain features. Can be overkill for very simple projects. Learning curve to understand and work within the framework.

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

The most significant advantage of the Model-View-Template (MVT) architecture, as used in Django, over the traditional Model-View-Controller (MVC) architecture is its streamlined workflow for web development. In MVT, the controller part is handled by the framework itself, which simplifies the development process. This means developers can focus more on designing models and views/templates, enhancing productivity, especially for complex web applications.

1. 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?

**Deep Dive into Django's Components**: I aim to thoroughly understand Django’s key components, such as models, views, and templates. Understanding these will enable me to efficiently develop web applications and utilize Django’s full potential.

* What do you want to get out of this Achievement?

**Mastering Django's URL Routing and Views**: I'm looking forward to mastering Django's URL routing system and view mechanisms, as they are critical for web app navigation and displaying content.

* Where or what do you see yourself working on after you complete this Achievement?

**Creating an Interactive Web Application**: My ultimate goal is to build an interactive, user-centric web application using Django. This project should not only showcase the technical skills I've acquired but also my creativity in solving real-world problems through web development.

### 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.*)

If I were in an interview and asked to convert a company's website into Django terms, I would start by identifying the different components of the website and mapping them to Django's structure. For instance, let's say the dream company's website has various sections like Home, About Us, Services, Blog, and Contact. In Django, I would consider each of these sections as separate 'apps'. These apps would collectively form the 'project', which in Django is the entire web application.

The 'Home' app might be the landing page, 'About Us' could contain static information about the company, 'Services' could be a dynamic section listing various services offered, 'Blog' would be a regularly updated section with posts and articles, and 'Contact' might have a form for user inquiries. In Django, each of these apps would have their own models, views, and templates, following the MVC architecture. The models define the data structure, views handle the business logic and user requests, and templates are for the presentation layer.

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

To deploy a basic Django application locally, I would first ensure that Python is installed on my system since Django is a Python framework. Then, I would install Django using pip, Python's package manager. Once Django is installed, I would create a new project using the command **django-admin startproject myproject**. This command creates a new Django project with the necessary directory structure.

Inside this project, I would create an app using **python manage.py startapp myapp**. This app is where I would define my models, views, and templates. I would then perform migrations using **python manage.py migrate** to set up the database schemas as defined in my models. After setting up the URLs in the project's **urls.py** and the app's **urls.py**, I would run the local development server using **python manage.py runserver**. This command starts a web server on my local machine where I can see my application in action.

1. Do some research about the Django admin site and write down how you’d use it during your web application development.

The Django admin site is a powerful feature for managing the data in my web application. After creating my models, I would register these models with the Django admin site. This registration allows me to view and manipulate the data of these models directly from a web interface.

To use the admin site, I would first create a superuser using **python manage.py createsuperuser**. This command prompts me to enter a username, email, and password, creating an administrative user for my application. After running my local server, I can access the admin site by visiting /admin on my local web server.

In the admin site, I can add, delete, and edit records of my models. This is extremely useful for tasks like initializing the database with some data, managing user accounts, or moderating content posted by users. It's an essential tool for quickly testing how my application handles data without having to create a separate interface for these admin tasks.

### 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.

Django models are essentially the blueprint of your database. They define the structure of your data, including the types of fields and possibly also their maximum size, default values, selection list options, and more. Think of a Django model as a specific kind of spreadsheet where each model represents a table in your database, and each attribute of the model is a column in that table. For example, if you're creating a web application for a bookstore, you might have a model for Books, where each book has properties like title, author, and price.

The beauty of Django models lies in their abstraction. You define your data models in Python code, and Django takes care of the heavy lifting of generating the appropriate database tables, queries, and commands. This approach has several benefits:

* **DRY Principle**: You define your data structure once and use it everywhere, reducing redundancy and mistakes.
* **Migration System**: Django automatically generates database migration scripts for you. When you modify your models, Django helps you update your database schema without losing data.
* **ORM (Object-Relational Mapping)**: This allows you to interact with your database using Python code instead of writing raw SQL queries, making your code more readable, maintainable, and secure.
* **Admin Interface**: Django automatically generates a powerful admin interface for your models, making it easy to view, add, delete, and edit records.

1. 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.

Writing test cases from the beginning of a project is crucial for several reasons, and doing so can significantly impact the development process and the quality of the final product. Let's consider an example project: developing a new social media platform.

In such a project, features might include user registration, posting updates, and following other users. Writing tests early for these functionalities ensures that:

* **Early Bug Detection**: Test cases can help catch bugs and issues at an early stage, before they become complex to solve. For instance, a test case for user registration might uncover issues with email validation or password storage practices.
* **Code Quality**: Regular testing encourages cleaner, more modular code, as functionalities need to be isolated for effective testing. This separation makes the codebase more manageable and easier to understand.
* **Refactoring Confidence**: With a comprehensive test suite, developers can make changes or refactor code with confidence, knowing that tests will catch any inadvertent errors introduced by their changes.
* **Documentation**: Tests act as a form of documentation, showing how the system is supposed to work. New developers on the project can look at the tests to understand the functionality and requirements of different parts of the application.
* **Agile Development**: In agile development, features are developed incrementally, and tests ensure that new changes don't break existing functionalities. This allows for continuous integration and delivery without compromising the stability of the application.

By integrating testing into the development process from the start, you ensure that your project is built on a foundation of quality and reliability, making it easier to maintain and scale over time.

### Exercise 2.4: Django Views and Templates

#### Learning Goals

* Summarize the process of creating views, templates, and URLs
* Explain how the “V” and “T” parts of MVT architecture work
* Create a frontend page for your web application

#### Reflection Questions

1. Do some research on Django views. In your own words, use an example to explain how Django views work.

Django views are essentially the middleman between the user's request and the final webpage they see. When a user requests a page, the view decides what data to fetch from the model (if needed) and which template to use for rendering that page. For instance, imagine a blog application where a user wants to see a specific post. The user's request hits a view, which then retrieves the post's details from the database (model) and passes this information to a template. The template formats this data into HTML, which is then displayed to the user as the blog post.

1. Imagine you’re working on a Django web development project, and you anticipate that you’ll have to reuse lots of code in various parts of the project. In this scenario, will you use Django function-based views or class-based views, and why?

In a project where reusability of code is key, I would lean towards using class-based views (CBVs). The reason is that CBVs are designed to encourage reusing code by extending classes. They provide a structured way to define views, where common patterns can be abstracted into base classes. This makes it easier to maintain and extend the functionality as the project grows. For example, if multiple views share common functionalities like pagination or form processing, you can create a base class that implements these features and then extend this class in your specific views. This approach reduces duplication and makes the code cleaner and more modular.

1. 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.

The Django Template Language (DTL) is a powerful tool for generating HTML dynamically. It allows you to incorporate Python-like logic into your HTML templates without writing actual Python code within the template files. Here are a few basics:

* **Variables**: You can pass variables from views to templates and use double curly braces to display them, like {{ variable\_name }}.
* **Tags**: These are more complex constructs enclosed in {% %} that perform looping, conditionals, and more. For example, {% for item in list %} loops over items in a list.
* **Filters**: Filters modify the display of variables, using a pipe symbol (|). For example, {{ name|lower }} would display the name variable in lowercase.
* **Inheritance**: Templates can inherit from one another, allowing you to create a base "skeleton" template that contains common elements like headers and footers. You can then extend this base template in other templates.

These basics of Django's views, class-based views, and template language are crucial for building and maintaining web applications effectively.

### 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 |  |

1. 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](https://docs.djangoproject.com/en/3.2/ref/models/querysets/). 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:
   1. What went well during this Achievement?
   2. What’s something you’re proud of?
   3. What was the most challenging aspect of this Achievement?
   4. 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.