Python for Web Developers 

Learning Journal

# Objective

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

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

## Directions

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

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

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

## Pre-Work: Before You Start the Course

Reflection questions (to complete before your first mentor call)

1. What experiences have you had with coding and/or programming so far? What other experiences (programming-related or not) have you had that may help you as you progress through this course? I’ve completed some self-study in python over 2 years ago and got to a point where I use python occasionally at work to help automate some of the more boring or monotonous tasks.
2. What do you know about Python already? What do you want to know? I know how to make basic scripts to help automate tasks. I would like to know how to make full applications and how I can use it to further a career in web development.
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. Most likely after going through this first exercise, the biggest challenge will be the lack of actual information provided in the exercises. The instructions for this first exercise might as well of said “good luck, remember google is your friend” as the actual reading didn’t go over most of what was asked in the exercise. I will make sure I have plenty of time to look up all of this information so that I don’t fall behind in the course.

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? Frontend web development involves creating the user interface and visual elements of a website, while backend development focuses on server-side programming and infrastructure. As a backend developer, you would work on operations such as handling user input, managing databases, implementing business logic, configuring servers, optimizing performance, and ensuring security. Backend development is responsible for the functionality and behind-the-scenes operations of a web application, while frontend development focuses on the user-facing aspects.
2. Imagine you’re working as a full-stack developer in the near future. Your team is asking for your advice on whether to use JavaScript or Python for a project, and you think Python would be the better choice. How would you explain the similarities and differences between the two languages to your team? Drawing from what you learned in this Exercise, what reasons would you give to convince your team that Python is the better option? *(Hint: refer to the Exercise section “The Benefits of Developing with Python”)* Python is a recommended choice over JavaScript for several reasons. Python prioritizes readability with its clean syntax, making it easy to understand and maintain. It also offers built-in package management through pip, simplifying the integration of external resources. Python's comprehensive standard library provides essential tools, saving development time. Its simplicity and strong community support result in an efficient development flow, while the active community ensures access to abundant documentation and resources. Overall, Python's readability, ease of use, package management, essential features, development efficiency, and community support make it the better option.
3. Now that you’ve had an introduction to Python, write down 3 goals you have for yourself and your learning during this Achievement. You can reflect on the following questions if it helps you. What do you want to learn about Python? What do you want to get out of this Achievement? Where or what do you see yourself working on after you complete this Achievement? I’m hoping to get a couple of different things out of this achievement which I have broken up into the 2 following categories: **Automation and Scripting**: Python is renowned for its automation and scripting capabilities. I want to leverage Python to automate repetitive tasks, build scripts for data processing, or create tools that enhance my front-end development workflow. **Career Flexibility and Demand**: Python is one of the most popular and in-demand programming languages. By learning Python, I can broaden my skill set and increase my versatility as a developer. Hopefully allowing me to more easily transition to different roles or work on more diverse projects that require Python, opening up more opportunities and ensuring long-term career growth.

### 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 several advantages over Python's default shell. It provides features such as syntax highlighting, making code more visually distinct and readable. IPython is designed to be more user-friendly, with improved tab-completion, command history, and helpful shortcuts. Automatic indentation saves time by automatically indenting code blocks. IPython shell offers the advantage of rapid and effortless testing of small code snippets. Each command is executed instantly upon entering, and the resulting output is promptly displayed. This quick and convenient approach proves to be significantly faster and more convenient for testing small code fragments compared to the time-consuming task of creating and executing separate script files.
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?** |
| Integer(int) | represent whole numbers without any fractional parts. | scalar |
| Floating-Point Num (float) | represent numbers with fractional parts (has decimals) | scalar |
| String (str) | are sequences of characters enclosed in quotation marks (either single or double) | non-scalar |
| Boolean (bool) | represent truth values and can have two possible states: True or False. | 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. Lists and tuples are ways to store groups of things in Python. The main difference is that lists can change (you can add or remove things from them), while tuples cannot be changed once you make them. To make a list, you use square brackets like [1, 2, 3], and for a tuple, you use parentheses like (1, 2, 3). Lists are good when you want to keep adding or changing things, while tuples are good when you want to keep things the same. Tuples are a bit faster and take up less space in your computer's memory.
2. In the task for this Exercise, you decided what you thought was the most suitable data structure for storing all the information for a recipe. Now, imagine you’re creating a language-learning app that helps users memorize vocabulary through flashcards. Users can input vocabulary words, definitions, and their category (noun, verb, etc.) into the flashcards. They can then quiz themselves by flipping through the flashcards. Think about the necessary data types and what would be the most suitable data structure for this language-learning app. Between tuples, lists, and dictionaries, which would you choose? Think about their respective advantages and limitations, and where flexibility might be useful if you were to continue developing the language-learning app beyond vocabulary memorization. For the described scenario, I would choose to use a dictionary for the following reasons:  
     
   Dictionaries in Python provide key-value pairing, which makes them suitable for storing and accessing data with associated attributes. In this case, each flashcard could have vocabulary words as keys and their corresponding definition and category as values.

Using dictionaries would allow for easy and efficient retrieval of the flashcard information. You can quickly access the definition and category of a word by providing its key, making it convenient for implementing the quiz functionality.

Additionally, dictionaries provide flexibility in terms of modifying or updating flashcard information. You can easily change the definition or category associated with a particular word by updating the corresponding value.

### 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!)*

|  |
| --- |
| def travel\_destination():  destination = input("Where do you want to travel? ")  if destination == "Paris":  print("Enjoy your stay in Paris!")  elif destination == "London":  print("Enjoy your stay in London!")  elif destination == "New York":  print("Enjoy your stay in New York!")  else:  print("Oops, that destination is not currently available.")  travel\_destination() |

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 used to manipulate boolean values, which can be either True or False. Python provides three main logical operators: 'and', 'or', and 'not'. The 'and' operator returns True only if both operands are True; otherwise, it returns False. On the other hand, the 'or' operator returns True if at least one of the operands is True; otherwise, it returns False. Lastly, the 'not' operator is used to flip the value of a condition. If the condition is true, it becomes false. If the condition is false, it becomes true. These operators are commonly used in conditional statements and loops to control the program flow based on the truth value of expressions.
2. What are functions in Python? When and why are they useful? Functions in Python, like JavaScript and other languages, are essentially just reusable blocks of code to perform specific tasks. They are useful for organizing code, improving readability, promoting reusability, and reducing redundancy of code by making the code more modular.
3. 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. In all honesty, so far this has mostly been a refresher of things I’ve already learned about python. We’re still at the super basic level of Python. I’m looking forward to the part where we get into reading from files and parsing data. That will be super useful for my current role which has a lot of mundane tasks like that. Also, I’m hoping to learn more about how Python helps in web development. This hasn’t been touched on yet. I’m assuming this is mostly backend stuff but am super excited to learn how knowing Python can help me get my foot in the door in a new industry and career. The refresher has been good though. It’s nice to see how similar different languages can be to one another. I’m seeing a lot of similarities between Python and JavaScript.

### 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 important in Python for saving and retrieving data. Without it, you wouldn't be able to persistently store information, leading to data loss after a program terminates. File storage allows you to maintain data between sessions and share information with other programs.
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? In Python, pickles are a module that allows objects to be serialized and deserialized, converting them into a byte stream that can be stored or transmitted. Pickling is useful when you want to preserve the state of complex objects or large data structures, for later use. It enables easy storage and retrieval of objects, simplifies data sharing between different Python programs, and facilitates communication between processes.
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? In Python, you can use the **os.getcwd()** function to find out the current working directory. It returns a string representing the current directory path. To change the current working directory, you can use the **os.chdir(path**) function, where path is the desired directory path.
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? To prevent the entire script from terminating due to an error in a specific block of code, you can use exception handling in Python. By wrapping the potentially problematic code within a **try** block, and include an **except** block or multiple **except** blocks to handle any specific exceptions that may occur. This way, if an exception is raised within the try block, the script will continue executing from the except block onwards, allowing you to handle the error gracefully and continue with the rest of the script's execution.
5. You’re now more than halfway through Achievement 1! Take a moment to reflect on your learning in the course so far. How is it going? What’s something you’re proud of so far? Is there something you’re struggling with? What do you need more practice with? Feel free to use these notes to guide your next mentor call. I feel like this last exercise was a struggle. I managed to work my way through it, but I’m a long way off from doing this without googling everything. Programming can be so backwards sometimes and I’m still struggling with it a little. For example, I had a rough time with the take\_recipe() function I this one because you had to include the difficulty in it’s return which is calculated in a function outside of this function, so just figuring out how to get that to work was a struggle and not immediately obvious.

### 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 paradigm that organizes code around objects, which are instances of classes. Its benefits include improved code organization, easier maintenance and debugging, enhanced code reusability, and increased productivity through inheritance and polymorphism.
2. What are objects and classes in Python? Come up with a real-world example to illustrate how objects and classes work. In Python, objects are instances of classes, which act as blueprints or templates. Think of a car manufacturing company where cars are produced based on a design blueprint. The blueprint represents the class, and each car produced is an object. The class defines attributes like color, model, and speed, as well as methods such as accelerating and braking. When a new car is manufactured, it becomes an object of the class, inheriting the blueprint's characteristics and behaviors. Multiple cars can be created from the same class, each with unique attribute values. In Python, you can create a class called "Car" with attributes and methods, and then create objects (cars) with specific attribute values. Objects can use the methods defined in the class, just like real-world cars following a design blueprint to perform actions.
3. In your own words, write brief explanations of the following OOP concepts; 100 to 200 words per method is fine.

|  |  |
| --- | --- |
| **Method** | **Description** |
| Inheritance | In Python, inheritance is like a family relationship between classes. Imagine you have a parent class, which is like a wise and experienced teacher. Then, you have a child class, who is like a student learning from the teacher. The child class can inherit the knowledge and skills (attributes and methods) of the parent class. It can also add its own unique knowledge and skills. This helps to avoid repeating the same things in multiple classes and makes it easier to create specialized classes based on more general ones. Inheritance is like passing down knowledge and abilities from one class to another, just like a family passing down traits and skills from one generation to the next. |
| Polymorphism | In Python, polymorphism is when multiple objects of the same class all have the same methods, but they can behave differently. Sort of like the ability of superheroes to perform different actions with the same method name based on their individual powers. For example, imagine a "Superhero" class with a method called "attack." Different superheroes, like "Superman" and "Spider-Man," can inherit from this class and override the "attack" method to perform their unique attacks, such as "Superman" using laser vision and "Spider-Man" using web-slinging. Despite having the same method name, each superhero's attack behaves differently. Polymorphism allows objects of different classes to be treated as interchangeable, enabling flexibility and dynamic behavior based on their specific implementations. |
| Operator Overloading | Operator overloading in Python is the ability to define how operators behave with user-defined classes. It allows customizing the behavior of operators such as +, -, \*, /, and others for objects of a class. For example, by overloading the + operator, you can define how two objects should be added together. This enables more intuitive and meaningful operations with custom classes, making them behave like built-in types. By implementing special methods like add or mul, you can define the desired behavior of operators when applied to instances of your class, extending the language's capabilities to suit specific object interactions. |

### 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 makes accessing and managing data more efficient. Using them in coding like Python enhances data integrity, consistency, security, and enables complex searching/querying, thereby simplifying data handling and promoting scalability in applications. Put simpy, databases are like digital filing cabinets for data. They keep data safe, easy to find, and easy to change. Using them in Python makes handling a lot of data easier and safer.
2. List 3 data types that can be used in MySQL and describe them briefly:

|  |  |
| --- | --- |
| **Data type** | **Definition** |
| INT | A whole number data type. This can be used to store integers, for instance, the age of a person or the number of items in a stock. |
| VARCHAR | A variable-length character string data type. This is typically used to store text information, like names or addresses. The length parameter indicates the maximum number of characters it can hold. |
| DATETIME | A data type for storing date and time values. This is useful for recording events, like when a record was created or updated. |

1. In what situations would SQLite be a better choice than MySQL? SQLite is best when you need a lightweight, file-based database for small-scale applications, local/single-user storage, or for development/testing. It doesn't require a server setup
2. Think back to what you learned in the Immersion course. What do you think about the differences between JavaScript and Python as programming languages? Python is a programming language that's easy to learn and use. It's great for making small programs quickly because it has a simple and clean way of writing code. It's like a straightforward tool for small tasks. On the other hand, JavaScript is mainly used for making websites more interactive and dynamic. It's like a powerful tool for making things happen on websites. So, Python is good for small stuff, while JavaScript is good for web stuff. To be a bit more technical, I could mention that Python employs indentation for blocks, while JavaScript uses braces and Python is an interpreted language while JavaScript is both interpreted and compiled, making it faster.
3. 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?  
     
   1. Speed: Python is interpreted, not compiled

2. Memory Use: Due to being so dynamic, Python’s data types can use up more memory than languages with statically typed variables

3. Dependencies: Python appears to rely heavily on it’s vast ecosystem of third-party libraries, I can imagine that staying up to date with these can become a challenge

4. GUI: We haven’t touched on GUI yet, but from what I’ve seen support for GUI development seems to be a bit limited.

### 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 like a translator between a database and a programming language like Python. It helps you work with databases in a more convenient way by treating database tables as objects. It makes your code simpler and easier to understand, and also protects against security issues. Overall, it makes working with databases in Python much easier and safer.
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? The process of building the app went fine. There wasn’t really a ton of time to get very creative here. These assignments took a lot longer than the suggested time, but if I had to redo it again I would probably spend more time on the user experience of the app. There are certain things I added such as displaying 2 separate lists of results from the search option, one that shows anything that has at least one ingredient and another that only shows recipes that have all ingredients that were search. It would be nice to add a cancel function to each option, say if you selected the wrong option, etc..
3. Imagine you’re at a job interview. You’re asked what experience you have creating an app using Python. Taking your work for this Achievement as an example, draft how you would respond to this question. I have previous experience in creating Python applications that involve database interaction and CRUD operations. In particular, I have worked with the SQLAlchemy library to manage databases. I have developed applications that allow users to create, read, update, and delete data entries. These applications follow an object-oriented approach, utilizing classes and methods to handle various operations on the data. Input validation and data integrity are key considerations in these applications to ensure accurate and reliable data handling. Overall, my experience with Python includes developing database-driven applications with a focus on functionality, efficiency, and data management.

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? Due to some limited previous Python experience I was able to navigate this achievement fairly swiftly and maintain a pace of about 2 exercises per week.
   2. What’s something you’re proud of? I am proud of the fact that my knowledge of basic python scripting has expanded somewhat. This is extremely useful for anyone that works an office job that has a lot of mundane time-consuming tasks.
   3. What was the most challenging aspect of this Achievement? Honestly, it was all the depreciated code, and lack of thorough explanations in the reading. This course moved at such a pace that if I didn’t have previous python experience, I bet I would’ve struggled mightily. From a more technical side, the SQL app was the hardest. Learning all the various SQL keywords and incorporating that into the Python app was a challenge.
   4. Did this Achievement meet your expectations? Did it give you the confidence to start working with your new Python skills? Overall, the course met expectations. The outdated code and confusing reading was a little disappointing at times, but for the most part this was a nice refresher on some Python skills I had forgotten and I even learned a few new things along the way. Confidence isn’t something I’ve gotten a lot of from Career Foundry courses. They move at such a pace that you just have to keep moving. You don’t really have any extra time to sit and practice more of what you just learned. By the time you finish one exercise you typically have to start the next and by the end of the achievement, you’re struggling to remember concepts from few weeks ago. If you move on to other achievements, it gets even worse. I find myself wondering what I remember from the React and JavaScript courses I just had a few months ago. I haven’t had the opportunity to go back and gain more experience with them.
   5. What’s something you want to keep in mind to help you do your best in Achievement 2? As a relatively new programmer it's important to keep a few key things in mind. First, Python emphasizes readability, meaning indentation is super important, so strive for clean and well-structured code. Second, I need to remember to take advantage of Python's extensive library, which offers numerous pre-built modules and functions.

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

### Pre-Work: Before You Start Achievement 2

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

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

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

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

### Exercise 2.1: Getting Started with Django

Learning Goals

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

#### Reflection Questions

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? Using vanilla (plain) Python means you have complete control over your project and can make it exactly how you want. However, it requires more work because you have to write everything from scratch and might miss out on useful features that frameworks provide. On the other hand, using a framework like Django gives you pre-built tools and features that make development faster and easier. You don't have to reinvent the wheel, but you might have less flexibility to customize things exactly the way you want.
2. 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 over the Model View Controller (MVC) architecture is the clear separation between the presentation logic and business logic. In MVT, the template handles the presentation layer, while the model handles the business logic. This separation allows for better code organization, maintainability, and reusability. Additionally, MVT simplifies the development process by eliminating the need for explicit controller components, resulting in cleaner and more concise code.
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?
* What do you want to get out of this Achievement?
* Where or what do you see yourself working on after you complete this Achievement? My three goals for learning Django are as follows: 1) understanding Django's features for URL routing, views, forms, validation, and authentication, 2) gaining the ability to create visually appealing applications using Django's template system, and 3) being able to independently develop a Python project with Django, such as building a database to track watched movies or TV shows.

### 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.*) – Well, my dream company would be Walt Disney Studios Animation, so looking at <https://disneyanimation.com> I would start by identifying the major components and functionalities of the website. This would include breaking down the website into Django models, views, and templates. For example, the homepage could be a Django template that displays various sections and featured content. Each section, such as Careers, What We Do, Life at Disney, and Events, could be Django models with corresponding views and templates. I would also consider implementing user authentication and registration using Django’s built-in authentication system. Lastly, I would leverage Django’s URL routing to map different URLs to appropriate views and templates.

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, start by setting up a virtual environment to isolate dependencies. Then, use pip to install Django and other required packages. Create a new Django project using the "django-admin startproject" command and configure the database settings in the project's settings.py file. Next, create Django apps within the project using the "python manage.py startapp" command, and define models, views, and templates within the apps (I believe this will be covered soon). Run database migrations with "python manage.py migrate" and start the development server using "python manage.py runserver". Finally, access the application locally through the provided localhost address and port number.
2. 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 tool that can be used during web application development. It provides an easy-to-use interface for managing the application's data and performing administrative tasks. Developers can quickly create, update, and delete database records through the admin site without writing custom views. It automatically generates forms based on the models, simplifying the process of managing data. Additionally, developers can define custom admin actions, permissions, and filters to customize the admin site's functionality based on specific project requirements, making it a valuable tool for efficient 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. Django models are Python classes that represent database tables. They define the structure of the data, including fields and relationships. Models handle the creation, retrieval, updating, and deletion of records (CRUD) in the database. The benefits of Django models include automatic schema generation, database abstraction, built-in validation, and support for complex queries. They simplify database operations, promote code organization, and enhance code reusability, making it easier to develop and maintain data-driven web applications.
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. Writing test cases from the beginning of a project is crucial for several reasons. Firstly, it helps ensure the correctness and stability of the codebase by identifying and preventing bugs early on. For example, in a banking application, test cases can verify that deposits, withdrawals, and transfers are accurately reflected in account balances. Secondly, tests act as documentation, providing clear usage examples and ensuring future modifications don't break existing functionality. Lastly, early test coverage facilitates continuous integration and deployment, enabling efficient development iterations and maintaining a high-quality codebase.

### 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. In Django, views handle the logic behind processing user requests and generating responses. They receive incoming HTTP requests and return corresponding HTTP responses. For example, consider a simple Django view for a blog post. When a user accesses a specific URL for a blog post, the view function retrieves the post from the database, processes any user input, and generates an HTML response containing the blog post's content. This response is then sent back to the user's browser.
2. Imagine you’re working on a Django web development project, and you anticipate that you’ll have to reuse lots of code in various parts of the project. In this scenario, will you use Django function-based views or class-based views, and why? In this scenario, I would choose class-based views in Django. Class-based views provide a more organized and modular approach for code reuse. They allow us to define common functionality in base classes and inherit from them in different views. This promotes code reusability, reduces duplication, and improves maintainability. Additionally, class-based views offer built-in features like mixins and decorators, which further enhance code organization and flexibility compared to function-based views.
3. 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.
4. Syntax: The Django template language uses curly braces ({}) and percent signs (%) to enclose its tags and variables, respectively. For example, {{ variable\_name }} represents a variable, while {% tag\_name %} represents a template tag.
5. Variables: Variables in templates are placeholders that display dynamic content. They are surrounded by double curly braces. For example, {{ article.title }} would display the title of an article object.
6. Tags: Template tags control the logic and flow within templates. They are surrounded by percent signs. Tags can perform various actions, such as loops, conditional statements, and including other templates.
7. Filters: Filters modify the output of template variables. They are appended to a variable using the pipe symbol (|). Filters can format data, truncate strings, apply date formats, and more. For example, {{ variable\_name|filter\_name }} applies a filter to the variable.
8. Comments: Comments in Django templates can be added using {# comment text #}. They are useful for adding notes or disabling certain sections of code without affecting the template rendering

### 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

#### Frontend Inspirations

1. Take inspiration from existing recipe applications. Create a section called “frontend inspirations” in your “Journal.doc” document and add links to your favorite recipe applications (three at most). Take screenshots and write a few lines on what you like most about them.
   1. Yummly – I really like how simple and clean the recipe details pages are. This can easily be altered to show just the difficulty level instead of calories. A description section can also be added.

A screenshot of a food recipe

Description automatically generated

* 1. Whisk – I really like the homepage look. It’s nice and simple and I can see a login button working really well here. Also the recipe detail views are laid out nicely and can be altered to show just the description, ingredients, and difficulty level.

A close up of a logo

Description automatically generated

A screenshot of a recipe

Description automatically generated

#### Reflection Questions

1. In your own words, explain Django static files and how Django handles them. Django static files are the static assets, such as CSS, JavaScript, and images, used by a Django web application. Django provides a built-in mechanism to handle static files efficiently. It uses the STATIC\_URL setting to define the URL prefix for serving static files. During development, Django's development server automatically serves static files from the STATICFILES\_DIRS specified in settings.
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 | is a generic view that displays a list of objects retrieved from a database. It handles the rendering of a template and provides pagination support. |
| DetailView | is another generic view that retrieves a single object from the database and displays it in a template. It is commonly used for displaying detailed information about a specific object. |

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. I feel as if the reading material is becoming less and less helpful. In this last exercise for example, the reading material picks up from a spot in the bookstore app that we hadn’t even completed. The last exercise did not tell us to perform the steps that this exercises reading said that we accomplished previously. I had to spend over an hour trying to figure out what was missing and add it. There are other areas where the reading asks you to test something that will fail without first doing something that isn’t explained until later on in the exercise. I just feel like the reading isn’t very helpful and I’ve had to supplement this course with a lot of online research and even additional online classes to figure some of this out that hasn’t been explained very well in this reading. As far as things I need more practice with, at this point in my development that amounts to literally everything as far as python goes, I would say I should do this whole project a couple more times before I would feel comfortable enough in Django to where I could answer interview questions about the process.

### 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. Incorporating authentication into a Django application is crucial for ensuring secure access and protecting user data. Authentication allows the application to verify the identity of users, preventing unauthorized access to sensitive information. For example, in a banking application, authentication ensures that only verified users can access their accounts, protecting their financial data from potential breaches. Implementing authentication in Django provides better user management features, such as password hashing, session management, and role-based permissions, enhancing overall security and user experience.
2. In your own words, explain the steps you should take to create a login for your Django web application. Simply put the steps for this are:
   1. Create the view: Define a view function in Django that handles the login logic, such as user authentication and session management.
   2. Create the template: Design an HTML template that includes the login form. This form should collect user credentials like username and password.
   3. Specify the URL mapping: In your application's urls.py file, define a URL pattern that maps to the login view function created in step This URL pattern should handle the submission of the login form.
   4. Register the URL to the project: In the main urls.py file of your Django project, include the URL pattern defined in step C, linking it to the corresponding application's urls.py file.
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() | Is a Django function used for authenticating a user. It takes in a set of user credentials, such as username and password, and verifies them against the authentication backend. If the credentials are valid, it returns a user object representing the authenticated user; otherwise, it returns None. |
| redirect() | This function is used to perform a redirect in Django. It takes a URL as a parameter and returns a redirect response to the specified URL. When a user visits a URL that triggers a redirect, they are automatically redirected to the new URL provided. |
| include() | Is a function used in Django's URL configuration. It allows you to include other URL configurations from other modules or files into your main project's URL configuration. This is useful for organizing and modularizing your URL patterns, making it easier to manage and maintain your Django project's URLs. |

### Exercise 2.7: Data Analysis and Visualization in Django

#### Learning Goals

* Work on elements of two-way communication like creating forms and buttons
* Implement search and visualization (reports/charts) features
* Use QuerySet API, DataFrames (with pandas), and plotting libraries (with matplotlib)

#### Reflection Questions

1. Consider your favorite website/application (you can also take CareerFoundry). Think about the various data that your favorite website/application collects. Write down how analyzing the collected data could help the website/application. Analyzing data collected from my favorite website/application (CareerFoundry, for example) could provide valuable insights. It could reveal user behavior patterns, such as popular courses or navigation preferences, enabling personalized recommendations. Additionally, data analysis could identify areas for improvement, optimize the user interface, and enhance overall user experience. Furthermore, it could aid in assessing the effectiveness of marketing strategies, content engagement, and student success rates, ultimately leading to better-tailored content and features to meet user needs.
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.

In the Django QuerySet API, a QuerySet can be evaluated in several ways, here are a few:

1. Iteration: Using a for loop to access each object in the QuerySet.
2. Slicing: Applying Python list slicing to retrieve a specific range of objects.
3. len(): Getting the length of the QuerySet.
4. exists(): Checking if any objects exist in the QuerySet.
5. bool(): Evaluating the QuerySet as a boolean, returning True if it contains objects and False if it's empty.
6. get(): Retrieving a single object matching the query (raises an exception if multiple or none found).
7. 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. QuerySet and DataFrame are data structures used for data processing, each with their own advantages and disadvantages. QuerySet, in Django, is designed for efficient database queries and integrates well with Django's ORM. It allows lazy evaluation, executing queries only when needed, optimizing performance.

On the other hand, DataFrame, part of the Pandas library, excels in data manipulation and analysis. It supports various data sources and offers a wide range of functions for tasks like grouping, merging, and data cleaning. DataFrame's versatility makes it ideal for working with tabular data from different sources, not just databases.

The advantage of DataFrame over QuerySet lies in its comprehensive capabilities and performance. With its optimized functions, DataFrame simplifies complex data processing tasks and provides a unified structure for data manipulation. It is highly efficient for data exploration, transformation, and analysis, making it a powerful tool in the realm of data science and analytics. However, QuerySet remains valuable within Django applications, ensuring seamless database interactions and leveraging Django's built-in features for web development.

### 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.