CAREERFOUNDRY

Python for Web Developers Learning Journal

Objective

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

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

Directions

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

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

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

Pre-Work: Before You Start the Course

Reflection questions (to complete before your first mentor call)

1. What experiences have you had with coding and/or programming so far? What other experiences (programming-related or not) have you had that may help you as you progress through this course?

I worked at a software engineer for about 5 years and in that time I worked primarily with Java, JSP, VXML and HTML in my work. My first job was creating self-service interactive voice response applications that were used in a variety of industries provide self-service to their customers through voice recognition or DTMF (touch tone) input dialogs and menus. My second job was working on a clothing retailer's website and backend business processes. I was responsible for making weekly updates to the website and corrected errors in internal reporting tools.

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

   I know you can use Python for backend scripting like JavaScript. The syntax is more intuitive and allows for easier implementation of common application needs through its libraries and frameworks. I would like to know first how to code in Python, but also what the difference is when I compare the JavaScript implementation of something like an API fetch that performs a CRUD operation vs how Python handles it. I am curious for that "side-by-side" comparison of how the code compares to one another.

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.

   I have encountered many challenges throughout the Full-Stack Immersion course from unclear/outdated instructions to version incompatibilities with frameworks and dependencies. I expect to encounter challenges because that is the nature of the tech field. With constant change and rapid advancement, there is no way that problems will not be encountered at some point. I have been quite impressed with the help that AI can provide with some of these issues. In my previous IT work, we never had the luxury of AI and had to do all the legwork and research ourselves or ask someone who was more experienced with the technology in use. I have found that I can often get more detailed information from asking the AI questions about errors I receive or why certain issues might arise given my dev environment setup. It doesn't always give me a solution to my problem, but I find I can start with an idea it suggested and try to work down a path looking for the solution. I also reached out to my tutor/mentor if things still were not resolving with my own research and trials.

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

Exercise 1.1: Getting Started with Python

Learning Goals

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

Reflection Questions

1. In your own words, what is the difference between frontend and backend web development? If you were hired to work on backend programming for a web application, what kinds of operations would you be working on?

   I would be working on things like the API functionality (internal or external), routing, authentication, managing database connections and queries, processing data, validating data, data security and of course, documentation for the backend.

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 and JavaScript are similar in that they are both a frontend & backend scripting language, they both have dynamic typing for variables, both support OOP, they can import external packages to add functionality using package managers and have strong developer communities. Since Python lends itself more easily to backend development, if the project is doing some heavy data processing or data fetching it would be easier and faster with Python. It also has the convenience of the virtual environments where you can configure and run different scripts with different requirements by simply switching to a different virtual environment that has the needed dependencies. I'm not sure I could make a strong case for frontend at this point in learning about Python but from what I understand the HTML pages could be generated using Django on the backend and then returned to the browser. But browsers don't readily support Python.

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?

   - Learn the basics of backend scripting with Python and the parallels with JavaScript.

- Learn how to create a server and manage database connections with Python.
- I would like to rewrite my portfolio email backend server project with Python. I created a Node.js / Express.js server with Resend to create and send emails with the data posted from my contact form. I think it would be good practice to be able to write this same server using Python.

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 provides the same functionality that the default shell does but extends it adding extra features to make the UI more friendly and easier to use. It adds indentation for code, line numbers, color coding syntax, helps with errors and debugging and handles sessions.

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?
str	A string, which is an immutable array of characters, each of which can be indexed and are also each of str data type. In spite of being an array of characters, the str value is considered a single, atomic value and therefore is scalar.	Scalar
int	An integer, being an immutable numeric value that is a positive or negative whole number, or zero.	Scalar
Tuple	A sequential, immutable array that stores multiple values within. The stored values can be of any type and can be mixed within the tuple.	Non-Scalar
Dictionary	An unordered, mutable structure that stores	Non-Scalar

	information within key-value pairs.	

3. A frequent question at job interviews for Python developers is: what is the difference between lists and tuples in Python? Write down how you would respond.

   Tuples and lists are very similar in physical structure, both being linear arrays that can store values. Both can store values of different data types and support indexing and slicing. But the main difference is that tuples are immutable and cannot be modified whereas the list is mutable and allows adding, removing, modifying or sorting the elements in the list.

4. In the task for this Exercise, you decided what you thought was the most suitable data structure for storing all the information for a recipe. Now, imagine you're creating a language-learning app that helps users memorize vocabulary through flashcards. Users can input vocabulary words, definitions, and their category (noun, verb, etc.) into the flashcards. They can then quiz themselves by flipping through the flashcards. Think about the necessary data types and what would be the most suitable data structure for this language-learning app. Between tuples, lists, and dictionaries, which would you choose? Think about their respective advantages and limitations, and where flexibility might be useful if you were to continue developing the language-learning app beyond vocabulary memorization.

   I think the dictionary structure still holds the most weight in the flashcard app because you can create a dictionary for each word. Using the example, the word will have its definition and part of speech stored with each word. If this were all that was being stored, you could simply create a tuple with 3 elements ('the word', 'the definition', 'the part of speech') and use that static structure to write your program. You *could* then create a list of those tuples so that you could add, remove, sort the list but you just would not be able to update values within the tuple. However, to expand on this application you may want to add other things to each word flashcard like a word may have multiple parts of speech, or multiple definitions. The flashcard may include using the word in a sentence (which would need to be provided for each definition). This expansion would be very tedious with a list of tuples, so the list of dictionaries would be much more practical in terms of growing the flashcard into something more complex.

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!)*

```
dest = str(input('Enter your travel destination: '))

if dest == 'London':
    print(f'Enjoy your stay in {dest}!')
elif dest == 'Tokyo':
    print(f'Enjoy your stay in {dest}!')
elif dest == 'Sydney':
    print(f'Enjoy your stay in {dest}!')
else:
    print('Oops, that destination is not currently available.')
```

2. Imagine you're at a job interview for a Python developer role. The interviewer says "Explain logical operators in Python". Draft how you would respond.

   Logical operators are a way of building conditional statements we can use in our code to check for specific circumstances and then based on that execute a specific block of code. You could check for meeting several conditions with the and operator, one condition out of several using the or operator or check to see that a condition isn't met using the not operator.

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

   Functions are a way of creating reusable code blocks that perform a specific task. They are useful because you won't have repetitious code in your files making them harder to follow and also harder to debug. Debugging the function for errors is much easier and efficient than checking the 10 different occurrences of that code block if you were to write it all out without functions. It makes it easier for the developer because with a properly working function, you know what you are passing into it and you know what to expect out of it. It also benefits anybody else who may work on this code because of the improved readability with using functions.

4. In the section for Exercise 1 in this Learning Journal, you were asked in question 3 to set some goals for yourself while you complete this course. In preparation for your next mentor call, make some notes on how you've progressed towards your goals so far.

- Learn the basics of backend scripting with Python and the parallels with JavaScript.

  This is progressing as I'm getting the feel for using logical and comparison operators, building loops and functions and even exploring out of the lesson for some questions that arise as I think about how Python would handle things. I see many parallels to JavaScript as most languages would support the operators, functions, variables, but it has a much simpler syntax in Python for writing loops or conditions than stringing together several methods in JavaScript.

- Learn how to create a server and manage database connections with Python.

- I would like to rewrite my portfolio email backend server project with Python. I created a Node.js / Express.js server with Resend to create and send emails with the data posted from my contact form. I think it would be good practice to be able to write this same server using Python.

  The last two goals are still yet to be encountered but I know they will be in the near future. I see that file I/O is the next heading so that will be interesting to learn as well. I recall way back in the early 2000s that file I/O in Core Java was a beast with wrapping a BufferedWriter around a FileWriter and the same sort of thing for reading files... and just to simply read a line or write a line was considerable code work. I am sure by now that has all improved in the language, but the example in Exercise 1.1 with reading a file was incredibly simple and basically a short line to open the file and a single command to read the lines. It's much more convenient and it gives developers more time to think about other things instead of which two wrapper classes are correct for reading a file. It's exciting seeing the progress.

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 in Python is important because using the shell to run scripts that need access to data from a previous session need someplace to hold that information. Sessions do not have persistent data and once the script completes or you quit the shell, the data that was stored during that session is erased. Storing information in a file is a way to allow scripts to load and store the data for future use.

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?

   Pickles are modules used to load or store complex information objects, such as dictionaries, nested objects or other data structures. They allow a Python script to read and write to binary data files which can store complex information quickly and in a very compact manner. They do have drawbacks like lack of security, so sensitive information should not be stored with pickles. But for complex, non-sensitive data storage that is suitable for a file, pickles are a great way to go about it.

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?
   To find both you need the os module imported and you could use os.getcwd() to find the current working directory or os.chdir(<path to new dir>) to change to another one.

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?

   You would use a try-except construct to run the code in question of throwing errors inside the try-block and then you can use except-blocks to catch specific exceptions or any exception and handle it with a message to the user or some other logic. With this construct in place, you can safely catch exceptions, and the script will pick up executing the code after your try-except blocks. Without them, an exception will halt execution and terminate the script.

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 it's going well so far, I'm getting better at the syntax and having less errors when I run my code. I am proud of the fact I have used principles from the Full-Stack Immersion course and applied it to Python in terms of keeping code more modular, creating reusable functions and even thinking through the code-along sections when walking through the lessons for weak points where issues may still be unaddressed by error handling or if an edge case might arise with data. If I had to pick something so far that I would say is struggling, I still must look 2 or 3 times whenever a lambda function pops up if it's not something straight-forward. The sorting of the heroes list using a lambda to set the key was tricky and I had some AI explanations to try to explain that logic. I might try to find some practice ones to write and see if I can't get better at writing and recognizing how they work with less trouble.

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 is a methodology where classes represent some entity which can be technical (a data parser, a database connection, a file I/O stream, etc.) or non-technical (a

patient, a bank account, a recipe, etc.) that have fields that store data as well as methods that perform tasks related to the class. The classes are instantiated as objects, and each can perform tasks and store data independently or in conjunction with other objects. Some benefits of using OOP are the ability to use inheritance and polymorphism to create reusable code in a very structured manner.

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

   Classes define some type of classification or entity, and objects are the actual specific instances of that class. For example, you could create a class called `StoreItem` that contained fields pertaining to item information, such as `name`, `price`, `sale_price`, `quantity`, and then define methods like setters/getters for `retail_price`, `sale_price`, `quatity`, or an `on_sale` method that returns if the item is on sale and would contain logic to display the `sale_price` vs `retail_price`. You could create objects for each item in the store using that class and maybe your business has an application that will track all `StoreItem` objects as part of monitoring inventory and sales.

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

Method	Description
Inheritance	Inheritance is a feature where you can define a class with variables and methods, then use it as a parent class. Other classes can be defined as child classes and "inherit" from the parent class. This allows all variables and methods defined in the parent to be available in the child without needing to manually define them all over again in each child class.
Polymorphism	The ability to have different behaviors for inherited methods from a parent class. If you had a parent class called `Baker` and you created 3 subclasses of `Baker` like `CakeBaker`, `CookieBaker` and `BreadBaker` each inheriting a method called "`bake()`" from `Baker`, they would all bake different things per se. They would each call their `bake()` method but one would bake cookies, one cake and the other bread because those methods were overridden with custom definitions in the subclasses.
Operator Overloading	Operator overloading is a way to customize comparisons with atypical objects. You can specify each operator and define a new definition that returns a boolean response. You then can use this custom operator to compare two objects of this class to one another.

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 structured storage systems that allow storing of data with certain rules about what data can be stored and how it relates to other data. It uses SQL, or structured query language, to engage in database operations through the DBMS, or database management system. It allows CRUD operations and can be used to pull very specific information from the database when specified conditions are met.

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

Data type	Definition
INT	An integer value which can be positive, negative or zero.
FLOAT	A floating point decimal value that can be positive, negative or zero.
VARCHAR	Variable length character string that is set with a max character limit. The string will only take up as many characters as needed (no padding to full length).

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

   SQLite is best suited for smaller data pools, direct disk writing, local single-user programs or game applications. It doesn't have the robust security, concurrency or handling of large quantities of data as a MySQL database server would be able to provide.

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

   There are lots of common points between the languages in that they have features/capabilities like OOP, server connection management, database connection management, file I/O, functional programming, etc... However, one major difference is the simplicity of the syntax in Python vs JavaScript and how easily some of these features are implemented and used in Python. Applications in JavaScript often need many external packages to be installed to provide functionality where Python includes a great deal of this functionality out of the box. Another

difference is the lack of browser support for Python, whereas JavaScript is supported by all browsers.

5. Now that you're nearly at the end of Achievement 1, consider what you know about Python so far. What would you say are the limitations of Python as a programming language?

   I think front-end development is one major limitation for Python since you can't do client-side scripting in the browser. I imagine if you could, it would be something similar to the difference between React and Angular when it comes to using forms. Albeit, the Angular learning curve is quite steep, the benefit of having all your data bindings to form fields is extremely helpful when processing user input. If using React, you would need to have event handlers get each element from the DOM and then collect the values to process the form. It would be interesting to see if Python expands into the browsers in the future.

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?

   It is a tool that is almost like a universal television remote, but for relational databases. The remote has basic functions that all remotes would have function buttons like channel, volume, mute, input buttons (or insert, select, update, delete functions in a database) and the remote will communicate with whatever television (or relational database) and send these common functions. It allows you to create classes that will represent your table and use objects to call methods that carry out SQL queries.

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?

   I feel that I did well with ensuring the user enters valid input for each request and my checks ensure that no invalid input gets through. I tried to make the menus different for each selection by adding some colored emoji squares to distinguish one menu from another. If I could change something, I would try to add more specific error handling to the program for each possible exception that may be thrown when calling the ORM functions.

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 used Python to create a command line recipe application that provides users with CRUD functionality to a recipe database as well as some searching features into ingredients used in each recipe. The application uses SQL Alchemy ORM to manage the database functions, and while working on this project I learned about using them to create the queries using the ORM methods.

4. You've finished Achievement 1! Before moving on to Achievement 2, take a moment to reflect on your learning in the course so far:
   a. What went well during this Achievement?
      Learning the language went well and just getting used to the new or more simplistic ways of doing certain things. There were a few newer methods I learned from searching out warnings I received and found updated methods in SQL Alchemy 2.0+ vs the legacy approaches.
   b. What's something you're proud of?
      Taking on the challenge of investigating Docker before Exercise 1.6 to create a containerized database locally and get a grasp of how you can use it to isolate an environment and run it on your machine.
   c. What was the most challenging aspect of this Achievement?
      Working through the ORM and manipulating the data going in or coming out of the database. Matching types, ensuring lists became strings for the database and back to lists to use in my Python code was tricky at times.
   d. Did this Achievement meet your expectations? Did it give you the confidence to start working with your new Python skills?
      Yes, it was a great foundation to start with Python. From the basics, to file I/O and pickles and finally database interactions, I think it covered a great starting footprint for learning the language.
   e. What's something you want to keep in mind to help you do your best in Achievement 2?
      Explore the paths for new ideas in Python since the language is new. What ways can I do things more efficiently or optimize different processes.

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

Pre-Work: Before You Start Achievement 2

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

- Was your study routine effective during Achievement 1? If not, what will you do differently during Achievement 2?
  I decided to take less physical notes on the lesson than I usually do to save some time, but the notes do help me remember more and think about what it is that I am actually writing. I have come up with many questions and points for clarification while taking notes and I typically go on my tangent then to find the answers and include them in my notes. I'm still taking some Notepad notes but they are very concise or specific commands I want examples. I think I did fairly well absorbing the content this way and saved some time.
- Reflect on your learning and project work for Achievement 1. What were you most proud of? How will you repeat or build on this in Achievement 2?
  I'm proud of getting some experience with Python basics and creating a simple application that interfaces with a database.
- 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?
  I ran into issues at times with outdated modules or methods and found warnings/errors where I had to go to research the most recent way of calling a method. Some of the SQL Alchemy ORM usage was tricky, but I tend to start logging values going in and coming out of functions when I am learning something new to try to see what exactly happens or why I might be seeing strange output.

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

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

Exercise 2.1: Getting Started with Django

Learning Goals

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

Reflection Questions

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

   If you go with vanilla Python, you have absolute control over the code and how it works. If you are building a small application without database interaction or multiple users with sensitive data,

it's easier to begin with Python vs trying to implement the Django framework. However, if the application is going to begin to grow into a high-traffic web app and you need to scale, it will be harder to scale over time without Django. If there are going to be more features added along the way (planned or unplanned) it would be easier to implement with Django as well. If the application does need database, user management and need for authentication and security, vanilla Python will take much more time to build these features vs using Django where they are already ready for use.

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

   Not having to define the controller logic code in the MVT architecture because it will be implicitly handled by the Django framework is probably the greatest advantage. Otherwise, they seem quite similar but swap terminology with some of the parts which are a little confusing. If it were "MTV" you could compare and say "model is model", "view is template", "controller is view" (but since there is no controller it's just the logic layer).

3. Now that you've had an introduction to the Django framework, write down three goals you have for yourself and your learning process during this Achievement. You can reflect on the following questions if it helps:
   - What do you want to learn about Django?
     How the applications are structured, how the framework allows for scaling, how to use the out of the box features, and why it makes a good fit for so many web applications.
   - What do you want to get out of this Achievement?
     A solid grasp on how Django works, so I feel comfortable using it if I were required to build an application using the framework.
   - Where or what do you see yourself working on after you complete this Achievement?
     I would still like to rewrite my email forwarding server to be used in conjunction with my portfolio contact form using Python. Since my portfolio is very simple and static html pages, Django would be overkill for that. But it might be fun to try writing the myFlix application using Django for practice.

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.*)

   The website itself would be considered the Django project. Within that project there would be many apps with different functionalities such as schedule, team, news, stadium, parking, tickets, photos, videos, mobile-app, podcasts, tv-radio, shop, etc(all entries on the menu bar could potentially be apps), chat-assistant, pop-up-video and game-tracker. Anything that can be functionally isolated could be apps that communicate with the database or each other. Models for the project could include team, player, article, product and ticket and the apps could interact with them accordingly. Templates could be created for displaying team-roster, player-info, news-stories, photo-media, video-media, stadium-seating, ticket-search, and merchandise.

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

   First, you need to have Python installed because Django is a Python framework. Then once you have Python installed, you will need to create a virtual environment. Install Django on this virtual environment. This lets you use Django to create a project. Once the project has been created, you will need to run a migration to setup the database. Once the migration completes successfully, you can either start the server up at this point or create a superuser and then start the server. This will bring up your project on a local server you can access through your web browser.

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

   The panel allows you as an admin/superuser to perform CRUD operations on models (or tables) in the database using the panel. This can be helpful during development to populate a table or change data on the fly if need be. It also allows for creating users and assigning permissions. For the recipe app, it might require some basic authentication, and then the average user may only be able to add/update/delete their own recipes in the database but have access to read all recipes. That way they won't change other user's recipe data. These permissions could be assigned via the admin panel (but should be assigned by default with the app when a user creates an account and saves the panel for creating higher level privileges). The panel can also be customized to show the models differently if there is something special that needs to be displayed, but I don't think we would need that for the recipe app as the database is not extremely complex.

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 a way to create a class that will translate into a database table using the Django ORM. This model can then be instantiated (since it’s a class) creating an object that will represent a row in the table for that model. This way we can create objects and work with them and the ORM will handle all database queries for us using the model and instances.

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.

   It’s important because if you start your test cases early, you have a smaller project to deal with. It would be overwhelming to take a complete codebase and then have someone say, “Can you write the unit tests for this application?” The test cases should be done as early as possible when the code is smaller and manageable. If I were working on a module, it would be beneficial to myself and others who may work on my code to complete the test cases to ensure that if anything changed, these tests show if the changes broke the expected functionality of the module.

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.

   Views handle requests coming in and return a template back to the web browser like a web page, JSON or a redirect. The view will also hold any business logic that might pull data from a model that is required for rendering the template. The view is like the controller in the MVC architecture.

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?

   Class-based views would be best because if you are going to need to reuse a lot of view code, you could create a view super class and then extend that for each of the sub classes with customizations. CBVs allow for polymorphism and inheritance through Python's OOP support.

3. Read Django's documentation on the Django template language and make some notes on its basics.

   Templates are text files that generate other text-based files (HTML, CSV, JSON, etc...). Templates contain variables and tags. Variables, denoted by `{{variable_name}}`, are replaced with values when the template is evaluated. Tags, denoted by `{%tag_name%}`, are logical constructs in the template.

   Variables can be modified using filters, `{{name|lower}}` for example would apply the lower filter changing the value of name to lowercase before replacing it in the template. There are many filters and they perform all kinds of functions from manipulating values, setting defaults, getting information such as length of list/string, etc.

   Templates also have some security built into the template language and will escape all output for every variable to ensure XSS attacks are prevented. This is the default behavior with one exception: string literals are not escaped and act as if they were passed through the safe filter.

   Methods and attributes are still accessed with dot operator.

   In summary, the template language gives a way for creating the text output file by using tags and variables to control values and logical flow in the file.

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.
   Static files are assets that are used in the templates such as images or CSS stylesheets that must be stored in a specific folder structure and their paths specifically defined in the settings.py of the project.

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	ListView is a generic class-based view that will use a model and returns all records for that model as a list that the template can iterate over.
DetailView	DetailView is another generic CBV that will pull a single record from a model via a primary key and return that record to be used in the template.

3. You're now more than halfway through Achievement 2! Take a moment to reflect on your learning in the course so far. How is it going? What's something you're proud of so far? Is there something you're struggling with? What do you need more practice with? You can use these notes to guide your next mentor call.

   The Django framework is challenging because it's another new "way" of doing things but it is starting to get easier with each day I look at it. I'm pleased with learning the new framework and proud that I can start to debug some of the issues and learn more about little details of Django. I have picked up on some small details that help like filter functions or just why some things will work in development but are not suitable for a production environment like adding the `MEDIA_URL` and `STATIC_URL` paths into the `urlpatterns` with a `+=` operator. I still need more practice with the structure for it all to become second nature, but that's understandable for working with it for the first time in this course.

4. Frontend Inspirations for the Recipe Application:

AllRecipes has some nice features like interactive display cards for recipes that show a testimonial on the back (but could show other information) and a link to the recipe details page. The details page shows images, cook time, total prep time, servings and then contains instructions, ingredients and even a pop up modal with a video detailing the process. The layout is not cluttered, and everything is visible and clear.

Chowhound has a nice listing for recipes and shows many options and categories that user can choose and search for recipes using. The site even offers shopping guides and local restaurant suggestions that align with users' meal search criteria. This would make for several new apps in the Django way.

Bon Apetit is a site I came across that I did not like the design of because it felt like there was no organization on the main page. The images were too large, and you had to scroll past the images to get to the links or text associated with them. The recipes were locked to registered users, but the layout of their detail pages also showed very large images and lots of scrolling to find information. There was also a point on the main page where recipes turned into news stories about the Starbucks strike. Overall I would say my inspiration would be to NOT create my application with images that take up the entire screen and requiring users to wear out their mouse wheel trying to simply find the information. I more concise layout is what I will aim for, so the important information is front and center. I did like the user ratings for recipes where they could rate 1-5 stars on a particular recipe, and the overall score was shown on the details page.

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.

   Authentication is a way of tracking users, keeping their information secure and limiting what access they may have within an application. For example, two users go to a banking application. It would be horrific to have User A be able to access User B's account and transfer money, open loans, etc... Without proper authentication to ensure the user performing these actions is indeed the owner of the account. Authenticated users should ONLY have access to their specific account and information and only perform actions related to their status. The user may be a

bank patron, a teller or a branch manger and each having access to different functionality within the application. A patron should not have access to bank transaction records that a teller or branch manager may.

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

   You must first create a view for the login at the application level (and may need to create a views.py file in the application folder). Within this view, the login form must be validated, and then the authentication is performed. If successful, the application can redirect to a particular view as the user's starting point in the application, otherwise return an error message and render the login page with it to display back to the user.

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()	A function that takes user credentials (username and password) and checks against the backend User model used by Django which stores users' records.
redirect()	This function returns an HttpResponseRedirect to the appropriate URL. If you specify an object, the object's get_absolute_url() function is called. If you specify a view, it will call the reverse() function to get the URL. You can also use a relative path or absolute URL.
include()	This function is used in the URL mappings to pull in another app's mappings in its own urls.py file.

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.

   By noting views on products the website could identify which products were viewed in the same session and use that data to suggest those other items to shoppers that may not have looked for other alternatives. It can also show when peak shopping periods are in terms of time of day, week, year, holidays, etc and coordinate sales/events to help draw maximum customer traffic and sales. Trends may also show where specific categories of items might be searched/viewed frequently and this could be used to perhaps notify a shopper about sales or new products that they frequently view/purchase.

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

   `QuerySet` will be evaluated under several circumstances:
   - If you iterate over the `QuerySet`, you must know what is in this set and it's evaluated.
   - If you `slice()` the `QuerySet` and include the step argument, this will also evaluate it and return the sliced portion as a new `QuerySet`. The documentation states that you can slice an unevaluated `QuerySet` (omitting the step argument) but the result will also be an unevaluated slice of the query set and might corrupt the SQL that evaluates this slice.
   - If you are storing the `QuerySet` in cache with a `pickle`, it will be evaluated.
   - When using `repr(qs)` on the `QuerySet`, which returns a string representation of the parameter, it will be evaluated.
   - When using the `len()` function to find the length, the `QuerySet` must first be evaluated to determine the length of the list.
   - If you call a `list()` on the `QuerySet`, it will evaluate it and return the list.
   - If you are checking a Boolean condition with an `if`, `and`, `or`, `not`, `bool()`, etc, it will evaluate the `QuerySet` in order to determine the Boolean value.

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.

   `QuerySet` is best for filtering and searching data because it operates in the database, so these are very quick operations, even on very large data sets. They `QuerySet` also does not evaluate until it needs to be so there is no hit on the database until necessary. `QuerySet` is NOT suited for complex transformations/formatting of the data, creating visualizations, rendering HTML or statistical/numerical calculations. These are the strengths of the `DataFrame`. `DataFrame`,

however, does not lazy load and triggers an evaluation immediately when used and also may not work well with sufficiently large data sets because it has to store the entire data set within memory (which is a system limitation).

Exercise 2.8: Deploying a Django Project

Learning Goals

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

Reflection Questions

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

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

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

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

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