CAREERFOUNDRY

Python for Web Developers Learning Journal

Objective

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

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

Directions

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

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

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

Pre-Work: Before You Start the Course

Reflection questions (to complete before your first mentor call)

1. What experiences have you had with coding and/or programming so far? What other experiences (programming-related or not) have you had that may help you as you progress through this course?
    a. *I've been a QA engineer for the last eleven years and have written automated test suites using C#, JavaScript and TypeScript. There was one project where I got to write a few tests using Python before our contract ended, but my knowledge of Python is very basic.*
2. What do you know about Python already? What do you want to know?
    a. *I think the idea of the virtual environment setup is interesting, but it's a bit confusing to get the hang of, especially after using simpler setups with NPM and Node. The idea of built-in packages is nice, especially after working with so many conflicting dependency error in the past. My QA coworkers use Python for their API testing, so I'm excited to see how I can apply this if I ever become involved in the backend automation.*

3. What challenges do you think may come up while you take this course? What will help you face them? Think of specific spaces, people, and times of day of week that might be favorable to your facing challenges and growing. Plan for how to solve challenges that arise.
    a. *I think getting used to the virtual environment setup and language syntax will be the most difficult parts of this course. When I've worked with Python, I end up rewriting pieces because I'm so used to JavaScript and TypeScript, but that's just a habit. Understanding the packages and functions is going to be a challenge, but I'm excited to dig deeper into those and find out as much as I can about Python's capabilities.*

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

Exercise 1.1: Getting Started with Python

Learning Goals

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

Reflection Questions

1. In your own words, what is the difference between frontend and backend web development? If you were hired to work on backend programming for a web application, what kinds of operations would you be working on?
    a. *Frontend development is creating the visible part of the application, which is what the end-user sees. Backend development handles the data transfer, storage and security. Working on the backend of an application may involve working with databases, APIs, serverless functions and security implementations.*

2. Imagine you're working as a full-stack developer in the near future. Your team is asking for your advice on whether to use JavaScript or Python for a project, and you think Python would be the better choice. How would you explain the similarities and differences between the two languages to your team? Drawing from what you learned in this Exercise, what reasons would you give to convince your team that Python is the better option? *(Hint: refer to the Exercise section "The Benefits of Developing with Python")*
    a. *Both Python and JavaScript are high-level, dynamically-typed scripting languages that have large library selections. While both languages are readable, Python comes out ahead in terms of readability, making it a great option for beginners.*
3. Now that you've had an introduction to Python, write down 3 goals you have for yourself and your learning during this Achievement. You can reflect on the following questions if it helps you. What

do you want to learn about Python? What do you want to get out of this Achievement? Where or what do you see yourself working on after you complete this Achievement?

a. *I want to get more in-depth knowledge of Python's libraries.*
b. *I'm excited to dig into the object-oriented programming and database exercises because those are the areas I really want to focus on improving.*
c. *I want to learn as much as I can in this achievement because I want to have an opinion on Python versus other languages I've used in the past. Python is a language that always sounded intriguing, but I never had enough time to dedicate to learning it until now.*

Exercise 1.2: Data Types in Python

Learning Goals

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

Reflection Questions

1. Imagine you're having a conversation with a future colleague about whether to use the iPython Shell instead of Python's default shell. What reasons would you give to explain the benefits of using the iPython Shell over the default one?
    a. *My reasoning for using IPython over the default shell would be its tab completion and command history. Both of these functions save time while using the IPython shell, making the development smoother. Also, the pretty-printing default makes viewing nested data types much easier.*

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?
bool	Stores two values: True and False	Scalar
NoneType	Stores one value: None, which is similar to null	Scalar
Tuples	Linear arrays that can store values of any type	Non-scalar
Dictionaries	Unordered set of items, each with a key/value pair	Non-scalara

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

    a. *A tuple is an indexed, linear array that can store values of any type without giving each variable a unique name. An example for miles per hour(mph): mph = (45, 25, 50, 70, 30).*
    b. *A list is similar to a tuple except that a list is mutable, which is ideal for situations where values may need to be added, removed or sorted. Lists are a bit more flexible than tuples, but tuples have greater readability and ease of access.*
4. In the task for this Exercise, you decided what you thought was the most suitable data structure for storing all the information for a recipe. Now, imagine you're creating a language-learning app that helps users memorize vocabulary through flashcards. Users can input vocabulary words, definitions, and their category (noun, verb, etc.) into the flashcards. They can then quiz themselves by flipping through the flashcards. Think about the necessary data types and what would be the most suitable data structure for this language-learning app. Between tuples, lists, and dictionaries, which would you choose? Think about their respective advantages and limitations, and where flexibility might be useful if you were to continue developing the language-learning app beyond vocabulary memorization.
    a. *For an application like this, I would choose a dictionary for its key/value pairing capabilities. A dictionary would cleanly store words, definitions and categories, as well as make them easily accessible when they're printed.*

Exercise 1.3: Functions and Other Operations in Python

Learning Goals

- Implement conditional statements in Python to determine program flow
- Use loops to reduce time and effort in Python programming
- Write functions to organize Python code

Reflection Questions

1. In this Exercise, you learned how to use **if-elif-else** statements to run different tasks based on conditions that you define. Now practice that skill by writing a script for a simple travel app using an **if-elif-else** statement for the following situation:

    - The script should ask the user where they want to travel.
    - The user's input should be checked for 3 different travel destinations that you define.
    - If the user's input is one of those 3 destinations, the following statement should be printed: "Enjoy your stay in _____!"
    - If the user's input is something other than the defined destinations, the following statement should be printed: "Oops, that destination is not currently available."

Write your script here. *(Hint: remember what you learned about indents!)*

```
destination = input("Where would you like to travel? ").strip()
available_destinations = ["Pittsburgh", "Charlotte", "New York"]


if destination in available_destinations:
    print(f"Enjoy your stay in {destination}!")
else:
    print("Sorry, that destination is not currently available.")
```

2. Imagine you're at a job interview for a Python developer role. The interviewer says "Explain logical operators in Python". Draft how you would respond.
    a. *Boolean logic operators -- and, or and not -- allow us to assert multiple conditions at the same time. The **and** operator requires all conditions to match, or else the output value is **False**. If only one condition in a set of conditions needs to be met to achieve a value of **True**, then you'd use the **or** operator. Using the **not** operator doesn't require conditions because the operator is used to reverse the logical expression following it.*

3. What are functions in Python? When and why are they useful?
    a. *Functions contain instructions that create or modify code to achieve a specific outcome. Custom functions reduce repetition in the code base because they can be called anywhere the functionality is required.*

4. In the section for Exercise 1 in this Learning Journal, you were asked in question 3 to set some goals for yourself while you complete this course. In preparation for your next mentor call, make some notes on how you've progressed towards your goals so far.
    a. *Throughout the first three exercises, I've become much more familiar and comfortable using the IPython shell. Until now, I haven't done much CLI scripting, but I'm starting to see the appeal of it. My previous experiences with programming helped me move quickly through the first few assignments. I'm looking forward to the upcoming exercises, especially Exercise 1.6. I work with our database on an almost-daily basis at work, so I'm excited to gain a deeper understanding of the setup.*

Exercise 1.4: File Handling in Python

Learning Goals

- Use files to store and retrieve data in Python

Reflection Questions

1. Why is file storage important when you're using Python? What would happen if you didn't store local files?
    a. *Data is preserved by using file storage to read from and write to files. Utilizing storage enables data to be displayed in a UI; read and write to documents; and store image files. Without local files, data would not be accessible, or even exist, once the script stops running.*

2. In this Exercise you learned about the pickling process with the **pickle.dump()** method. What are pickles? In which situations would you choose to use pickles and why?
    a. *Pickles convert complex data into bytes to be written to a binary file, which saves machine-readable information. Some situations to use pickles would be saving complex data objects; preserving program states; and short-term data storage.*

3. In Python, what function do you use to find out which directory you're currently in? What if you wanted to change your current working directory?
    a. *To find you current directory, you'd use the **os.getcwd()** command.*
    b. *To change your current directory, you'd use the **os.chdir()** command.*

4. Imagine you're working on a Python script and are worried there may be an error in a block of code. How would you approach the situation to prevent the entire script from terminating due to an error?
    a. *Prompting the user to correct the folder name allows the script to continue running instead of causing the application to terminate.*

5. You're now more than halfway through Achievement 1! Take a moment to reflect on your learning in the course so far. How is it going? What's something you're proud of so far? Is there something you're struggling with? What do you need more practice with? Feel free to use these notes to guide your next mentor call.
    a. *I'm proud of how quickly I've learned using the IPython shell. It may be simple, but it's something I haven't used before, but I'm feeling much more comfortable with it after the first four exercises. I think the different types of exceptions is an area I could understand better. Other languages I'm used to mainly use **catch()**, so having a few options is new, but I like the specificity of more exception types.*

Exercise 1.5: Object-Oriented Programming in Python

Learning Goals

- Apply object-oriented programming concepts to your Recipe app

Reflection Questions

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

a. *Object-oriented programming a way of programming that organizes code into objects that bundle data and its code rather than functions and logic. Classes define the structure and behavior, whereas objects are created from the class.*

2. What are objects and classes in Python? Come up with a real-world example to illustrate how objects and classes work.
    a. *Objects contain data and procedural attributes that are instances of the class blueprint. A real-world example would be a report card. The class would consist of a student's name, ID, subject, a dictionary of grades for each subject, along with the ability to add grades and calculate the student's grade point average. The objects would be each individual student's report card which includes the class data.*

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 allows a child class to inherit properties and behaviors from another class, creating a parent-child relationship. Using inheritance enables the developer to create classes that utilize the attributes and methods of the parent class, while also adding its own custom attributes and methods. Having the parent-child relationship between the super-class and sub-class enables developers to write easily maintainable, reusable code that follows the DRY principle. An example of inheritance is a Car class that contains attributes for speed and color, as well as methods for starting and stopping. Sub-classes could be things like Sudan, Motorcycle, Truck, Van, with each sub-class adding their own specific attributes in addition to the ones inherited from the Car class.
Polymorphism	With polymorphism, different classes can respond to methods in their own specific ways, adding abstraction more modular and extensible. Compile-time polymorphism allows developers to utilize the same method name multiple times in a class, with each method containing different parameter lists. When this type of polymorphism is used, the compiler finds the necessary method based on the passed arguments. Runtime polymorphism allows sub-classes to create its own implementation of an inherited method.

Operator Overloading	Developers can use operator overloading to redefine an operator's use in custom classes. Something developers should be wary of is creating confusing overloading, such as * being used for subtraction or + for division. Using operator overloading can help make code more readable, if it's properly written. In Python, developers need to define a function with a name reserved for the operator, surrounded by double-underscores, like __add__().

Exercise 1.6: Connecting to Databases in Python

Learning Goals

- Create a MySQL database for your Recipe app

Reflection Questions

1. What are databases and what are the advantages of using them?
    a. *Databases use organized collections to easily store and access data with a standardized format. Password access enhances database security, and the universal formatting enables database access for other applications. Data can be added to databases, and existing data can be retrieved, modified and deleted.*

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

Data type	Definition
VARCHAR	This data type holds a string of variable length with the maximum number of allowed characters defined in the format.
INT	This data type holds standard integers and only integers.
DATETIME	This data type holds date values in a YYYY-MM-DD HH:MM:SS format.

3. In what situations would SQLite be a better choice than MySQL?
    a. *SQLite would be fitting for a small project, or a simple test, because it does not require installation or setup. Data can be stored in **.db** files and easily accessed through other applications, including Python.*

4. Think back to what you learned in the Immersion course. What do you think about the differences between JavaScript and Python as programming languages?
    a. *Between the two languages taught in this bootcamp, I prefer JavaScript. In my professional life, I've used TypeScript, and I prefer TypeScript over JavaScript. The type setting and early error detection are things I really appreciate. Personally -- and possibly because I haven't truly used it before -- I'm not really a fan of using colons in place of parentheses; the lack of curly brackets; and the use of snake case. Maybe it's because of my familiarity with languages other than Python that give me a heavy bias, but for me, it's harder to read Python and separate functions, loops, etc. That said, I am curious about more of Python's capabilities, and would like to dive deeper into the language and its libraries.*

5. Now that you're nearly at the end of Achievement 1, consider what you know about Python so far. What would you say are the limitations of Python as a programming language?
    a. *So far, I think most of the limitations I've encountered are due to the lack of familiarity of Python. The last decade of my career has revolved mainly around JavaScript/TypeScript, C# and SQL, with a month or two of testing with Python, which was limited to the I/O library. I think it's the syntax deviation that's making it difficult for me. For example, I have automated utility scripts that connect to a PostgreSQL database and run parameterized* ***SELECT****,* ***INSERT****,* ***UPDATE*** *and* ***DELETE*** *statements, but I found it more difficult. It's definitely been my personal shortcomings, but those also make me more interested in continuing to learn Python, especially with it having such a big impact in the industry.*

Exercise 1.7: Finalizing Your Python Program

Learning Goals

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

Reflection Questions

1. What is an Object Relational Mapper and what are the advantages of using one?
    a. *Object Relational Mappers eliminate the need for SQL syntax by converting the structure and contents of databases into classes and objects, allowing the developer direct access to the data. ORMs can be used in web application frameworks, like Django, without the need to push SQL queries for each operation.*

2. By this point, you've finished creating your Recipe app. How did it go? What's something in the app that you did well with? If you were to start over, what's something about your app that you would change or improve?

    a. *If I could start over, I'd add step-by-step instructions so that the user could have everything they need to add, modify or delete entire recipes. The recipe instructions seem to be the only thing missing from having an entire recipe, though I understand that including instructions may be overdoing it for assignment purposes, especially depending on the chosen recipes.*

3. Imagine you're at a job interview. You're asked what experience you have creating an app using Python. Taking your work for this Achievement as an example, draft how you would respond to this question.
    a. *I'd say: "I built a Python command-line application that adds, modifies and deletes recipes, and it uses an Object-Relational Mapper to interact with a MySQL database."*

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?
    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 Python skills?
    e. What's something you want to keep in mind to help you do your best in Achievement 2?

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 of the Orientation course if you're not sure whom to reach out to for help and support.

Exercise 2.1: Getting Started with Django

Learning Goals

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

Reflection Questions

1. Suppose you're a web developer in a company and need to decide if you'll use vanilla (plain) Python for a project, or a framework like Django instead. What are the advantages and drawbacks of each?
    a. *Vanilla Python is a flexible, lightweight option that gives developers complete control, but its slower development process, lack of security implementation and maintenance may be a few reasons developers might opt for a framework. Django is an easily-scalable, secure, batteries included, fully-fledged framework that enables rapid development. Some things that may make developers shy away from Django are its opinionated architecture and learning curve, and having a full framework may be too robust for small projects.*

2. In your own words, what is the most significant advantage of Model View Template (MVT) architecture over Model View Controller (MVC) architecture?
    a. *I think the removal of writing controller and frontend code, mapping URLs and sending lists to users are the biggest advantages of a Model View Template for developers. Taking those things out of the equation could be a big time-saver for one or more developers, allowing them to focus on other aspects of their project.*

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?
        - *In my current role as a QA engineer, instead of using our database, I use the Django admin interface to provide guidance to our product support team. I'd like to learn more about how the interface is built and maybe identify ways or places where I can help.*
    - What do you want to get out of this Achievement?
        - *I use the admin interface at work, but having tested custom features, I know there's a lot more that goes into it. I'd like to better understand what our engineering team in doing behind the scenes.*
    - Where or what do you see yourself working on after you complete this Achievement?
        - *Once I complete this achievement, I'd like to return to this specialization and see if I can practice by creating applications similar to the Recipe app just to reinforce what I've learned in these exercises. The immersion part of this*

bootcamp spent a lot of time on JavaScript and React, and I'd like to feel as confident with Python as I do with JavaScript.

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.*)
    a. *If I were to convert the Amazon Music website, I'd create the **amazon_music_project**, which would include the following applications:*
        i. ***accounts** to handle profiles and authentication*
        ii. ***homepage** to handle the homepage and features*
        iii. ***search** to handle search functionality*
        iv. ***recommendations** to handle songs and artist recommendations*
        v. ***compilations** to handle types of things like frequent plays, end-of-year playlists, favorites*
        vi. ***playlist** to handle the creation of playlists and storage*

2. In your own words, describe the steps you would take to deploy a basic Django application locally on your system.
    a. *To deploy a Django application locally, I'd create and activate a virtual environment with **mkvirtualenv env_name**; install Django with **pip install django**; create the project with **django-admin startproject project_name**; apply database migrations with **python manage.py migrate**; create a superuser with **python manage.py createsuperuser**; run the server with **python manage.py runserver**; then navigate to **http://localhost:8000/admin**; and log into the admin site with the superuser's credentials.*

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

    a. *I'd use the Django admin site to extend capabilities to others in less technical roles; creating an option for those who may not know how to seed a database through SQL or other backend means enables them to work with data as they need it. In my current company, we specifically chose Django so that our customer support team has greater data visibility, as well as the opportunity to fix issues that may not necessarily require a developer. Django's admin site provides an easy solution for users, technical and non-technical, to work with data.*

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.
    a. *Django models are Python classes where each attribute represents a field in a database. Models automatically check data types before committing to the database, and the model code can work across different databases without the need to rewrite its data layer. The admin interface is accessible to tech and non-tech people who, with the proper authorization, can work with data.*

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.
    a. *Testing from the beginning of a project enables features to be fine-tuned before the application is deployed to production. Adequate testing not only verifies expected functionality, but also negative test cases, edge cases, performance and load. In my current role, manual and automated tests must pass before a feature or bug fix is merged into the main branch. This process gives us the time we need to fully test our changes so that we can confidently merge our pull requests. Test cases are especially important when it comes to complex features where there may be dozens of scenarios. Personally, I've had a few large features that required nearly 100 test cases, and without the test cases, I wouldn't have covered all the necessary scenarios.*

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.
    a. *Views represent Python functions or class methods that accepts a request, runs Python code and returns a response. Functions can be as basic as returning static data for an HTML page to more complex uses like interacting with databases or accepting user input.*

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?
    a. *I would use class-based views to take advantage of their OOP principles and extendability. Class-based views would be ideal for large-patterned projects with standard CRUD operations.*

3. Read Django’s documentation on the Django template language and make some notes on its basics.
    a. *The Django Template Language (DTL) is designed to separate the presentation and business logic layers, allowing developers build static and dynamic HTML content generated by the application. The DTL syntax is based on filters, tags and variables.*

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.

a. *Static files are used in Django for content that does not change, like images and CSS. These files are not generated by Python, but they are necessary for the website's behavior. With these files contained in a static location, they can be added to HTML by using {% static %}.*

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 class-based view that displays data as a list.The ListView imports objects from a model and sends them to a template.
DetailView	The DetailView package displays data details by using a single object's primary key to send the object to a 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.
    a. *So far, I'm interested in seeing how Django is used to build a web application. At my current job, we use it as a CMS connected to our Postgres database that allows our customer support team to have a UI to make changes that may not require a developer, so this is a new way of seeing Django's capabilities. I think something I need to focus on is properly connecting views and URLs. It's simple, but I did run into multiple errors because of small changes that I made. Those mistakes gave me the big, ugly error message when I tried to open the recipe 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.
    a. *Authentication is important in that it keeps users, and their personal information, safe and secure. At my current and previous jobs, due to the nature of our work, we were legally required to follow specific security protocols. Not only do authentication practices enhance security, they provide confidence to users about the validity of the product. The better the security, the easier it is to answer, "Is this website legitimate?"*

2. In your own words, explain the steps you should take to create a login for your Django web application.
    a. *To create a login, you first need to include the necessary applications in **settings.py**, then set up the URLs for the login and logout views. Next, you create the template and add views, protecting them by using **@login_required**. Once your login is set up, the admin user can create users and determine their level of authorization.*

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()	This function verifies user credentials, and if the user exists, a **User** object is returned, otherwise **None** is returned. Once credentials are verified, a session is created by **login()**.
redirect()	This is a shortcut function to send users to a different URL based on the information sent to it. Form submissions, logins and similar processes use this function to move a user to a new URL without directly rendering a template.
include()	This function enables developers to assign URL patterns to individual applications, such as routing all paths that begin with something like **user/**.

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.
    a. *LinkedIn collects data similar to a resume: name, location, education history, work experience, and career interests. These collected pieces of data help tailor the job search experience to the user's specific needs. Otherwise, users would wade through a sea of job postings to find something relevant to their career choice. This data also enables the application to suggest potential jobs for the user based on their job titles, work experience and career plans. With these suggestions, users might find out about open positions they wouldn't have found.*

2. Read the Django official documentation on QuerySet API. Note down the different ways in which you can evaluate a QuerySet.
    a. *With the QuerySet API, developers can access and read data in the database, or they can apply operations for iteration, filtering, finding lengths and converting to lists. QuerySet can also be used for slicing, indexing and be used in a boolean context.*

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.
    a. *QuerySets can only access, and work directly with, the database once they are evaluated. Using QuerySets also provides security and its syntax is easily chainable. A few drawbacks are that QuerySets don't work well for statistical computations or data transformations.*
    b. *DataFrame works well with analytics and provides flexibility when working with different data types. While DataFrame works well in numerical scenarios, it also uses a lot of memory and can't automatically write to the database.*
    c. *DataFrame works better with processing data because of its ability to use vectorized computations; it can handle complex data transformations; contains plotting libraries for machine learning and visualization; and solves problems in memory.*

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.
    a. *CSS can be added inline, internally or externally. Once the path to the **static** directory is specified in **settings.py**, the CSS and JavaScript files can be loaded into the HTML by adding **{% load static %}** above the **<head>** tag. With JavaScript, the **src** attribute inside the **<script>** tag will be loaded like this: **src="{% static 'sales/home.js' %}"**.*

2. In your own words, explain the steps you'd need to take to deploy your Django web application.
    a. *To deploy my Django application, I'd create a **Procfile** and commit my final code to my repository; update my application so it can be hosted on a platform like Heroku; make sure my environment variables are properly set so that my database connections work, as well as avoid any secrets being exposed; then push my code to the hosting platform.*

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?
        i. *During this achievement, I gained a new perspective on Django; I know how I use Django as a CMS at work, but actually building a web application with Django allowed to me to see a lot of its capabilities.*
    b. What's something you're proud of?
        i. *I'm proud of how quickly I was able to create an application with Django, and how creating the Admin panel was much less intimidating than I thought it might be to develop.*
    c. What was the most challenging aspect of this Achievement?
        i. *The hardest part of this achievement was understanding the project structure. So many directories, and some files, share the same name. More than once, I ran into errors and was attempting to fix the code when in actuality I was working in the wrong file or directory with the same name. I'd have to say that the directory structure is the main thing that bothered me, but I really liked the built-in functionality and libraries.*
    d. Did this Achievement meet your expectations? Did it give you the confidence to start working with your new Django skills?
        i. *This achievement went beyond my expectations. It was kind of daunting to think about building an entire web application in eight exercises, but Django was intuitive enough to work with, and its libraries made everything much easier than I expected.*

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