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 have coded projects using Javascript, CSS, and HTML. I have used monogdb, Express,

React, and Angular technologies. There is a wide range of projects that I have done, from web apps to mobile apps, using APIs, and other things like simple games.

2. What do you know about Python already? What do you want to know? I have heard that Python is easier to read but much more popular than most languages. It can do more than just create web-based applications. I have seen use in Raspberry Pi to create relatively simple electronics.

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 think challenges that I might face include debugging code, using the command prompt, and things not working that I have no idea how to fix. I will fix them by using Google, chat GPT, my mentor, and stack overflow. I try to stay consistent with my Career Foundry work because it helps me to do it in smaller chunks and stay motivated that way.

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? Frontend web development focuses on the visual and interactive elements of a website that users see and interact with, using technologies like HTML, CSS, and JavaScript. Backend Development handles the server side, managing databases, business logic, and APIs that support the application's functionality. While frontend developers focus on design and user experience, backend developers ensure data processing, security, and server management are efficient. If I worked on backend development, I'd handle tasks like building APIs, managing databases, implementing security, and optimizing performance. Essentially, the

backend ensures everything runs smoothly behind the scenes for the front end to function properly.

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") JavaScript is great for building interactive web apps, especially in the browser, while Python excels in backend development, data science, and automation. Python's simpler, more readable syntax makes it easier to learn and maintain compared to JavaScript's complex async handling. If the project leans towards backend services, data processing, or machine learning, Python's versatility and ease of use make it the better choice.

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?
    - Be able to code functions easily.
    - Understand how to use Python for the web
    - Be able to code projects based on the Python language

    I want to learn how to use Python in and outside of web development. I want to be able to understand Python after this achievement. After this achievement I see myself working on applications on hardware like a Raspberry Pi.

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 is much easier to use instead of Python's default shell. It is much more user-friendly, in that it will show you possible commands when tab is pressed, the syntax is color-coordinated, and I can write multiple lines at once. Each

command is executed instantly, the result output is displayed. This quick and convenient approach makes it much better to test small code snippets.

2. Python has a host of different data types that allow you to store and organize information. List 4 examples of data types that Python recognizes, briefly define them, and indicate whether they are scalar or non-scalar.

Data type	Definition	Scalar or Non-Scalar?
int	integer (whole number)	Scalar
float	decimal number	Scalar
str	string of characters (numbers or letters)	Non-Scalar
bool	boolean True/False statement	Scalar

3. A frequent question at job interviews for Python developers is: what is the difference between lists and tuples in Python? Write down how you would respond. In Python, the key difference between lists and tuples is that lists are mutable, meaning you can modify, add, or remove elements after the list is created. Tuples, on the other hand, are immutable, so once created, their content cannot be changed. Lists are defined using square brackets '[]', while tuples use parentheses '()'. Because tuples are immutable, they can be used as keys in dictionaries, whereas lists cannot. Lists are generally slower than tuples in terms of performance, due to the overhead of mutability. Lastly, tuples are often used when data integrity is important, while lists are better suited for dynamic collections of items.
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. For a language-learning app that uses flashcards to help users memorize vocabulary, the most suitable data structure would likely be a dictionary. Each flashcard could be represented as a dictionary entry where the vocabulary word acts as the key, and its value is another dictionary or list containing the word's definition, category (e.g., noun, verb), and perhaps other attributes like language or difficulty level. A dictionary provides flexibility and quick access, making it easy to look up specific vocabulary by word. Furthermore, it allows for easy updates, such as adding new words, modifying definitions, or changing categories, which is essential in an app where users continuously input and revise data.

   While lists and tuples have their strengths, they lack the flexibility needed for this scenario. Lists are mutable but unordered, which may make retrieval less efficient, especially as the dataset grows. Tuples, being immutable, would limit the ability to modify flashcards over time, which is a drawback when users may want to update or delete entries. A dictionary offers the best

of both worlds: fast lookups (thanks to key-value pairs) and mutability, making it an ideal choice not only for vocabulary but also for expanding the app's functionality to other language-learning tasks like grammar exercises, conjugation tables, or quizzes. This versatility makes it a strong foundation for future development.

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?")

if destination.lower() == "paris":
  print("Enjoy your stay in Paris!")
elif destination.lower() == "tokyo":
  print("Enjoy your stay in Tokyo!")
elif destination.lower() == "new york":
  print("Enjoy your stay in New York!")
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 in Python are used to

perform logical operations on values, typically in conditions or expressions. There are three main logical operators: 'and', 'or', and 'not'. The 'and' operator returns 'True' if both operands are true; otherwise, it returns 'False'. For example, 'True and False' results in 'False'. The 'or' operator returns 'True' if at least one of the operands is true, and only returns 'False' if both are false, as in 'True or False', which results in 'True'. Lastly, the 'not' operator is a unary operator that inverts the truth value, so 'not True' becomes 'False'. These operators are commonly used in conditional statements like 'if' to control the flow of a program based on multiple conditions.

3. What are functions in Python? When and why are they useful? Functions in Python are blocks of reusable code that perform a specific task, encapsulated within a defined structure. A function is defined using the 'def' keyword, followed by a name, parameters (if any), and a block of code to execute. Functions are useful for organizing code into manageable, reusable parts, which makes programs easier to read, maintain, and debug. They also promote the DRY (Don't Repeat Yourself) principle by allowing you to call the same function multiple times without rewriting the code. Functions are especially helpful when you need to execute similar operations multiple times with different inputs, improving both efficiency and modularity in your code.

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. I am glad I am learning functions in Python. I like how the syntax of Python works, it is easier to read in comparison to Javascript. I am looking forward to learning how to apply my knowledge to create web apps using Python. I will focus on using Python in hardware applications, such as a Raspberry Pi later down the road when my knowledge is greater. For now, I am happy the the progress that I am making

Exercise 1.4: File Handling in Python

Learning Goals

- Use files to store and retrieve data in Python

Reflection Questions

1. Why is file storage important when you're using Python? What would happen if you didn't store local files? File storage is important in Python because it allows you to persist data across different sessions, making it possible to save, retrieve, and manipulate data even after the program ends. Without storing local files, any data generated or processed during runtime would be lost once the program terminates, requiring you to input or compute it again. This makes file storage essential for efficiency and for maintaining state between program executions.

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 serialized

representations of Python objects, created using the 'pickle' module, which allows complex data types like lists, dictionaries, or custom objects to be saved to a file and later retrieved. You would use pickles when you need to store Python objects in a way that preserves their structure, so they can be easily reloaded and used in future program executions. Pickling is particularly useful for persisting data that is not easily stored in text formats, such as machine learning models or large datasets with intricate relationships.

3. In Python, what function do you use to find out which directory you're currently in? What if you wanted to change your current working directory? In Python, you can use the os.getcwd() function to find out which directory you're currently in. To change the current working directory, you can use the os.chdir() function, passing the path to the new directory as an argument.

4. Imagine you're working on a Python script and are worried there may be an error in a block of code. How would you approach the situation to prevent the entire script from terminating due to an error? To prevent the entire script from terminating due to an error, you can use a 'try-except' block to catch and handle exceptions. This way, if an error occurs in the block of code, it will be caught, and you can define specific actions or messages for handling the error without stopping the program. Additionally, using the 'finally' clause ensures that certain cleanup actions (like closing files) happen regardless of whether an error occurs.

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. It feels like I'm making solid progress with the Python module, and despite the volume of material, I've been able to steadily work through it. However, one area where I feel I could benefit from more focused practice is writing functions. While I understand the basic concepts, I believe that improving my ability to design and implement more complex, reusable functions will help deepen my understanding of problem-solving in Python and enhance my overall coding efficiency.

Exercise 1.5: Object-Oriented Programming in Python

Learning Goals

- Apply object-oriented programming concepts to your Recipe app

Reflection Questions

1. In your own words, what is object-oriented programming? What are the benefits of OOP? Object-oriented programming (OOP) is a programming paradigm that organizes code into objects, which combine data (attributes) and behavior (methods). It encourages reusability, modularity, and encapsulation by grouping related functionality into classes. The benefits of OOP include

easier code maintenance, the ability to model real-world systems more intuitively, and greater flexibility through features like inheritance and polymorphism.

2. What are objects and classes in Python? Come up with a real-world example to illustrate how objects and classes work. In Python, a **class** is a blueprint for creating objects, defining the properties (attributes) and behaviors (methods) that the objects will have. An **object** is an instance of a class, representing a specific example with its own values for the class's attributes. For a real-world example, consider a **Car** class. The class defines attributes like 'make', 'model', and 'year', as well as methods like 'drive()' or 'stop()'. A specific object could be a **Car** instance representing "Toyota Camry 2020," with those attributes filled in for that car. Each object (car) is built from the same class blueprint but holds different values for its attributes.

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 fundamental OOP concept that allows a new class (called a child class or subclass) to acquire the properties and methods of an existing class (called a parent class or superclass). This enables code reuse, as the child class inherits attributes and behaviors from the parent class but can also extend or modify them. Inheritance promotes hierarchical relationships, where more specific classes build on the functionality of more general classes. For example, if you have a parent class Animal with methods like eat() and sleep(), a child class Dog can inherit those behaviors while also adding its own specific method, like bark(). The Dog class would inherit common functionality from Animal but can also have unique attributes, like breed. Inheritance simplifies code maintenance, as changes in the parent class automatically propagate to child classes. It also helps model real-world relationships, such as "a dog is a type of animal."
Polymorphism	Polymorphism in OOP allows objects of different classes to be treated as instances of the same superclass, even though each object may implement the shared method in its own way. This means "many forms," allowing different classes to respond to the same method call in ways that are appropriate to their specific behavior. Polymorphism can be achieved through method overriding (when a subclass modifies a method inherited from a parent class) or through interfaces that ensure different classes implement common methods. For example, if both Dog and Cat classes inherit from the Animal class and override the speak() method, calling speak() on a Dog object might return "Bark," while calling it

	on a Cat object might return "Meow." Even though the speak() method exists in both classes, each object behaves differently when the method is invoked. Polymorphism allows for more flexible and extensible code since you can interact with different objects through a common interface without knowing their specific class.
Operator Overloading	Operator overloading in Python allows custom classes to define how operators like +, -, *, and == behave when applied to objects of that class. It gives meaning to these operators beyond their default behavior when dealing with built-in data types, enabling objects to interact with each other in more intuitive ways. Python allows you to define how operators work on your objects by implementing special methods such as __add__() for + or __eq__() for ==. For example, if you create a class Vector to represent a mathematical vector, you can overload the + operator to add two vectors by adding their corresponding components. So instead of manually calling a method like v1.add(v2), you can simply use v1 + v2. This makes the code more readable and aligns with how operators are naturally used, leading to more intuitive interactions between objects. Operator overloading enhances code flexibility and expressiveness, particularly for classes that represent mathematical or logical entities.

Exercise 1.6: Connecting to Databases in Python

Learning Goals

- Create a MySQL database for your Recipe app

Reflection Questions

1. What are databases and what are the advantages of using them? Databases are organized collections of data that allow for efficient storage, retrieval, and management of information. They offer advantages such as fast data access, improved data integrity, and the ability to handle large amounts of data with complex relationships. In coding, using databases simplifies data persistence, promotes scalability, and enables multi-user access.

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

Data type	Definition
INT	Used to store whole numbers, both positive and negative, without decimal points.
VARCHAR	Stores variable-length text or alphanumeric data with a user-defined maximum length.
DATE	Stores calendar dates in the format YYYY-MM-DD for date-related information.

3. In what situations would SQLite be a better choice than MySQL? SQLite is a better choice than MySQL when you need a lightweight, serverless database for small applications or local development, as it requires no setup or configuration. It's ideal for mobile apps, embedded systems, or single-user desktop applications because it stores the entire database in a single file. Additionally, SQLite is useful when simplicity and portability are more important than performance and scalability.

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? JavaScript and Python differ in their use cases and syntax simplicity. JavaScript is primarily used for web development, enabling interactive functionality in browsers, while Python is known for its readability and is versatile, often used in data science, automation, and backend development. While JavaScript is event-driven and excels at handling asynchronous operations, Python's simplicity makes it easier to learn and better suited for rapid development across various domains.

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? Compared to JavaScript, Python is slower in execution since it's interpreted and not optimized for browser environments, where JavaScript excels in performance. While Python is better for data science, machine learning, and backend development, JavaScript's primary strength lies in its ability to handle asynchronous tasks and dynamic content for web applications. Additionally, JavaScript is more integrated into front-end development, making it the go-to language for creating interactive web interfaces, while Python lacks native browser support and is mainly used server-side.

Exercise 1.7: Finalizing Your Python Program

Learning Goals

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

Reflection Questions

1. What is an Object Relational Mapper and what are the advantages of using one? An Object Relational Mapper (ORM) is a programming technique that enables developers to interact with a relational database using an object-oriented paradigm, effectively bridging the gap between object-oriented programming and relational databases. ORMs automate the process of translating data between the database and application objects, allowing developers to work with database records as if they were native objects in their programming language. The advantages of using an ORM include increased productivity through reduced boilerplate code, enhanced maintainability by abstracting database interactions, and improved portability across different database systems.

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? Building the Recipe app was definitely challenging, as there was a lot of material to learn and apply, but I'm proud to say I got through it successfully. One thing I did well was using 'elif' statements to create smooth logic flows, making the app more efficient in handling user input. If I were to start over, I would focus on adding slight delays between actions and improving the text formatting to make it easier for users to follow and understand the instructions clearly.

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. In my experience creating a Python app, I developed a Recipe app using MySQL for database management and implemented various classes to structure the functionality. The app, which is over 300 lines of code, allows users to create, view, edit, delete, and search for recipes by ingredient. I used 'elif' statements and other control structures to handle user inputs effectively and ensure a smooth interaction with the database, creating a dynamic and user-friendly experience.

4. You've finished Achievement 1! Before moving on to Achievement 2, take a moment to reflect on your learning in the course so far:
    a. What went well during this Achievement?
    I felt very good about my pace during this Achievement. It was a alot of material but I was able to get through most of it by myself and on pace, not falling behind
    b. What's something you're proud of?
    I am proud of the app that I made and how there are no errors that are not covered in the if-else statements.
    c. What was the most challenging aspect of this Achievement?
    The most challenging aspect of this Achievement was creating the app without SQLalchemy for the first time.
    d. Did this Achievement meet your expectations? Did it give you the confidence to start working with your new Python skills?
    This Achievement met my expectations. I gives me the confidence to start working with my Python skills in other projects that I find interesting.
    e. What's something you want to keep in mind to help you do your best in Achievement 2?

I want to keep in mind that I will probably get more difficult, so I have to stay humble and keep my expectations low.

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? Using **vanilla Python** gives you full control over your code and design, allowing for maximum flexibility. However, this can lead to longer development times since you'll need to

implement everything from scratch, including features like authentication and routing. Additionally, without the built-in security measures provided by frameworks, you may face increased risks if you're not careful about securing your application.**Django** offers rapid development through its built-in features, which significantly speed up the process of building web applications. It also includes many security features by default, helping to protect against common vulnerabilities and allowing developers to focus more on functionality. On the downside, Django can have a steep learning curve for beginners, and its structure may feel restrictive for those seeking a high level of customization.

2. In your own words, what is the most significant advantage of Model View Template (MVT) architecture over Model View Controller (MVC) architecture? The most significant advantage of the Model View Template (MVT) architecture over Model View Controller (MVC) is its clear separation of concerns, especially in handling templates. In MVT, the template is responsible for the presentation logic, allowing developers to focus on how data is displayed without coupling it too tightly with the controller. This separation makes it easier to maintain and update the presentation layer independently from the business logic.

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?
      I want to learn how to deploy websites using Django that include, user authentication and are smooth and easy to use
    - What do you want to get out of this Achievement?
      I want to be confident in my skills as a Python web developer, able to create website for whatever it is I might want, using python
    - Where or what do you see yourself working on after you complete this Achievement?
      I see myself applying for a new career in web development. I am nervous about applying for jobs, but my expectations are low. I am just excited by the prospect of working for a tech for a company of any size

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.*) To convert a typical company website into Django terms, the entire website is represented as a single Django project, with project-wide settings and configurations. Within this project, separate sections of the website (like "Home," "About Us," "Products," and "Contact") would be represented as individual Django apps, each handling a specific functionality. For example, there could be an app for blog posts, an app for products, and an app for user accounts, making it easy to develop and maintain each section independently.

2. In your own words, describe the steps you would take to deploy a basic Django application locally on your system. To deploy a basic Django application locally, start by setting up a virtual environment and installing Django via pip install django. Next, create a Django project using django-admin startproject and create individual apps using python manage.py startapp for different parts of the site, like "home" or "blog." Finally, configure database settings, set up models, views, and templates, then run the server with python manage.py runserver to view the app locally and test its functionality.

3. Do some research about the Django admin site and write down how you'd use it during your web application development. The Django admin site is a built-in feature that provides a user-friendly interface to manage and inspect database records without coding. During development, it's used to quickly add, update, or delete data in each app's models, such as adding new blog posts or product entries. The admin site also makes it easy to review data relationships and test how new models and fields will appear, saving time by handling data management within the browser.

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 serve as the blueprint for creating and managing database tables in a Django project. Each model is a Python class that defines the fields and behaviors of the data you want to store, automatically translating Python code into database queries. This abstraction layer makes it easier to work with databases by providing built-in methods for data retrieval, creation, and updates, while maintaining clean and readable code.

2. In your own words, explain why it is crucial to write test cases from the beginning of a project. You can take an example project to explain your answer. Writing test cases from the start of a project ensures that your code is reliable and functions as expected as you develop new features. It helps catch bugs early, making them easier and cheaper to fix, and prevents regressions when making future changes. Additionally, having test coverage boosts confidence in the stability of the codebase and makes collaboration smoother for teams.

Exercise 2.4: Django Views and Templates

Learning Goals

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

Reflection Questions

1. Do some research on Django views. In your own words, use an example to explain how Django views work. Django views are Python functions or classes that handle web requests and return web responses. They act as the middle layer between the models (data) and templates (HTML). When a user requests a URL, Django uses the URL dispatcher to map the URL to the appropriate view, which then processes the request, interacts with the model if necessary, and renders a template to generate the final HTML response.
   Example
   ```
   from django.shortcuts import render
   from .models import Post

   def post_list(request):
       posts = Post.objects.all()  # Fetch all posts
       return render(request, 'blog/post_list.html', {'posts': posts})
   ```
   In this example, the post_list view retrieves all Post objects and passes them to the post_list.html template for rendering.

2. Imagine you're working on a Django web development project, and you anticipate that you'll have to reuse lots of code in various parts of the project. In this scenario, will you use Django function-based views or class-based views, and why? In a project where code reuse is essential, class-based views (CBVs) would be preferable over function-based views (FBVs). CBVs provide built-in methods for common actions (like CreateView for creating objects), which help reduce repetitive code and make views easier to manage and extend. They also support inheritance, making it easier to share and customize functionality across views, improving efficiency and maintainability in larger projects.

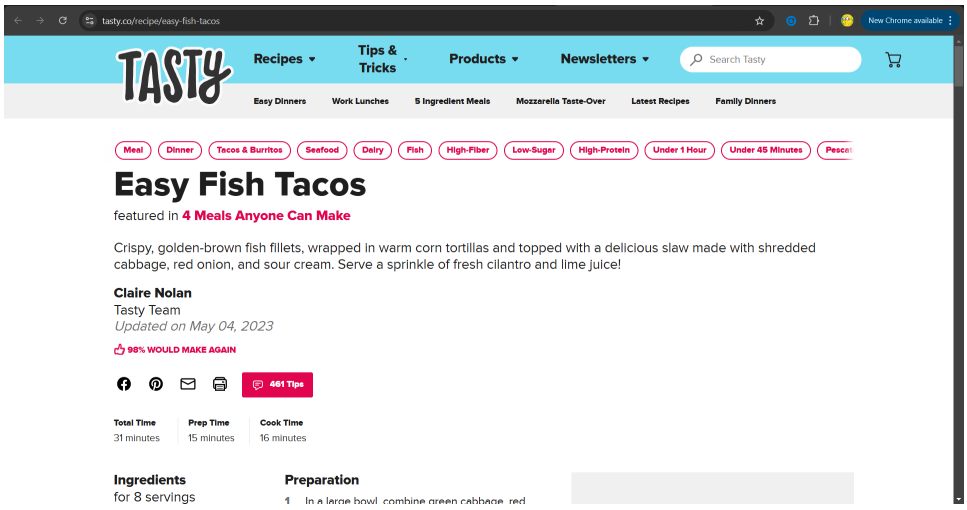
3. Read Django's documentation on the Django template language and make some notes on its basics. Django's template language allows you to dynamically generate HTML by embedding expressions, tags, and filters within templates. Template tags like {% for %} and {% if %} control logic in templates, while filters such as |lower or |date modify data formatting. Django templates also provide safe separation between presentation and logic, letting you include data in a controlled way without writing Python directly into HTML, which keeps code clean and secure.

Exercise 2.5: Django MVT Revisited

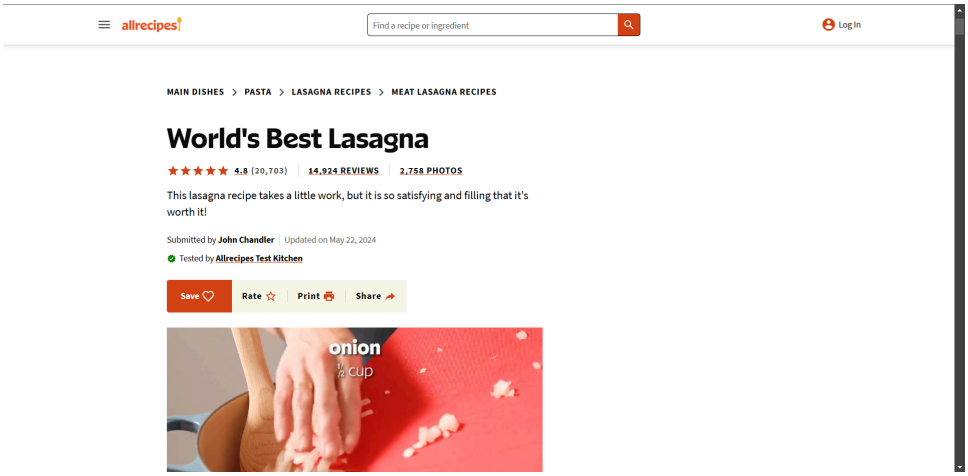
Learning Goals

- Add images to the model and display them on the frontend of your application
- Create complex views with access to the model
- Display records with views and templates

Frontend inspirations



Tasty - This website is easy to understand, there are no excessive ads. Many recipe website have their whole life story and many ads that slow it down excessively, but not Tasty.



Allrecipes - This website has a huge array of recipes to choose from, that are show on a good layout. The website is responsive and there is not a lot of junk crowding the website

Reflection Questions

1. In your own words, explain Django static files and how Django handles them. In Django, static files refer to assets like images, CSS, and JavaScript that don't change dynamically with each request. During development, Django can serve these files directly, but in production, they're often collected and served by a web server for efficiency. The 'collectstatic' command gathers all static files into a single location, making it easier for the server to handle them.

2. Look up the following two Django packages on Django's official documentation and/or other trusted sources. Write a brief description of each.

Package	Description
ListView	The ListView in Django is a generic class-based view designed to display lists of objects from the database. It automatically retrieves all instances of the specified model, renders them using a template, and offers features like pagination to manage large data sets. This view can be customized to filter, sort, or limit the displayed objects, making it a convenient option for list-based pages.
DetailView	The DetailView is another generic class-based view that displays details for a single object based on its primary key or unique identifier. When accessed, it retrieves the specified object and renders it in a template, simplifying the process of building detail pages for individual database

	entries. It is customizable for various display needs, making it effective for showing specific item details in an app.

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 material is becoming increasingly complex, and the reading has become more challenging to follow. I feel I need significantly more practice with these topics to gain a solid understanding. However, I am proud of how far I've come and my ability to tackle problems independently. Although I'm struggling with several concepts, I believe that continued practice will help me achieve a deeper comprehension.

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 essential in applications to verify the identity of users and protect sensitive data from unauthorized access. For example, in a banking app, it ensures that only the account holder can view their financial information or make transactions. This not only secures user data but also builds trust and prevents malicious activities.

2. In your own words, explain the steps you should take to create a login for your Django web application. To create a login in Django, start by configuring the built-in authentication system, which includes adding django.contrib.auth to your installed apps. Next, set up a login view using Django's LoginView or a custom view with the authenticate() and login() functions, and connect it to a URL pattern. Finally, create a simple template for the login form and define redirection behavior using settings like LOGIN_REDIRECT_URL.

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()	Validates user credentials and returns a User object if successful, or None otherwise, enabling custom login workflows.
redirect()	Sends a response to redirect users to another URL, often used after login, logout, or form submission.
include()	Used in urls.py to include URL configurations from other apps, making route management modular and organized.

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. My favorite application that I built was the **'Chat-App'**. I particularly enjoyed how seamlessly Google Firebase allowed me to store and manage data. One feature I'm excited about exploring further is analyzing the types of messages users send. For instance, if I notice that many users frequently share images, I could enhance the app by adding a "Save to Device" button, enabling users to easily download and keep their favorite images.

2. Read the Django official documentation on QuerySet API. Note down the different ways in which you can evaluate a QuerySet. In Django, a 'QuerySet' is lazily evaluated, meaning the database is not accessed until the results are explicitly needed. Evaluation occurs in various scenarios, such as when iterating over a 'QuerySet', slicing it, converting it to a list, or using it in a boolean context. Other triggers include calling 'len()', 'repr()', or serializing the 'QuerySet' with pickling, all of which load the data into memory. For optimized performance, methods like 'count()' and 'exists()' are recommended for tasks such as counting records or checking for their existence, as they avoid unnecessary data retrieval.

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. Django's QuerySet is optimized for database operations, allowing efficient querying and retrieval of data directly from the database. However, for complex data processing and analysis tasks, converting a QuerySet to a pandas DataFrame can be advantageous. DataFrame offers a rich set of functionalities for data manipulation, including handling missing data, merging datasets, and performing statistical operations. Additionally, pandas integrates seamlessly with other data science libraries, enhancing its capabilities for in-depth data analysis.

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. I styled my app using CSS, incorporating hover effects that make buttons lighten in color and adding a subtle bouncy animation to confirm user clicks. The content on each page is centrally aligned, creating a clean and visually appealing design. The user interface is intuitive, with no hidden or unclear controls, ensuring a seamless and user-friendly experience.
2. In your own words, explain the steps you'd need to take to deploy your Django web application.
    - To deploy your Django web application to Heroku, start by preparing your app for production by setting 'DEBUG = False' in 'settings.py', adding your Heroku domain to 'ALLOWED_HOSTS', and installing dependencies like 'gunicorn' and 'dj-database-url'.
    - Add a 'Procfile' to specify the app's startup commands, configure your database using 'dj-database-url', and set up static file management with 'whitenoise'. Install the Heroku CLI, log in, and initialize a Git repository if you haven't already.

- Create a Heroku app using 'heroku create' and set environment variables like 'SECRET_KEY' with 'heroku config:set'.
- Push your code to Heroku with 'git push heroku main', and then run 'heroku run python manage.py migrate' to apply database migrations.
- Finally, test your app at the Heroku-provided URL

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?
    During this achievement, I felt much more confident in my abilities and what I was able to accomplish. The material felt more intuitive, and concepts like writing functions, creating classes, and defining variables were much easier to grasp and implement in Python.
    b. What's something you're proud of?
    I'm proud of how much my confidence and understanding of programming concepts have grown during this achievement. Being able to effectively write functions, create classes, and define variables in Python felt natural and rewarding, showcasing how far I've come in my development journey.
    c. What was the most challenging aspect of this Achievement?
    The most challenging aspect of this achievement is having the images of the recipes show up on the website. I am still trying to get it to work but there are a ton of steps that I am not familiar with. The most challenging aspect of this achievement has been getting the images of the recipes to display on the website. It has been a bit difficult due to the many steps involved, some of which I'm not familiar with yet. I'm working through it, though, and trying to figure out how to set it up properly.
    d. Did this Achievement meet your expectations? Did it give you the confidence to start working with your new Django skills?
    I really enjoyed this achievement as it taught me how to code a web app using Python/Django. However, I wish the process had included guidance on how to display images on the hosted version on Heroku, since Heroku doesn't natively support serving images through Django. This would have been helpful in making the app fully functional when deployed.

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