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?

Ans: I have learned JavaScript and other programming languages both in this course and in the past. My most relevant experience includes working as a data analyst and studying Python at NTNU, where I gained foundational knowledge of the language. This prior experience gives me confidence that I can progress through this course more quickly and smoothly.

2. What do you know about Python already? What do you want to know?
   Ans: I already know that Python is one of the most popular programming languages today, supported by a highly active and motivated community. It offers numerous modules for a wide range of applications. I would like to learn more about various Python libraries and how to integrate them to build functional and impactful applications.
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.
   Ans: Time management will be a key challenge, as I need to balance my college studies, part-time job, and this course. To overcome this, I plan to dedicate more of my weekends to studying in the quiet and focused environment of my room. By creating a structured schedule and minimizing distractions, I believe I can stay on track and complete this course successfully.

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?
   Ans: Frontend development focuses on the visible and interactive parts of a website or application, such as buttons, forms, and layouts, which run on the user's browser. Backend development, on the other hand, deals with server-side operations, including database management, authentication, and application logic.

If I were hired as a backend programmer, my responsibilities would include creating and managing APIs, implementing secure authentication systems, managing databases to store and retrieve user data, and ensuring smooth server-side operations for the web application.

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")
   Ans:
   Both JavaScript and Python are powerful, high-level programming languages suitable for web development.
   JavaScript:
   a. Widely used for front-end development to create interactive user interfaces.
   b. Can also be used for backend development with Node.js.
   Python:
   a. Excels in backend development due to its simplicity, readability, and rich ecosystem of frameworks like Django and Flask.
   b. Ideal for projects involving complex data processing, machine learning, or AI integration.
   Advantages of Python for the project:
   a. Extensive libraries and frameworks significantly reduce development time.
   b. Excellent support for scalability and maintainability.
   c. Shorter learning curve, enabling new developers to onboard quickly and contribute effectively.

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?
   Ans:
   To review and strengthen my existing Python skills.
   To explore and gain deeper knowledge of Python tools and libraries.
   To learn how to effectively use Python for backend development.

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?
   Ans:
   a. Syntax Highlighting: iPython provides colored syntax, which makes it easier to read and understand the code by visually distinguishing keywords, variables, and strings.
   b. Automatic Indentation: iPython automatically handles indentation, which is crucial in Python since the language uses indentation to define loops, conditionals, and code blocks. This feature reduces errors and improves coding efficiency.
   c. Rich History Management: iPython allows you to access and search previous commands using the up arrow or the history command, making it easier to re-run or debug code.

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	Represents whole numbers (positive, negative, or zero)	Scalar
Tuple	An immutable sequence that can store multiple values of different data types	Non-Scalar
List	A mutable sequence that can store multiple values of different data types	Non-Scalar
Dictionary	A collection of key-value pairs for efficient data storage and retrieval	Non-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.
   Ans: The main difference between lists and tuples in Python is mutability. Lists are mutable, meaning their elements can be modified, added, or removed after creation. On the other hand, tuples are immutable, so their elements cannot be changed once defined. This makes tuples more memory-efficient and suitable for storing fixed data, while lists are more flexible and ideal 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.

Ans:

A dictionary is a good choice for storing individual vocabulary words, as it allows you to organize information using keys such as 'spelling,' 'definition,' and 'category.' This structure is flexible, making it easy to add additional elements to each vocabulary entry in the future, such as example sentences, synonyms, or other related details.

A list can then be used to store all the vocabulary dictionaries. Lists are efficient for organizing multiple entries and allow for easy indexing. This makes it convenient to randomly select vocabulary entries to create quizzes for users, enhancing the app's functionality.

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!)*

```
avaSites = ["Taiwan", "Toronto", "Vancouver" ]
site = input("Where do you want to travel: ")
if site in avaSites:
    print(f"Enjoy your stay in {site}")
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.
   Ans:
   Logical operators in Python are used to evaluate conditions involving Boolean values (True and False). There are three main logical operators:
   a. and: Returns True if both conditions are True.
   b. or: Returns True if at least one condition is True.
   c. not: Reverses the Boolean value; True becomes False and False becomes True

3. What are functions in Python? When and why are they useful?
   Ans: Functions in Python are blocks of reusable code that can be called and executed when needed. Functions can accept arguments, return values, or perform specific tasks. We use functions to break complex programs into smaller, manageable parts, making the code easier to maintain and debug. Additionally, functions help reduce redundancy by allowing us to reuse code multiple times.

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.
   Ans:
   a. To review and strengthen my existing Python skills => I have revisited data types and loops. Some of these concepts were familiar, so it felt like meeting old friends.
   b. To explore and gain deeper knowledge of Python tools and libraries => I learned about virtual environments in Python, which I find to be a practical and essential tool for development.
   c. To learn how to effectively use Python for backend development => I am looking forward to gaining more knowledge in this area as the course progresses.

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?

Ans: File storage is important because it allows you to save and reuse data, making your work more efficient. Without local file storage, users would need to input the same data every time, which is time-consuming and error-prone.

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?
   Ans: Pickles are serialized binary data created using Python's pickle module. They are useful for saving and loading complex data types like objects, images, or audio. Pickles make it easy to store and retrieve these data types for future use.

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?
   Ans: Use os.getcwd() to find the current directory. To change the working directory, use os.chdir().

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?
   Ans: Use a try-except structure to handle errors. Place the potentially error-prone code inside the try block and handle exceptions in the except block to prevent the script from crashing.

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.
   Ans: After pausing for a few weeks, I need to refresh my memory of the syntax. This is common when switching between tasks. To manage this better, I'm focusing on designing meaningful file names and adding clear comments to my code.

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?
   Ans:
   Object-oriented programming (OOP) is a programming paradigm that organizes a program into entities called objects. Each object represents a real-world concept and has its own attributes (data) and methods (functions) to perform actions.

The benefits of OOP become more apparent in larger programs, as it promotes modularity, code reusability through inheritance, and better manageability of complex systems. By defining similar entities as subclasses of a parent class, OOP allows for more efficient and structured development.
However, for small programs, OOP might not always be necessary or efficient, as its structure can feel overly complex for simple tasks.

2. What are objects and classes in Python? Come up with a real-world example to illustrate how objects and classes work.
   Ans:
   A class is a higher-level blueprint, while objects are instances of that class. For example, "Human" can be a class, and Anderson (me) is an object of the "Human" class. Anderson has features of the "Human" class, like standing, walking, and using tools. Similarly, Alex (my friend) is another object of the "Human" class, sharing the same features but having different ways of using tools or a different walking speed.
   This demonstrates polymorphism in OOP, where different objects of the same class can have variations in their behavior or 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 allows a class (child class) to inherit attributes and methods from another class (parent class). This promotes code reusability and creates a hierarchical relationship. For example, a Vehicle class might have methods like start() and stop(). A Car class can inherit these and add its own methods, like honk().
Polymorphism	Polymorphism let different classes can have the same attributes or method, but has different way to run. It lets methods in derived classes override or behave differently while maintaining the same interface. For example, a Shape class might have a draw() method. Subclasses like Circle and Square can implement their own versions of draw(), while the program can call draw() on any Shape object without knowing its exact type.
Operator Overloading	Operator overloading allows operators like +, -, and * to be redefined for custom behaviors in user-defined classes. For example, adding two Vector objects can be implemented using __add__() to make v1 + v2 return a new Vector object. This enhances code readability and makes custom objects behave more intuitively. (Magic Methods" or "Dunder Methods")

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?
   Ans:
   A database is a structure that contains logically related data in a single repository. It acts as a central data storage facility, which can be simultaneously accessed by multiple devices and users in real-time.
   [Advantages of using databases]
   1. Data Management:
      a. Elimination of data redundancy
      b. Elimination of data repetition
      c. Storing a great amount of data with little/no physical storage
   2. Operation & Decision Support:
      a. Supports day-to-day business operations
      b. Facilitates tactical & strategic decision-making
      c. Helps users get more information efficiently
   3. Data Sharing & System Benefits:
      a. Sharing of data among multiple users
      b. Security of data: Ensures that only authorized users have access
      c. Software independence, allowing different systems to interact with the data

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

Data type	Definition
INT	Stores integer values, typically used for counting or IDs.
VARCHAR(n)	Stores variable-length text with a maximum length of n. Suitable for names and short descriptions.
DATETIME	Stores date and time values in the format YYYY-MM-DD HH:MM:SS.

3. In what situations would SQLite be a better choice than MySQL?
   Ans:
   SQLite is a better choice in the following situations:
   - Small-scale applications: Suitable for mobile apps, embedded systems, and small web applications.
   - Local storage needs: Works well for applications that do not require a client-server database.

- Lightweight and easy setup: No need for a database server; just a single file.
- Lower resource consumption: Uses minimal system resources, making it ideal for simple applications.

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?
   Ans:
   - Syntax: Python has a cleaner, more readable syntax, while JavaScript has more complex rules.
   - Usage: JavaScript dominates web development, while Python excels in data science and backend development.
   - Execution: JavaScript runs in the browser, whereas Python usually runs on a server or local machine.

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?
   Ans:
   - Slower execution speed: Python is an interpreted language, making it slower than compiled languages like C++.
   - Memory consumption: Higher memory usage compared to lower-level languages.
   - Limited for high-performance applications: Not ideal for real-time systems or applications requiring extremely low latency.

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?
   Ans:
   Object Relational Mappers (ORMs) provide an abstraction layer that simplifies database interactions using object-oriented programming principles. They enhance code maintainability, security, and database portability while reducing the complexity of SQL queries. Additionally, ORM allows developers to interact with the database in a way that looks more like Python code, making it more intuitive and easier to integrate with the rest of the application.

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?
   Ans:
   The development process was a great learning experience in working with databases using SQLAlchemy ORM. It provided hands-on practice in designing, querying, and managing data while improving application structure.
   [Did well]
   1. Structured Code and Input Validation – The app ensures that user inputs are properly validated, preventing incorrect data entries and improving the overall reliability of CRUD operations.
   2. Effective Use of ORM – Using SQLAlchemy ORM made database interactions more readable and maintainable, reducing the need for raw SQL queries.

   [Could be improved]
   1. Searching Features – Allow users to search by multiple ingredients with AND/OR options for more flexible filtering might be a good idea
   2. DRY (Don't Repeat Yourself) – Some parts of the code repeat similar logic, and refactoring could help make the implementation more concise and maintainable.

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.
   Ans:
   I developed a recipe management app using Python and SQLAlchemy ORM, which allows users to create, view, search, edit, and delete recipes from a database. I focused on structured code design, input validation, and efficient database interactions, ensuring smooth CRUD operations. Additionally, I implemented search functionality and user-friendly prompts to enhance usability.

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?
      Ans:
      Learning python and using mySQL
   b. What's something you're proud of?
      Ans:
      1. Building a fully functional, database-driven app from scratch.
      2. Writing clean, structured, and maintainable Python code.
   c. What was the most challenging aspect of this Achievement?
      Ans:
      Handling user input validation while keeping the app user-friendly.
   d. Did this Achievement meet your expectations? Did it give you the confidence to start working with your new Python skills?
      Ans:
      Yes, it did. Python is easy to write and understand, and this achievement provided me with fundamental knowledge of working with databases.
   e. What's something you want to keep in mind to help you do your best in Achievement 2?
      Ans:

Focus on writing modular and reusable code to make future enhancements easier.
Continue practicing good debugging techniques to troubleshoot issues efficiently.

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?
  Ans:
  Yes, it was effective. My structured approach to learning Python and MySQL, along with hands-on practice, helped me grasp the concepts well.
  I will focus more on writing modular and reusable code to make enhancements easier and continue practicing debugging techniques to troubleshoot issues efficiently.
- 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?
  Ans:
  Successfully building a fully functional, database-driven app from scratch.
  I will continue to learn new framework and integrate them to practice.
- 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?
  Ans:
  It took time to complete all the exercises due to other life and study commitments. I tried to manage my time more effectively to ensure I finished all the exercises. Moving forward, I will continue improving my time management and aim to complete tasks as soon as possible to successfully achieve the final goal.

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?
   Ans:
   [advantages of vanilla python]
   1. Flexibility: No predefined structure, allowing for complete customization.
   2. Lightweight: No extra dependencies, making it suitable for small projects.
   3. Fine-grained control: You can build everything from scratch according to specific project needs.

   [drawbacks of vanilla python]
   1. Time-consuming: Requires building authentication, database handling, and routing manually.
   2. Scalability issues: Managing a growing codebase without a framework can become complex.
   3. Security concerns: Needs manual implementation of security measures like CSRF protection.

   [advantages of Django]
   1. Rapid development: Comes with built-in features like authentication, ORM, and security measures.
   2. Scalability: Designed to handle large applications efficiently.
   3. Security: Protects against common vulnerabilities (e.g., SQL injection, CSRF, XSS).
   4. Community support: Extensive documentation and a large developer community.

   [drawbacks of Django]
   1. Learning curve: Can be complex for beginners.
   2. Opinionated structure: Limits flexibility by enforcing a predefined architecture.
   3. Overhead for small projects: Might be overkill for simple applications.

2. In your own words, what is the most significant advantage of Model View Template (MVT) architecture over Model View Controller (MVC) architecture?
   Ans:
   The biggest advantage of MVT over MVC is that Django automatically manages the "Controller" part, allowing developers to focus on building the Model (database logic) and Template (frontend). This reduces boilerplate code and speeds up development.

3. Now that you've had an introduction to the Django framework, write down three goals you have for yourself and your learning process during this Achievement. You can reflect on the following questions if it helps:
    - What do you want to learn about Django?
    - What do you want to get out of this Achievement?
    - Where or what do you see yourself working on after you complete this Achievement?

    Ans:
    1. Improve understanding of Django's architecture
    2. Build a functional web application using Django
    3. Develop skills for real-world Django development

Exercise 2.2: Django Project Set Up

Learning Goals

- Describe the basic structure of a Django project
- Summarize the difference between projects and apps
- Create a Django project and run it locally
- Create a superuser for a Django web application

Reflection Questions

1. Suppose you're in an interview. The interviewer gives you their company's website as an example, asking you to convert the website and its different parts into Django terms. How would you proceed? For this question, you can think about your dream company and look at their website for reference.
   (*Hint: In the Exercise, you saw the example of the CareerFoundry website in the Project and Apps section.*)
   Ans:
   We can consider THE WALL STREET JOURNAL as an example. (https://www.wsj.com/)
   This website includes many apps.
    1. news – Manages news articles, categories, and tags for displaying the latest news and filtering by category.
    2. subscriptions – Handles user subscriptions, payments, and access management for premium content.
    3. users – Manages user authentication, profiles, and saved articles.
    4. opinion – Hosts editorial and opinion pieces, categorized by contributors.
    5. markets – Provides real-time financial market data, stock trends, and company insights.

2. In your own words, describe the steps you would take to deploy a basic Django application locally on your system.
   Ans:
   1. Set Up a Virtual Environment
   2. Install Django
   3. Create a Django Project
   4. Apply Database Migrations
   5. Run the Development Server
   6. Create a Superuser (Optional for Admin Panel)

3. Do some research about the Django admin site and write down how you'd use it during your web application development.
   Ans:
   The Django Admin Site is a built-in interface for managing database records during development. It allows developers to easily create, update, and delete data without writing SQL queries. I would use it to manage users, permissions, and content, making testing and debugging more efficient. By registering models in admin.py, I can customize the admin panel with search, filters, and display options. Accessing it via /admin/ after creating a superuser provides a convenient way to control application data, improving development speed and accuracy.

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.
   Ans:
   Django models act as a bridge between Python code and the database, allowing developers to define database tables using Python classes instead of raw SQL. Each model corresponds to a table, and its attributes represent columns. Django's ORM (Object-Relational Mapping) enables developers to interact with the database using Python methods instead of writing complex queries. The benefits of using Django models include database abstraction, automatic table creation through migrations, built-in data validation, security against SQL injection, and scalability.

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.
   Ans:
   Writing test cases from the beginning of a project is crucial because it helps catch bugs early, ensures code stability, and makes debugging easier. Tests verify that each feature works as expected, reducing the chances of introducing errors when modifying the codebase. For example, in a recipe application, testing the creation of recipes can prevent issues such as saving recipes without required fields.

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.
   Ans:
   Django views handle HTTP requests and return responses, acting as the link between a user’s request and the data or templates the application provides. When a user visits a URL, Django routes the request to a specific view function, which processes the request and returns a response such as an HTML page, JSON data, or a redirect.

   For example, a simple function-based view can return an HTML page:

   ```
   from django.shortcuts import render
   def home(request):
       return render(request, 'home.html')
   ```

   When a user visits /, Django calls the home function, which loads and returns home.html. Alternatively, Django provides class-based views (CBVs) like ListView, which automatically retrieves and displays database records, reducing repetitive code. Views allow developers to control the logic of their web applications while keeping the backend and frontend separate.

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?

Ans:
If I anticipate needing to reuse a lot of code in a Django project, I would use class-based views (CBVs) instead of function-based views (FBVs). CBVs provide built-in generic views, such as ListView, CreateView, UpdateView, and DeleteView, which handle common tasks like retrieving, displaying, and modifying database records with minimal code duplication, and they are easy to be reused.

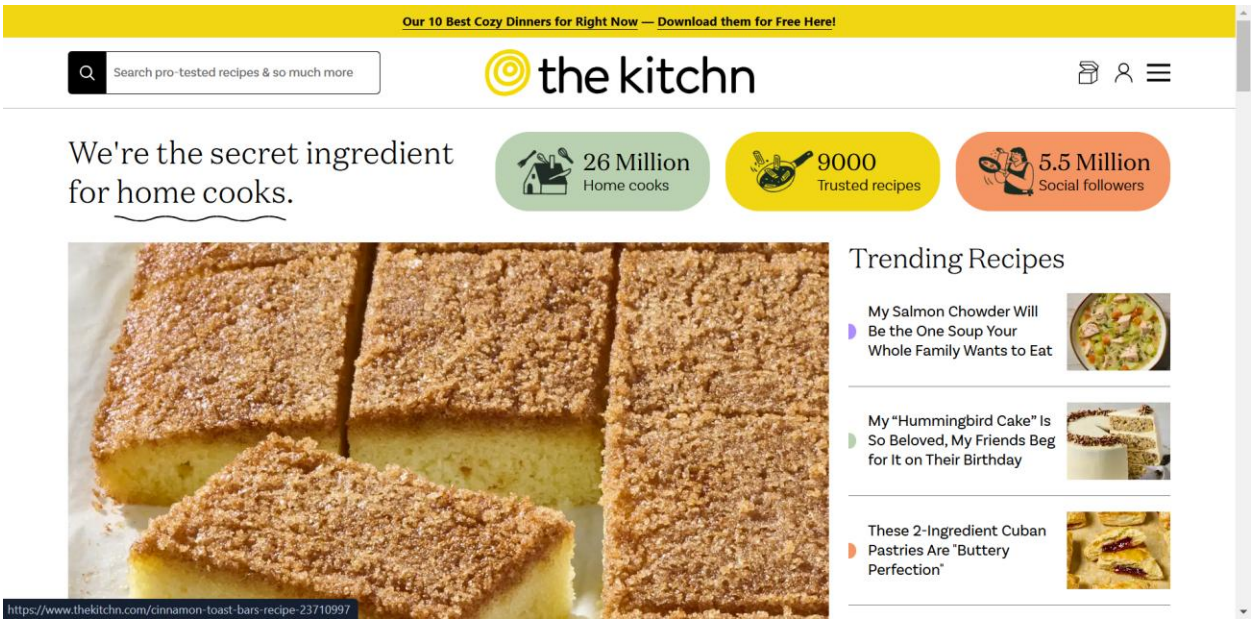
3. Read Django's documentation on the Django template language and make some notes on its basics.
   Ans:
   Django's template language (DTL) is a system for dynamically generating HTML content using template tags, filters, and template inheritance. It allows separating logic from presentation, making templates more readable and maintainable.

Exercise 2.5: Django MVT Revisited

Learning Goals

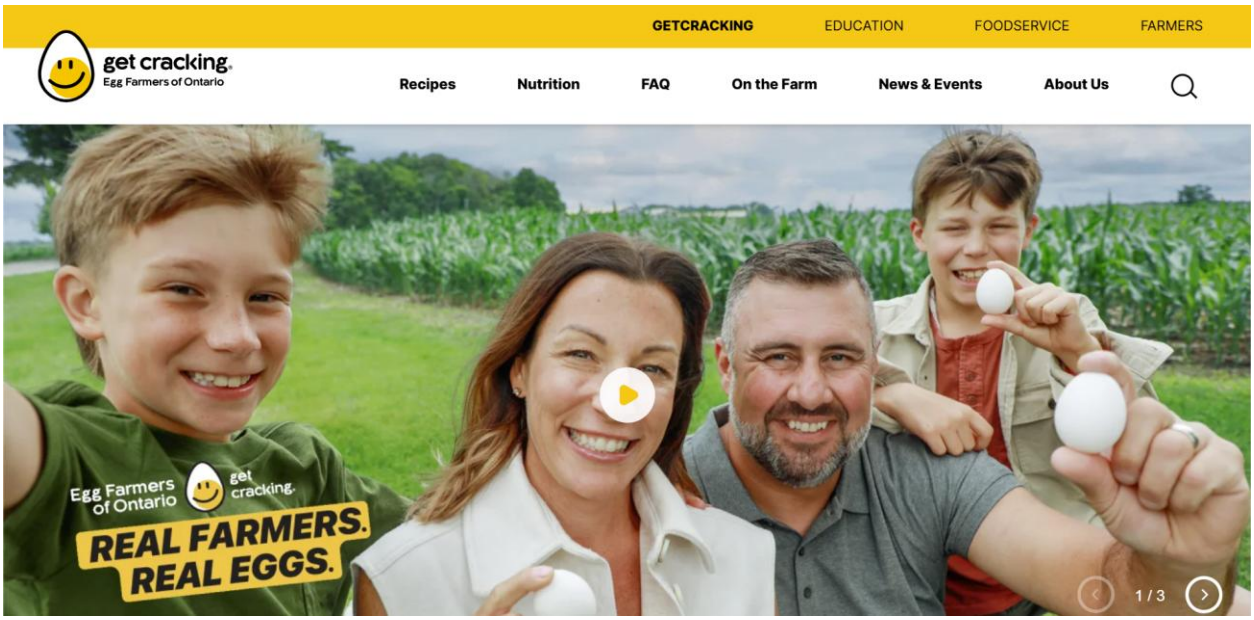
- Add images to the model and display them on the frontend of your application
- Create complex views with access to the model
- Display records with views and templates

Frontend Inspirations

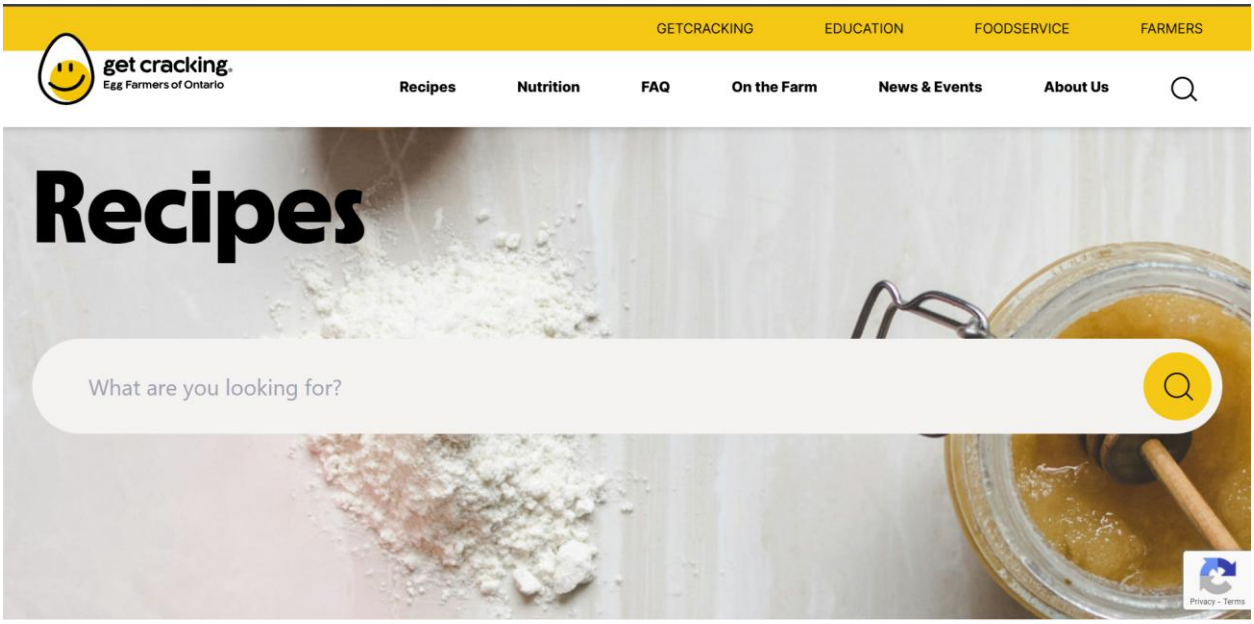
1. [The Kitchn] - https://www.thekitchn.com/
2. 

The top section highlights special recipes, possibly the most popular ones, while the bottom section displays a complete list of all recipes and blogs. It's a great idea.

3. [Get Cracking] - https://www.getcracking.ca/egg-recipes?gad_source=1&gclid=Cj0KCQiAoJC-BhCSARIsAPhdfShxYnyVYjfC20YRe75v4RoNrDR-1xBmjYSaGDM-1-Zj3lVHxceMyeoaAm6vEALw_wcB



This website is related to an egg farm, so it puts some introduce video and other information in the beginning. It's a good idea if you have something you want to mention like some articles or ads.



This page has a big filter on the top. It makes sense because people visit a recipe website to find specific recipes or ingredients.

Reflection Questions

1. In your own words, explain Django static files and how Django handles them.
   Ans:
   Django static files are non-dynamic assets like CSS, JavaScript, and images used to enhance the frontend. Django manages them by defining static file locations (STATIC_URL, STATICFILES_DIRS), serving them automatically in development, and collecting them into a single directory for production using collectstatic. In templates, static files are referenced using the {% static %} tag, ensuring efficient organization and deployment.

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	A generic class-based view that displays a list of objects from a specified model. It automatically retrieves and paginates querysets, reducing the need for manual query handling in views.
DetailView	A generic class-based view used to display a single object's details. It fetches the object based on a primary key or slug and renders it in 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.
   Ans:
   [Something I'm proud of]: I successfully created the ingredients view and used queries to connect multiple models, which felt like a big achievement.
   [Something I'm struggling with]
   - Connecting different models and resolving related issues wasn't easy.
   - Debugging those connections took a lot of time and effort.

   [What I need more practice with]
   - Getting more comfortable with Django's structure and best practices.
   - Understanding and setting up URLs more efficiently.

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.
   Ans:
   Authentication ensures the security of an application by preventing unauthorized users from accessing sensitive information. For example, in an e-commerce website, only authenticated users should be able to view their shopping cart, place orders, or manage personal information. This helps protect user data and enhances security.

2. In your own words, explain the steps you should take to create a login for your Django web application.
   Ans:
   a. Set up Django's user model, either the built-in User model or a custom model.
   b. Create a login form, using Django's AuthenticationForm or a custom forms.Form.
   c. Handle login in views.py, using authenticate() and login() functions.
   d. Define login URLs in urls.py, pointing /login/ to the login view.
   e. Use Django's built-in LoginView to simplify the login process.
   f. Protect views with LoginRequiredMixin or @login_required to restrict unauthorized access.

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()	Verifies user credentials (e.g., username, password). Returns a User object if valid, otherwise returns None.
redirect()	Redirects the request to a specified URL, e.g., redirect('home') navigates to the home page.
include()	Used in urls.py to include URL patterns from other apps, making it easier to manage multiple Django applications.

Exercise 2.7: Data Analysis and Visualization in Django

Learning Goals

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

Reflection Questions

1. Consider your favorite website/application (you can also take CareerFoundry). Think about the various data that your favorite website/application collects. Write down how analyzing the collected data could help the website/application.

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

3. In the Exercise, you converted your QuerySet to DataFrame. Now do some research on the advantages and disadvantages of QuerySet and DataFrame, and explain the ways in which DataFrame is better for data processing.

Exercise 2.8: Deploying a Django Project

Learning Goals

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

Reflection Questions

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

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

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

4. You've now finished Achievement 2 and, with it, the whole course! Take a moment to reflect on your learning:
    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.