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 just completed JS course, which I believe can help me understand the Python quicker.*

1. **What do you know about Python already? What do you want to know?**  
   *I know the basic syntaxes and strings, know the value can be updated directly in the code – last line of code defines the updated value. I also learned basics through Udemy courses and still learning it. I want to know how to make a solid backend code for games and applications, so it will be easy to read and understand.*
2. **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.**

*The main challenge for me is a lack of complete understanding the task due to new material or poor explanation. For example, in this exercise stated in several places that I need to have a Python version 3.8.7. It does not say this one or newer. I had the last version installed, then spent so much time for reinstalling it and installing the right one. All to find out at the end that the last version I usable as well. Now I have to type Phyton3 –version, not python –version in order to see what python I do have installed. This makes very angry.*

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

### Exercise 1.1: Getting Started with Python

#### Learning Goals

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

#### Reflection Questions

1. **In your own words, what is the difference between frontend and backend web development? If you were hired to work on backend programming for a web application, what kinds of operations would you be working on?**  
   *Simply put, the main difference between the frontend and backend is that the frontend is what the user sees: the app, design, pictures, graphics, and buttons. In contrast, the backend is the behind-the-scenes code that processes and supports the commands from the frontend. As a backend programmer for a web application, I would work on database management, server-side logic, authentication, API integrations, and ensuring data security.*
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”)***

*Both JavaScript and Python are versatile, high-level programming languages, but they have distinct uses and strengths. JavaScript excels in front-end web development, enabling dynamic and interactive web pages. Python, on the other hand, is known for its simplicity, readability, and vast libraries, making it ideal for backend development, data analysis, and machine learning.*

*To convince the team to use Python, I'd highlight its ease of learning and use, which speeds up development and reduces bugs. Python’s extensive libraries and frameworks, like Django and Flask, simplify backend development. Additionally, Python’s strong support for data processing and integration with other technologies makes it a versatile choice for comprehensive projects.*

1. **Now that you’ve had an introduction to Python, write down 3 goals you have for yourself and you’re 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?**

*My goals will be straightforward:*

*1. Master Python Fundamentals: I want to gain a solid understanding of Python's syntax, data structures, and core programming concepts to build a strong foundation.*

*2. Build Real-World Applications: I aim to develop practical skills by creating real-world applications, such as a command-line Recipe app, which will enhance my coding experience and problem-solving abilities.*

*3. Prepare for Future Projects: I want to be well-prepared to work on complex web applications and data analysis projects, using Python's powerful libraries and frameworks, after completing this Achievement.*

### Exercise 1.2: Data Types in Python

#### Learning Goals

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

#### Reflection Questions

1. **Imagine you’re having a conversation with a future colleague about whether to use the iPython Shell instead of Python’s default shell. What reasons would you give to explain the benefits of using the iPython Shell over the default one?***The IPython Shell provides a more interactive - user friendly, productive, and feature-rich environment compared to Python’s default shell, making it a preferred choice for many developers, especially those involved in data science, research, and interactive computing tasks.*
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 | A whole number (positive or negative) can use math functions with this; no theoretical limit | scalar |
| float: decimal number | A decimal number (positive or negative) can also use math with this; has a limit | scalar |
| str: string | A string (number or letters) cannot use math functions with a string of numbers. Adding would just mean the numbers are concatenated | non-scalar |
| bool: boolean | Only two values: true/false. Can be used to check outputs of a condition | scalar |

1. **A frequent question at job interviews for Python developers is: what is the difference between lists and tuples in Python? Write down how you would respond.***The choice between lists and tuples depends on whether you need a mutable sequence (use a list) or an immutable sequence (use a tuple) in your Python program. Understanding these differences allows you to select the appropriate data structure for your specific use case, optimizing both performance and functionality.*
2. **In the task for this Exercise, you decided what you thought was the most suitable data structure for storing all the information for a recipe. Now, imagine you’re creating a language-learning app that helps users memorize vocabulary through flashcards. Users can input vocabulary words, definitions, and their category (noun, verb, etc.) into the flashcards. They can then quiz themselves by flipping through the flashcards. Think about the necessary data types and what would be the most suitable data structure for this language-learning app. Between tuples, lists, and dictionaries, which would you choose? Think about their respective advantages and limitations, and where flexibility might be useful if you were to continue developing the language-learning app beyond vocabulary memorization.**

*This language-learning app would be built best with a dictionary. The data of the flashcards would be organized in key/value pairs, with the words being the keys, and the definitions and categories as the values. Dictionaries are also mutable, so the users would be able to easily add and remove the cards.*

### Exercise 1.3: Functions and Other Operations in Python

#### Learning Goals

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

#### Reflection Questions

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

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

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

|  |
| --- |
| def travel\_destination():  destination = input("Where to?")  if destination == "Bali":  print("Enjoy your stay in Bali!")  elif destination == "Banff":  print("Enjoy your stay in Banff!")  elif destination == "Cairo":  print("Enjoy your stay in Cairo!")  else:  print("Oops, that destination is not currently available.")  travel\_destination() |

1. **Imagine you’re at a job interview for a Python developer role. The interviewer says “Explain logical operators in Python”. Draft how you would respond.**

*Logical operators help us decide if conditions are true or false. They are also called “Boolean operators” and include “and”, “or”, and “not”. We use these operators to check conditions before moving forward in a program.*

Here’s a quick look at each one:

• and: Both conditions must be true for the result to be true.

• or: At least one condition must be true for the result to be true.

• not: Reverses the truth value; true becomes false and false becomes true.

A truth table can help us see how these operators work by showing all possible combinations of true and false conditions and their results.

For example:

• **and:**

• true and true = true

• true and false = false

• false and true = false

• false and false = false

• **or:**

• true or true = true

• true or false = true

• false or true = true

• false or false = false

• **not:**

• not true = false

• not false = true

*Using these operators, we can easily check if certain conditions are met before continuing with the rest of our code.*

1. **What are functions in Python? When and why are they useful?***Functions are a set of instructions that your script uses to do something with your code for some purpose (to put if vaguely as the question itself is pretty open-ended). Functions are useful as they can be reusable throughout your code, where more functions or operators can call on them in multiple other places. Functions can also save space and time while coding as you can plug in different values at different points to generate results. For example, looking at our recipe app, the same function can be called to make a recipe, but each time the recipe would use different ingredients (value inputs).*
2. **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.**

*It's still fairly early in the course, but I have accomplished parts of my goals so far:*

*- I've improved some of how I work in my current job, knowing now how to quickly test parts of a python script by using iPython.*

*- In this exercise though, I've been able to write staple Python functions from scratch*

### 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 a crucial part of applications that need to read data from files and write data to files. Without file storage, any data that is produced using an earlier part of your code will be lost. With files stored, the application is able to continue using any data that is created at any time -- even after the application is closed.*
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 a way of storing data, but instead of storing data as human-readable text, data is stored as bytes written to a binary file that only the computer can read.*

*You would use pickles if you are dealing with more complex data, such as data in a dictionary. The pickle will store the data in the binary file so that you could retrieve it with its original structure preserved.*

1. **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?**  
   *- Find out which directory you're currently in:*

*print(os.getcwd())*

*- Navigate to a different directory:*

*os.chdir("<path to target dir>")*

1. **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?**  
   *Using error handling in your code will allow you to see an error that the script encountered without crashing the entire script. A specific way to do this to ensure the script does not terminate is by using a try-except block.*

*A try-except block can 'try' a block of code where there could be a probable error. If there is no error, the script continues as normal. If there is an error, the 'except' block notifies the user of the error and can suggest a fix for it. Your entire script will not be terminated if an error is found in a try-except block.*

1. **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.***So far, I think everything is going really well! I'm enjoying the content and feel like I'm learning a lot. It's getting me excited about potentially creating different apps with Python.*

*Right now, I might be struggling a bit with time management. I tend to spend a lot of time trying to format things and make them appear a certain way, but I also recognize that Achievement 1 might not need so much focus on appearance.*

*I will say I could use some more practice with writing common functions - similar to those that might be part of an interview process at some point.*

### 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?**  
   *OOP is a way of organizing code in a readable and logical way. It's a programming model where code is organized into objects that are contained in a class. The focus is to create reusable and modular code. OOP is beneficial is it helps to keep code non-repetitive, non-redundant, and efficient. It follows DRY principles, meaning "Don't Repeat Yourself".*
2. **What are objects and classes in Python? Come up with a real-world example to illustrate how objects and classes work.***`Objects` are containers that hold attributes (data) and methods (attributes). The attributes in an object can be any data type. Objects are an instance of a class, which is the template that defines the structure of the objects, what they do, and how they interact with each other.*

*One example could be a fitness tracker app. You can create a FitnessTracker class, which could have attributes like user\_name, user\_weight\_, and step\_goal.*

*Individual objects (like tracker1 and tracker2) can be created from this class template. For example:*

**tracker1 = FitnessTracker("JohnDoe", 185, 10000)**

**tracker2 = FitnessTracker("JaneDoe", 142, 8000)**

*Each of these is an instance of the FitnessTracker class.*

1. **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 to "inherit" attributes and methods from another class (a "subclass" inheriting from a "superclass") |
| Polymorphism | A concept that allows objects of different subclasses to be treated as objects of a superclass. The goal here is to make code generic and non-repetitive. Methods/functions are able to work on objects of multiple classes that share the same behavior. |
| Operator Overloading | A feature of OOP that allows you to define custom behaviors for standard operators (+, -, \*, /, etc.). Operators can behave differently based on different datatypes. For example, `+` acts as an addition operator, or a concatenate operator. A *TypeError* would be thrown if you tried to use the `+` operator on a custom class. Operator overloading can avoid this by defining what you want the operator to do when it's applied to the objects of your class. |

### Exercise 1.6: Connecting to Databases in Python

#### Learning Goals

* Create a MySQL database for your Recipe app

#### Reflection Questions

1. **What are databases and what are the advantages of using them?**  
   *Databases are organized collections of data, that makes accessing and managing data more efficient. Using them in coding like Python enhances data integrity, consistency, security, and enables complex searching/querying.*
2. List 3 data types that can be used in MySQL and describe them briefly:

|  |  |
| --- | --- |
| **Data type** | **Definition** |
| VARCHAR(n) | String of variable length, with n representing the maximum number of characters |
| INT | Standard integers; stores whole numbers (without decimal points) |
| DATETIME | used to store date and time values. It holds both the date and time components, allowing you to store specific moments in time |

1. **In what situations would SQLite be a better** **choice than MySQL?**  
   *SQLite might be preferable over MySQL for small-scale projects, embedded systems, or applications where simplicity, low resource usage, and single-user access are key. MySQL is better suited for larger-scale applications with multiple users, higher concurrency, and advanced features.*
2. **Think back to what you learned in the Immersion course. What do you think about the differences between JavaScript and Python as programming languages?**  
   *Python is much easier to read and it is very forgiving, thus making it seem easier to write than JavaScript. Even though it's easier to understand than JavaScript, it still remains an incredibly powerful language. The neat part about JavaScript, however, is the ability to work on both front and backend environments, while Python is made more for backend.*
3. **Now that you’re nearly at the end of Achievement 1, consider what you know about Python so far. What would you say are the limitations of Python as a programming language?**  
   *Python is relatively slower in execution speed compared to compiled languages and its Global Interpreter Lock (GIL), which can hinder performance in multi-threaded applications. Additionally, its mobile and frontend development capabilities are somewhat limited compared to languages like Java or JavaScript.*

### Exercise 1.7: Finalizing Your Python Program

#### Learning Goals

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

#### Reflection Questions

1. **What is an Object Relational Mapper and what are the advantages of using one?**  
   An Object-Relational Mapper (ORM) is like a translator between databases and programming languages. It helps developers store and retrieve data from databases using normal code instead of complex database language. This makes coding faster, cleaner, and helps avoid mistakes.
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?**Overall I think creating this app went pretty well. Although challenging and a bit time consuming, it was helpful to "recreate" the app several times but in different ways. This last exercise, 1.7, was the most enjoyable as I had a better understanding of what was expected, and object relational mapping is just a very efficient way of storing data. If I could to start over, I'd want to spend more time on just one "version", such as this last exercise. It would be nice to get more creative and try to make different features/play with the display more.
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 a recent project, I developed a command-line version of a Recipe app using Python, which served as a precursor to a web app counterpart. This project allowed me to dive deep into Python fundamentals, data structures, object-oriented programming, and database interaction.

1. You’ve finished Achievement 1! Before moving on to Achievement 2, take a moment to reflect on your learning in the course so far:
   1. **What went well during this Achievement?**I was able to get a pretty firm grasp on basic coding skills with Python. Though I have a bit of experience prior, there was always plenty I had not previously known about.
   2. **What’s something you’re proud of?**  
      After writing out the basic structure of my code, I enjoyed going back and refactoring. I think I did well with keeping my code simple, finding ways to shorten the length of functions, without affecting the readability.
   3. **What was the most challenging aspect of this Achievement?**Balancing the various tasks and ensuring steady progress towards milestones required effective time management. Applying OOP principles to create modular and reusable code was challenging.
   4. **Did this Achievement meet your expectations? Did it give you the confidence to start working with your new Python skills?**Yeah, I'm very excited to move on to the next achievement where I can make my code come to life more than just in a terminal.
   5. **What’s something you want to keep in mind to help you do your best in Achievement 2?**I think planning is a key here. Before each achievement even starts, I do read preparation to it and creating a folders, repositories and README files on GitHub. This allows me to be familiar with the project and to get done 30% of the work. After that comes consistency. As they say: "Motivation is temporary, but consistency is lasting—provided you maintain it through continuous effort."

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

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

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

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

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

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

### Exercise 2.1: Getting Started with Django

Learning Goals

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

#### Reflection Questions

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

***Advantages***

- flexibility, complete freedome to build the application exactly how you want without predefined structure

- lightweight, no overhead from a framework, which would be suitable for a smaller projects

- little learning curve if you're already familiar with Python

***Drawbacks***

- not very scalable as projects grow

- potentially time consuming, as you'd need to handle many aspects yourself, such as setting up the server, managing databases, and implementing security measures

Django:

***Advantages***

- rapid development, as pre-built components are provided for common tasks, such as user authentiation, DB management, and form handling

- active community, extensive documentation, and a wide range of third-party packages can help speed up development troubleshooting

***Disadvantages***

- learning curve, if you're not already familiar with the framework

- less flexibility, as it comes with a lot of pre-built tools and some conventions and restrictions that might limit flexibility in certain cases

- some overhead in using a framework, as it might include features you don't need for a specific project.

1. **In your own words, what is the most significant advantage of Model View Template (MVT) architecture over Model View Controller (MVC) architecture?**  
   A significant advantage of the Model View Template (MVT) architecture over the traditional Model View Controller (MVC) architecture lies in the distinct separation of the application's tasks without getting mixed up with each other. In other words, MVT pays a lot of attention to making sure the design and appearance of the app (the template) are kept separate from the data and technical attributes (the model and view). This separation keeps things organized and easier to work on. In contrast, in the MVC architecture, the view often includes both presentation logic and some level of backend interaction, which can lead to more complex and coupled code.
2. **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?

- Learn how to create and manage data models, build views that handle user requests, and design a template for displaying information to the user

- Gain proficiency in Django's framework to perform CRUD operations, allowing data to be created, read, updated, and deleted

- Learn the ins and outs of using Django's features (and likely additional third-party tools) to create an efficient, easy-to-use, and aesthetically pleasing user interface.

### 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.*)**  
Imagine we have a website that sells outdoor gear. In Django, the whole website is called a "project." Inside this project, we have smaller parts called "apps" for different features, like:

Customer Login: A part of the site where users can log in.

Available Products: A part where users can browse and buy products.

Newsletters: A part for users to sign up for updates and news.

Now, suppose the same company has another website for seasonal merchandise and local events. This new website can reuse the existing apps like "Customer Login" and "Available Products" because it also needs these features. Additionally, this new site might have a new app specifically for "Upcoming Events."

So, in simple terms:

The project is the whole website.

The apps are the different parts or features of the website.

Apps can be reused in other projects if they need the same features.

1. **In your own words, describe the steps you would take to deploy a basic Django application locally on your system.**  
   Step 1 - Run migrations. This step creates the necessary DB tables for your project to initiate the backend environment. In Windows, run `py manage.py migrate`.

Step 2 - Run the server. This step deploys the project to localhost. The developer can access this application through their browser from the weblink that is provided after running this step. You can create a super user that will be able to perform all CRUD operations (create, read, update, delete).

1. **Do some research about the Django admin site and write down how you’d use it during your web application development**.  
     
   Using the Django admin site can significantly simplify the development and management of the web application. It provides an out-of-the-box user interface for managing the application's data, which is especially useful during development and for non-technical users.

Managing data - I'd be able to use the admin interface to add, edit, and delete data. I could have a user-friendly form for adding new recipe entries, updating existing ones, and even searching for specific records.

Permissions and Access - (for multiple users) I could define user groups, assign specific permissions to different user roles, and control who can perform any of the CRUD operations.

### Exercise 2.3: Django Models

#### Learning Goals

* Discuss Django models, the “M” part of Django’s MVT architecture
* Create apps and models representing different parts of your web application
* Write and run automated tests

#### Reflection Questions

1. **Do some research on Django models. In your own words, write down how Django models work and what their benefits are.**  
     
   Django models are a fundamental part of the Django web framework and provide an abstraction for defining the structure and behavior of your application's data. They simplify database management, improve code organization, and enhance development productivity by providing a Pythonic way to define, query, and interact with one's data.
2. **In your own words, explain why it is crucial to write test cases from the beginning of a project. You can take an example project to explain your answer.**  
     
   Writing test cases from the beginning of a project is a proactive approach that pays off in terms of code quality, early issue detection, collaboration, and long-term project success. It reduces risks, enhances maintainability, and ultimately leads to a more reliable and valuable software product.

### 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 handle the logic of processing user requests and returning responses.

Example:  


When a user visits a URL mapped to my view, Django calls this function, processes any logic, and returns the specified response ("Hello, world!").

1. **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?**  
   When reusing lots of code i would choose a Class-Based Views (CBVs).

Reason: CBVs provide a more modular and reusable way to structure views. They allow you to use inheritance and mixing to reuse common patterns and behaviors across different views, reducing redundancy and making the codebase more maintainable.

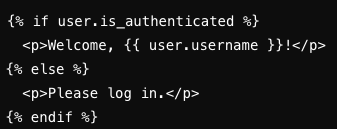
1. **Read Django’s documentation on the** [**Django template language**](https://docs.djangoproject.com/en/3.2/ref/templates/language/#templates) **and make some notes on its basics.**Variables: Use {{ variable\_name }} to output variables.

****

Filters: Modify variable using filters with {{ variable|filter }}.



Tags: Implement logic with tags {% tag %}.



Inheritance: Reuse templates using {% extends %} and {% block %}.



### Exercise 2.5: Django MVT Revisited

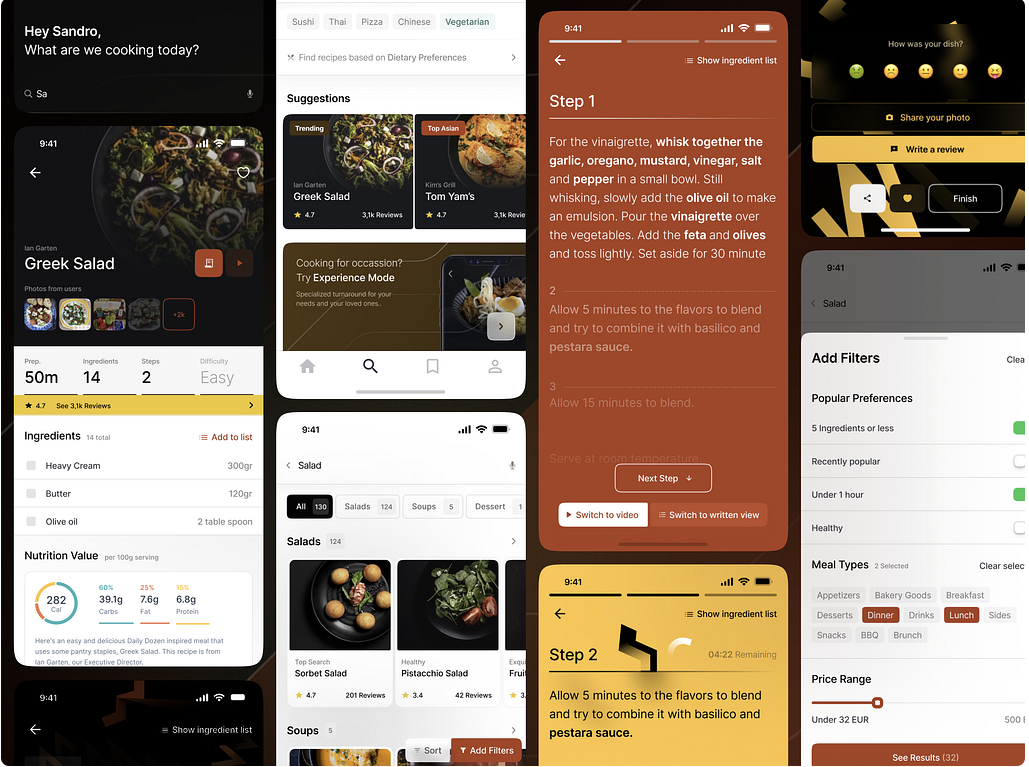
#### Learning Goals

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

#### Reflection Questions

**frontend inspirations**

[**https://dribbble.com/shots/19656468-Foodnetwork-App**](https://dribbble.com/shots/19656468-Foodnetwork-App)

****

Easy to navigate and a lot to choose from. The developer even added a nutrition value, which is useful to those who want to follow calories intake.

[**https://www.motocms.com/website-templates/motocms-3-template/66522.html**](https://www.motocms.com/website-templates/motocms-3-template/66522.html)

Slick design as I like it, no information is too much on the page here.

1. **In your own words, explain Django static files and how Django handles them.**  
     
   Django static files are files like CSS, JavaScript, images, and other assets that are served directly to web clients (browsers) without any processing. Django provides a built-in mechanism for managing and serving these static files, using template tags to include them in HTML templates.
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 class-based view used for displaying a list or collection of objects from a database and rendering them in a template. It’s commonly used for displaying a list of products, blog posts, articles, etc. |
| DetailView | A class-based view used for displaying detailed information about a single object retrieved from a database. It takes a primary key or a unique identifier as a parameter and fetches the corresponding object. It’s commonly used for displaying the details of an object, such as a product description. |

1. **You’re now more than halfway through Achievement 2! Take a moment to reflect on your learning in the course so far. How is it going? What’s something you’re proud of so far? Is there something you’re struggling with? What do you need more practice with? You can use these notes to guide your next mentor call.**  
     
   I feel that I am doing pretty well overall. I am thoroughly enjoying the process of creating the front end of my application and exploring creative ways to display all the elements.

One of the major challenges I faced was designing the models to work together seamlessly. Initially, I over-complicated the design by creating three models instead of two. However, this approach eventually helped me avoid errors and achieve a working solution. Additionally, I struggled a bit with managing my time effectively. However, after taking a break, I feel more focused and ready to tackle the remainder of the course. The upcoming tasks, which involve finalizing the project and making it user-ready, are particularly exciting for me.

I am proud of how the application is turning out visually. The effort put into the front end is paying off, and I am pleased with the data display. This assignment has taken me the longest to complete compared to previous tasks, but it has been enjoyable and rewarding to create the app according to my vision. I am looking forward to the final stages of the project, where I will focus on polishing the application and ensuring it is user-ready. This part of the course is something I am particularly excited about.

### 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.**  
     
   The importance of incorporating authentication into an application is to ensure that only authorized users can access sensitive or restricted parts of the application. Authentication enhances security and privacy, protecting user data and preventing unauthorized access.

For example, in a banking application, authentication ensures that only account holders with valid credentials can view their account balance, transfer funds, or perform financial transactions. Without authentication, anyone could access sensitive financial information, leading to potential security breaches and privacy violations.

1. **In your own words, explain the steps you should take to create a login for your Django web application.**  
     
   - Design a Login Template: Create an HTML template that displays the login form. This template should include input fields for the username and password, as well as a submit button.

- Register the Login View and Define URL: In the project, register the login view and map it to a URL route. This URL should be accessible from the homepage.

- Provide a Login Link or Button: On the homepage, add a button that routes users to the authentication form (the URL you defined in the previous step).

- Secure Protected Pages: Identify which pages or views in the application require authentication. For the class views of this application, use int “loginRequiredMixin import from Django.contrib.auth.mixins. When an unauthenticated user tries to access a protected page, they will be redirected to the login page.

- Redirect After Successful Login: Configure the login view to redirect users to the recipes list view.

1. Look up the following three Django functions on Django’s official documentation and/or other trusted sources and write a brief description of each.

|  |  |
| --- | --- |
| **Function** | **Description** |
| authenticate() | This function is used for user authentication in Django. It takes a username and password as arguments and returns a User object if the credentials are valid, or None if not. You can also pass optional keyword arguments to specify the authentication backend. |
| redirect() | The redirect function is used to perform HTTP redirects in Django. It takes a URL (the destination) as its first argument and returns an HttpResponseRedirect to that URL. You can also use additional arguments and keyword arguments to customize the redirect response. |
| include() | include is used to include URLs from other URL configuration modules. It allows you to modularize your URL routing by including URL patterns defined in other Python modules. |

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

#### Learning Goals

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

#### Reflection Questions

1. **Consider your favorite website/application (you can also take CareerFoundry). Think about the various data that your favorite website/application collects. Write down how analyzing the collected data could help the website/application.**Analyzing the collected data from other websites/applications, such as CareerFoundry, can help in several ways. It allows the platform to understand user behavior and preferences, enabling personalized content recommendations and targeted advertising. It can also identify areas for improvement in user experience, content, and course offerings. Moreover, data analysis aids in tracking the effectiveness of marketing campaigns and overall platform performance, helping the website/application make data-driven decisions for growth and enhancement.
2. **Read the** [**Django official documentation on QuerySet API**](https://docs.djangoproject.com/en/3.2/ref/models/querysets/)**. Note down the different ways in which you can evaluate a QuerySet.**The Django QuerySet API provides various methods to evaluate a QuerySet, including:

● Iteration: QuerySets can be iterated like lists, allowing you to loop through the results.

● Slicing: You can use Python slicing (e.g., queryset[:5]) to limit the number of results.

● Filtering: Methods like filter(), exclude(), and get() are used to filter and narrow down the results.

● Chaining: QuerySets can be chained together, allowing you to apply multiple filters and conditions.

● Indexing: You can access a specific element of a QuerySet by its index, e.g., queryset[0].

● Counting: The count() method returns the number of results without fetching all of them.

● Existence: Methods like exists() check whether any results exist for a QuerySet.

● Aggregation: You can use methods like aggregate() to perform aggregate functions (e.g., Sum, Avg) on the data.

● Conversion to Lists or Dictionaries: QuerySets can be converted to lists or dictionaries using list(queryset) or queryset.values().

1. **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.**  
   QuerySets and DataFrames serve different purposes in different contexts. QuerySets in Django are specifically designed for querying and interacting with a database, making them more suitable for database operations and Django-related tasks. In contrast, DataFrames, often associated with libraries like Pandas, excel at data manipulation, analysis, and transformation. DataFrames offer advantages in data processing because they provide a more extensive set of tools for filtering, grouping, aggregating, and performing calculations on data in-memory, which is beneficial for tasks like data analysis, data science, and statistics. However, they may not be as efficient for direct database interactions as QuerySets.

### Exercise 2.8: Deploying a Django Project

#### Learning Goals

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

#### Reflection Questions

1. Explain how you can use CSS and JavaScript in your Django web application.
2. In your own words, explain the steps you’d need to take to deploy your Django web application.
3. (Optional) Connect with a few Django web developers through LinkedIn or any other network. Ask them for their tips on creating a portfolio to showcase Python programming and Django skills. Think about which tips could help you improve your portfolio.
4. You’ve now finished Achievement 2 and, with it, the whole course! Take a moment to reflect on your learning:
   1. What went well during this Achievement?
   2. What’s something you’re proud of?
   3. What was the most challenging aspect of this Achievement?
   4. Did this Achievement meet your expectations? Did it give you the confidence to start working with your new Django skills?

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