## Exercise 1.1

**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 had no knowledge of coding before registering for the Full-stack developer course with careerFoundry. I worked as an Account Executive for FMCG companies which is not related to programming.

**What do you know about Python already? What do you want to know?**

I have heard that Python is a popular programming language that is beginner-friendly for non-programmers to learn.
I would like to know how to use Python to build a game because I would like to create one.

**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 the week that might be favorable to your facing challenges and growing. Plan for how to solve challenges that arise.**

The biggest challenge for me is that I have only 3 weeks to complete this course. (If not then I need to pay extra money and void the job guarantee.)
Moreover, I am not so good at learning anything new with reading only texts and this course is full of heavy texts. My plan is to submit at least 5 tasks per week. There is no space for failing.

**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 creating the user interface and experience of a website or application, involving technologies like HTML, CSS, and JavaScript to design and implement the visual elements and interactivity that users interact with directly.
***Backend*** web development, on the other hand, deals with server-side logic and database management, using languages like Python, Ruby, or Node.js to handle data, user authentication, and the overall functionality that supports the front end.

**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?**
Python and JavaScript are both high-level programming languages with versatile applications, yet they serve distinct purposes. They share similarities in being interpreted, dynamically typed, and having active communities.

Python stands out as the superior choice due to its unparalleled readability, versatility, and robust ecosystem. Its clean and intuitive syntax fosters efficient collaboration and reduces development time, making it an ideal language for teams.

**Write down 3 goals you have for yourself and your learning during this Achievement.**

1. Be able to code with Python effectively
2. Have Python projects on my portfolio
3. Convince the HR and IT team lead that I am well-versed in Python

## Exercise 1.2

**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 offers numerous benefits over the default Python shell due to its enhanced features and capabilities. It provides an interactive and user-friendly environment with features like tab completion, syntax highlighting, and history navigation, making code exploration and debugging more efficient.

**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? |
|---|---|---|
| Integer (int) | whole numbers, both positive and negative, without any decimal points | scalar |
| Float (float) | numbers with decimal points, allowing for representation of real numbers | scalar |
| String (str) | sequences of characters, such as text | non-scalar |
| List (list) | ordered collections of items that can be of any data type, including other lists | non-scalar |

**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 key difference between lists and tuples in Python lies in their mutability. Lists are mutable, meaning their elements can be modified, added, or removed after creation, making them suitable for situations where data needs to change. Tuples, on the other hand, are immutable, meaning their elements cannot be modified after creation, providing stability to the data. This makes tuples suitable for cases where data should remain constant throughout the program. Lists are defined using square brackets [], while tuples are defined using parentheses ().

**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.**

The necessary data types would be String('str') and Dictionary('dict') to represent the information
***String:*** To store vocabulary words, definitions, and categories. These textual data can be stored using the string data type.
***Dictionary:*** Dictionaries would be a suitable data structure to store flashcards. Each flashcard can be represented as a dictionary with keys like "word," "definition," and "category," where the corresponding values hold the actual word, definition, and category information. Dictionaries allow you to organize and access flashcard data efficiently using meaningful keys.

## Exercise 1.3

**In this exercise, you learned how to use if-elif-else statements to run different tasks based on the 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."**

```
available_destinations = ["Bangkok", "Tokyo", "London"]

user_input = input("Where do you want to travel? ")

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

**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 combine and manipulate boolean values (True or False). There are three main logical operators: "and", "or", and "not".

The "and" operator returns True only if both operands are True.

The "or" operator returns True if at least one of the operands is True.

The "not" operator is a operator that negates the boolean value, turning True into False and False into True.

**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.**

Today I gained a deeper understanding of conditional statements in Python, mastering the art of making decisions within my code. Explored the realm of loops in Python programming. I feel more confident in my grasp of these fundamental concepts, ready to apply them creatively to a variety of coding challenges.

## Exercise 1.4

**Why is file storage important when you're using Python? What would happen if you didn't store local files?**

File storage is really important when I use Python because it lets me keep information even after my program finishes running. If I don't store files locally, any data I use or create during our program's time won't stay once the program stops. This means I won't be able to save data for later use, share it with others, or use it in different sessions. Storing files helps Python programs to work with information over time and do more useful things.

**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 type of file format in Python that's used for storing data in a way that preserves its structure and type. It's like putting data in a container I can open later and get the same data back. I would choose to use pickles when I want to save complex data, like lists or dictionaries, to a file so that I can use it again later, even in a different session.

**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?**

I can use the function os.getcwd() to discover the current. If I wanted to change my current working directory I would use os.chdir(new_directory_path).

**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?**

I would enclose the potentially problematic code with a "try" section, and if an error occurs, the script won´t terminate but will instead move to the corresponding "except" section where I can handle the error.

**You're now more than halfway through Achievement 1! Take a moment to reflect on your learning in the course so far. How is it going? What's something you're proud of so far? Is there something you're struggling with? What do you need more practice with? Feel free to use these notes to guide your next mentor call**.

I find Python readable and enjoy learning. I am proud of my cocktail recipe and how I could fix the bugs on my own so far. To use Python professionally, I would need to practice doing more projects.

## Exercise 1.5

**In your own words, what is object-oriented programming? What are the benefits of OOP?**
Object-oriented programming (OOP) is a way of organizing and designing computer programs by using objects. In OOP, these objects can interact with each other to perform tasks, making it easier to manage complex systems and promote code reusability.

**What are objects and classes in Python? Come up with a real-world example to illustrate how objects and classes work.**
In Python, objects are individual instances of a class, which is a blueprint or template defining the structure and behavior of objects. For instance, imagine a class named "Car" that has attributes like "color" and "brand." An object created from this class could represent a specific car, such as a red Toyota, which embodies the characteristics and behaviors defined by the class.

**In your own words, write brief explanations of the following OOP concepts.**

| Method | Description |
|---|---|
| Inheritance | A new class can inherit the properties and methods of an existing class, allowing for the reuse of code. |
| Polymorphism | The ability of different objects or classes to be used interchangeably through a common interface. |
| Operator Overloading | Giving special meanings to operators when used with custom objects, allowing them to perform specific actions beyond their usual functionality. |

## Exercise 1.6

**What are databases and what are the advantages of using them?**
A database is a structured collection of organized information that can be stored, managed, and retrieved electronically. In Python, databases offer benefits like efficiently storing large amounts of data in a structured manner and providing quick data access and manipulation through queries.

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

| Data type | Definition |
|---|---|
| INT (Integer) | Used for holding whole numbers. |
| VARCHAR(n) | String of variable length, with n representing the maximum number of characters. |
| DATE | Stores calendar dates like 'YYYY-MM-DD' for recording specific days. |

**In what situations would SQLite be a better choice than MySQL?**
SQLite would be a better choice than MySQL in simpler projects or when simplicity is important. For instance, if you need a database for a small mobile app or a personal project, SQLite is handy as it's embedded directly into the application, requiring no separate server setup. It's also easier to manage because it's a single file, while MySQL needs a separate server to be set up. However, for larger projects with many users and complex needs, MySQL is more suitable due to its client-server architecture and advanced features.

**Think back to what you learned in the Immersion course. What do you think about the differences between JavaScript and Python as programming languages?**
I find Python often chosen for its simplicity and readability, making it suitable for beginners. On the other hand, JavaScript is mainly used for web development and focused on web-related tasks and can be more complex due to its asynchronous nature.

**Now that you're nearly at the end of Achievement 1, consider what you know about Python so far. What would you say are the limitations of Python as a programming language?**
I've learned the fundamentals of Python to construct a database. I used my knowledge of Python to create a structured storage system for organizing data. This involved understanding how to define, add, retrieve, and modify information within the database. By applying my Python skills, I gained the ability to design a simple but functional database that can store and manage data effectively.

My point of view is that the limitation of Python is the execution environment. JavaScript runs directly in browsers, making it easily accessible for web development, while Python often requires additional tools and frameworks to run in a browser context.

## Exercise 1.7

**What is an Object Relational Mapper and what are the advantages of using one?**
An Object Relational Mapper (ORM) in Python is a helpful tool that links software code with databases. It makes it simpler for developers to manage and retrieve data from databases (like MySQL or PostgreSQL) using familiar programming objects instead of writing complex database queries. This makes coding faster and less error-prone. In Python, using an ORM like SQLAlchemy or Django's ORM provides advantages such as easier code maintenance, increased productivity, and improved data security through built-in features for data validation and protection.

**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?**
At this stage, I have successfully completed building the Recipe app using Python. I am pleased with how smoothly I managed to store and retrieve recipe information in the app's database. However, in hindsight, I would prioritize incorporating more comprehensive unit and integration testing during development to ensure the app's reliability and minimize potential bugs.

**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.**
During my previous projects, I gained valuable experience in creating a database for a recipe app using Python. I successfully designed and implemented a database schema to efficiently store various aspects of recipes, such as ingredients, difficulties, and user preferences.

I utilized Python's database libraries, like SQLAlchemy, to establish connections between the app and the database, allowing seamless data retrieval and manipulation. I ensured data integrity and security by implementing proper validation and encryption techniques. Through this project, I honed my skills in database design, Python programming, and data management within the context of recipe applications.

**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?**
   It went well in understanding the concept of data type in Python and applied it to the project including methods because of how readable Python language is.

b. **What's something you're proud of?**
   I often struggle to learn new programming languages but I am proud every time when I find the solution even if I know it from StackOverflow or Google.

**c. What was the most challenging aspect of this Achievement?**

To complete this achievement within 10 days and I did it.

**d. Did this Achievement meet your expectations? Did it give you the confidence to start working with your new Python skills?**

I would be more confident starting from the front end because it is also an essential part of knowing Python.

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

The deadline will keep me running. I must finish the course before next week.

## Exercise 2.1

**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 (plain) Python for a project offers the advantage of complete flexibility and control, allowing for a highly customized solution tailored to specific project requirements. However, this approach often involves writing more code from scratch, which can be time-consuming and prone to errors.

On the other hand, utilizing a framework like Django provides a structured and efficient development process, with built-in features like authentication, database handling, and routing. This speeds up development, ensures best practices, and benefits from a supportive community. Nonetheless, it may limit some flexibility and might be perceived as overbearing for smaller or simpler projects. Ultimately, the choice between vanilla Python and Django depends on project complexity, development speed, and the need for standardized features.

**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 the Model View Controller (MVC) architecture is its simplified and clearer separation of concerns. In MVT, the Template layer handles the presentation logic, the View layer deals with user interactions and acts as a bridge between the Template and the Model (data), and the Model layer handles data manipulation and storage. This clear division makes it easier for developers to understand and work on different parts of the application independently, leading to improved code organization and maintainability. In contrast, MVC, while similar in concept, can sometimes lead to confusion regarding responsibilities and interconnections between the components, making MVT a more straightforward and coherent approach for web application development.

**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**
I want to learn the fundamentals of Django to build web applications effectively. I aim to gain a solid understanding of Django's models, views, templates, and overall framework structure. My primary goal for this achievement is to create functional web applications and gain confidence in handling real-world projects. After completing this achievement, I envision myself working on more complex web development projects, either as a freelance developer or within a web development team, and continuing to expand my skills and knowledge in Django and web application development in general.

## Exercise 2.2

**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.**

I would start by breaking down the YouTube website into Django terms:

***Frontend Views:*** The various pages on YouTube, such as the homepage, video watch page, user profiles, and search results, would correspond to Django views. Each view would have its URL route defined in Django's URL patterns.

***Templates:*** The HTML templates used to render the pages would be Django templates. These templates would include placeholders for dynamic content, which would be filled using Django's template language.

***Models:*** YouTube's database would translate into Django models. For example, there would be a User model to represent user accounts, a Video model for videos, and other models for comments, likes, and subscriptions. These models would define the data structure and relationships.

***URL Routing:*** Django's URL patterns would map the different URLs of the YouTube website to specific views. For instance, '/watch/video_id' would map to a view for watching videos.

***Authentication:*** YouTube's user accounts and authentication would be handled using Django's built-in authentication system. Users would be able to register, log in, and manage their profiles.

***Database:*** YouTube's vast data storage, including videos, user information, and comments, would be managed using Django's database features, typically with PostgreSQL, MySQL, or SQLite.

***Static Files:*** Static assets like CSS, JavaScript, and images would be served using Django's static file-handling capabilities.

***Forms:*** Any forms on YouTube, such as the comment form or the search bar, would correspond to Django forms for user input handling and validation.

***Middleware:*** Various middleware, such as security and authentication middleware, would be used to enhance the security and functionality of the Django application, similar to how YouTube employs security measures.

***Admin Panel:*** Django's admin panel would be used to manage the site's content and users, just as YouTube's administrators manage the platform.

**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 on my system, I would first ensure that Python and Django are installed. Then, I'd create a virtual environment for my project to keep dependencies separate. After activating the virtual environment, I'd use Django's command-line tool to start a new project. Next, I'd configure the database settings in the project's settings.py file and run migrations to create the database tables. Then, I'd create a superuser for admin access and run the development server. Finally, I'd open a web browser, access the local server, and verify that my basic Django application is up and running.

**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 powerful tool that I'd use during my web application development to manage the application's data and user accounts. It allows me to easily add, edit, or delete records in the database using a user-friendly interface, making it efficient for testing and populating initial data.

## Exercise 2.3

**Do some research on Django models. In your own words, write down how Django models work and what their benefits are.**

Django models are the basis of Django apps. They store data in a database. It is easy to use and has many benefits, such as automatic database schema creation and validation. Django models can be used to implement security features as well.

**In your own words, explain why it is crucial to write test cases from the beginning of a project. You can take an example project to explain your answer.**

It helps to identify and fix defects early in the development process. Besides, ensuring that the website meets its requirements improves the quality of user's needs, and reduces the risk of defects

## Exercise 2.4

**Do some research on Django views. In your own words, use an example to explain how Django views work.**

Django views are like the boss of a website. They receive requests from users and decide what to show them. For example, imagine a restaurant website. When you want to see the menu, you click on a link. This sends a request to the Django view, which then fetches the menu and decides how to display it. If you want to see the specials, you click another link, and a different Django view handles that request, showing you the specials instead. So, views in Django are like waiters in a restaurant who take your order (request) and serve you the right dish (response) based on what you asked for.

**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 this situation, I would choose to use Django class-based views because they offer a more organized and reusable way to structure code. With class-based views, I can create a base view class containing common functionality, and then extend it in different parts of the project as needed. This makes it easier to maintain and update code, reducing duplication. For example, if I have similar features like user authentication or data filtering across multiple pages in my project, class-based views allow me to create a consistent and modular approach, resulting in cleaner and more efficient code.

## Exercise 2.5

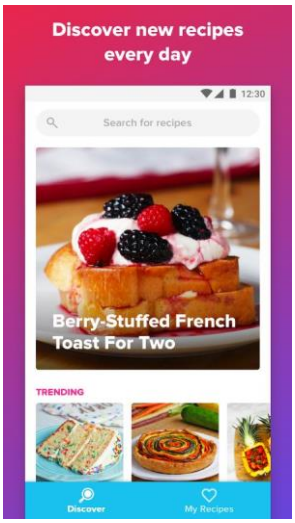
**In your own words, explain Django static files and how Django handles them.**
Django handles these static files by creating a designated folder, usually named "static," where developers can organize and store these assets. When a user accesses a web page, Django's template system allows you to include links to these static files within HTML templates. To ensure proper delivery, Django provides a {% load static %} tag for loading static files and a {% static 'file_path' %} template tag to generate the correct URL for the static file.

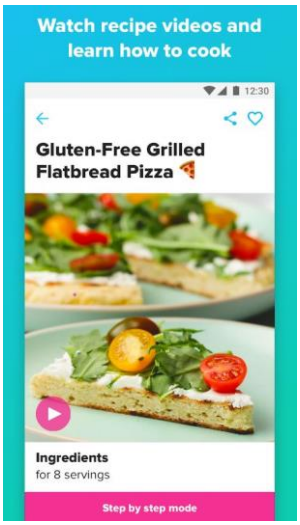
**Look up the following two Django packages on Django's official documentation and/or other trusted sources. Write a brief description of each.**

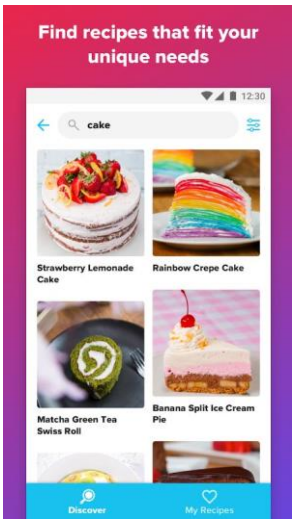
| Package | Description |
|---|---|
| ListView | a class-based view that helps display a list of objects from a database typically used for generating paginated lists of items. |
| DetailView | a class-based view used to display detailed information about a single object retrieved from a database often used for viewing a specific item's details. |

# frontend inspirations

**Tasty:** The front end of Tasty is very user-friendly and easy to navigate. The recipes are presented in a visually appealing way, with clear instructions and step-by-step photos. The app also has a built-in timer, which is helpful for keeping track of cooking times.

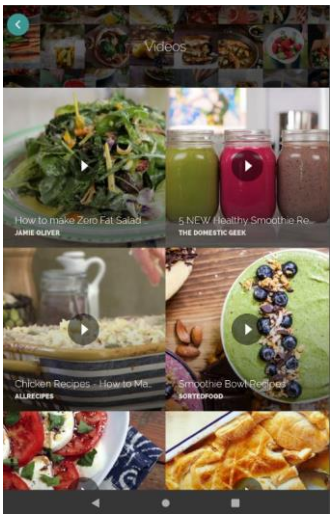


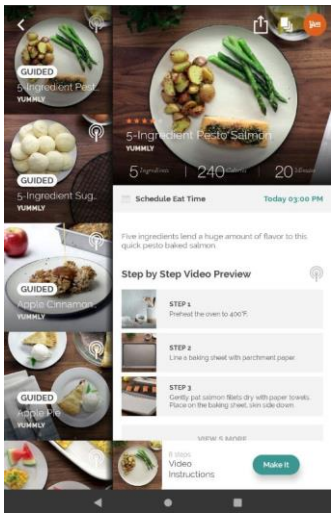



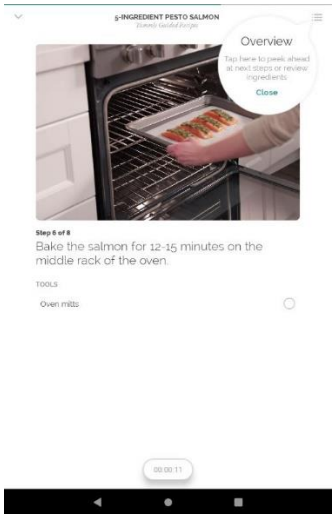




https://play.google.com/store/apps/details?id=com.buzzfeed.tasty&hl=en_US&pli=1

**Yummly:** The front end of Yummly is also very user-friendly. The app allows you to search for recipes by ingredient, dietary restriction, or cooking time. You can also save recipes to your favorites list or create meal plans. The app also has a voice search feature, which is helpful if you're cooking with your hands dirty.

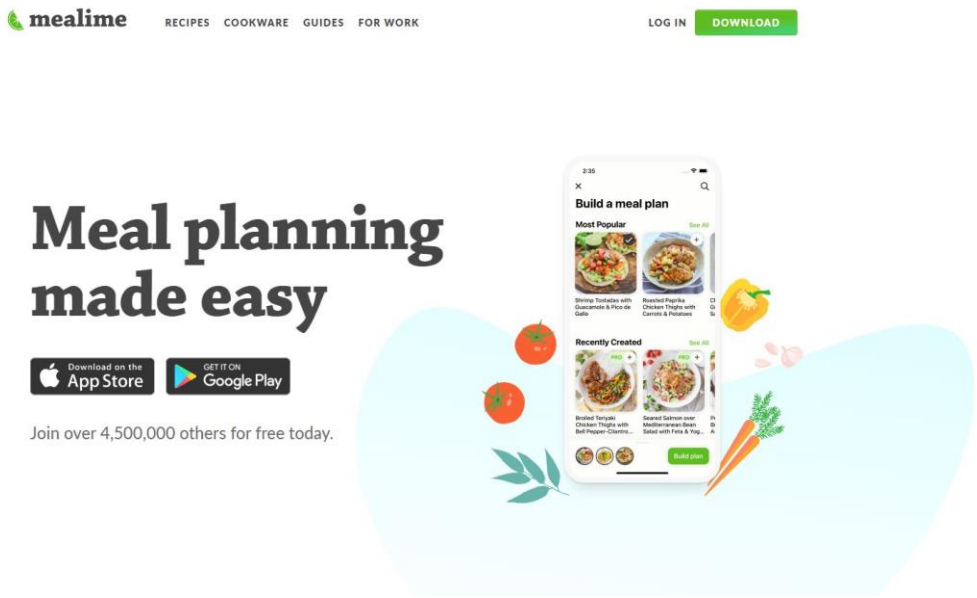



https://play.google.com/store/apps/details?id=com.yummly.android&hl=en_US

**Mealime:** The front end of Mealime is designed to be as simple and straightforward as possible. The app allows you to create meal plans based on your dietary preferences and budget. You can also add recipes to your meal plan from the Yummly database. Mealime also has a grocery list feature that can help you make sure you have all the ingredients you need.




https://www.mealime.com/

## Exercise 2.6

**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 crucial in a recipe application because it helps ensure the security and privacy of user data. When users create accounts and log in, authentication verifies their identity, preventing unauthorized access and protecting personal information.

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

1. **Create Login and Registration Forms:** Define Django forms to handle user input for login and registration.
2. **Add Authentication URLs:** Include authentication URLs in your project's urls.py file.
3. **Create Authentication Views:** Implement login and registration views in your app, handling form submissions and user authentication.
4. **Templates and Styling:** Link your HTML templates to the views, and add CSS styling for a user-friendly interface.
5. **Test:** Test the login and registration functionality thoroughly to ensure it works as expected.

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

| Function | Description |
|---|---|
| authenticate() | a function to verify the identity of a user. It takes user credentials, and checks them against stored user data |
| redirect() | a function used to navigate users from one web page to another |
| include() | to include reusable pieces of code or templates within a larger web page |