Python
Learning Journal

Pre-Work: Before Starting

Reflection questions

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?

   Answer: Prior to this specialization course on Python, I've completed a frontend and a full-stack development course at CF. Throughout these courses, I built a website, several web apps and a mobile app. In order to do so, I've used different technologies / tools such as HTML, CSS, JavaScript, Bootstrap, Node.js, MongoDB, PostgreSQL, React, React Native, Angular, and more. I also learned how to develop different coding technics such as the test-driven development (TDD) or the behavior-driven development (BDD) methods.

   Before enrolling into CF, I also had different working experiences that were not related to coding, but related to data and technologies, which could help me in this course and my coding learning in general.

   For example, I worked as a Senior analyst in data and technology policies, where my main focus was oriented towards data governance, data lifecycle and R&D on sensitive data sharing across organizations. All the work carried on while I was occupying this position was done in an Agile environment. Before that, I also worked as a Professional researcher on a project related to supporting decision-making in a municipal infrastructure development context, using GIS and non-GIS data. During this project, I've been exposed to some Python codes, but only as an observer, not a developer.

2. What do you know about Python already? What do you want to know?

   Answer: I don't know much about Python so far, except that it's a widely used language across different domains, and that it's one of the most popular coding technology among developers.

   I am very enthusiastic about learning Python. I would like to learn everything useful about this language, from how to set a Python coding environment, to how Python codes are generally structured, what are Python specificities and what makes it a better tool than another language on different types of projects. In general, I would like to acquire a good Python knowledge, both on theory and practice, allowing me to use Python whenever it would be the best tool for X project.

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.

   Answer: As for the challenges that may arise, the main one I am thinking of is the ability to distance myself from the coding concepts, patterns, and structures of other coding languages

I've learned so far (mainly JavaScript), if necessary. I am wondering if, in order to learn Python, it is necessary to distance myself from JavaScript concepts and principles, or if, in the opposite, both languages are close to each other and so knowing one makes it easier to learn the other.

Learning a completely new language also represents a good challenge, since it requires more time and effort in comparison to developing additional knowledge in a language for which the basis is already well-know. This learning from the scratch on Python is however extremely important in order to have a solid foundation on which to build further knowledge.

In order to face these challenges and allow me to successfully acquire a good Python knowledge, I will apply several methods I've developed over my coding learning journey so far. They could be breakdown as follows:

- Take the time to read each text content on Python correctly, and read it back when something isn't clear.
- If something in the course material is still not clear (even after a second read), go online to search for that same topic and see if complementary information / examples are available to help my understanding.
- Always do each task at the best of my capacities – the goal not being to simply get the task accepted, but to understand everything I've done, and why.
- In each task, when something isn't working as expected, always try to troubleshoot the problem by myself first via all the debugging technics I could think of.
    - This not only allow me to develop my debugging skills, it also helps me to remember how to fix that specific problem if I ever encounter it again in the future.
- Compare my Python codes to some other codes I've written in the past (e.g.: in JavaScript) if I think it could be useful to better understand the differences and the functioning of each language respectively.
- Ask my mentor for any questions I have not been able to find the answer by myself.
- Continue learning about Python even after the course end, by reading theory books/articles and by creating other apps by myself.

1.1 - Getting Started 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?

   Answer:

- **Backend**: Backend is related to all the codes and structures that are not seen by the users on their UI, but that are still essential to ensure that all the elements the users can interact with / see on their screen actually work as expected.

  For example, a backend developer could work on setting up the different API endpoints that allow users to perform different operations in the web app (or other product) database. If a user creates an account, he'll be using a form displayed on the frontend. However, as soon as he submits the form, the API endpoint created to handle user creation account in the backend (POST request) will be used, and the database (which is also part of the backend) will be updated accordingly, by registering the new user info.

  On top of endpoints and database, other elements could also be implemented by a backend developer, such as codes to hash users' passwords, a CORS policy to protect the web app, or codes to define database schemas / models.

  In the MERN and MEAN tech stacks, the backend is represented by Node.js, Express and MongoDB for example.

- **Frontend**: Frontend is related to all the codes and structures that are seen by the users on their UI. The frontend is closely related to the backend, as those two work together to ensure the web app (or other product) is fully functional.

  For example, if the backend of a web app has been developed to allow users to create an account and delete it, the frontend will display a user registration form on the users UI, as well as a delete account button. When these elements are used by users, the backend logic is called and perform the required action.

  Important considerations for frontend development usually include easily navigation through the web app, accessibility, great visual, responsiveness, good communication with the backend, and so on.

  In the MERN and MEAN tech stacks, the frontend is represented by React (MERN) and Angular (MEAN) for example.

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?

   Answer: Both languages could be interesting, as both of them share some similarities. For example, both are a high-level scripting language and uses easily understandable keywords to make commands and perform tasks, which is a great plus. Python, as well as JS, uses dynamic typing, an approach that allows variables to assume any kind of value without producing errors, which is another interesting element.

   However, I would suggest to use Python for the following reasons:

   - Python frameworks come with pre-installed common web operations such as URL routing, form handling and validation, template engines, etc. This helps developers to move quickly with application development.
   - Python code is easily readable, which help to understand documentation and performing debugging more easily. Python also comes with its own shell called REPL, which is fairly easy and simple to use, and can be of a great help to test python code syntax and perform debugging quickly.
   - Finally, Python is widely used, so there's a big community of developers around it, a lot of documentation, discussions, GitHub topics and so on. This is very helpful to find different important information over the development process.

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?

   Answer:

   1- By the end of this achievement, I want to have a very good base about core Python principles, in order to be able to discuss on Python with very good level of confidence if someone was to ask me questions such as:

   - What is Python?
   - What are the main characteristics of Python?
   - In which scenarios / projects Python would be the best tool, and why?
   - What are the main differences between Python and JavaScript?

   2- By the end of this achievement, I want to know enough about Python to at least be able to identify what personal projects I could build using Python, and have a good idea of how the different features could be implemented.

3- By the end of this Achievement, I would like to improve the way I developed my documentation. More precisely, I would like to have a professional level README related to my project, to show my Python technical communication skills in the best way possible.

1.2: Data Types in Python

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?

   Answer: In essence, Python's default shell is not that user-friendly, especially when it comes to reading code. It's a good tool for executing and debugging Python commands, but the text in the Python default shell is the same color, making it difficult to distinguish writing, keywords and individual lines of code. Code also needs to be indented manually each time if we are writing out functions or other nested statements, which is not that practical.

   On the other hand, the iPython Shell provide a clearer code visibility, because it displays different features of the code in contrasting fonts and colors. Indenting text for nested statements is also done by iPython shell automatically. Those details, among others, makes it easier to work with iPython than Python default shell.

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?
Integer	Data type that represents both negative and non-negative numbers (from zero to infinity). Does not include numbers with decimal, since these are **float.**	Scalar
Bool	Data type that represents either of two values : **True** or **False**.	Scalar
List	Data type that represents linear arrays that can store multiple values of different types, such as numbers or string. Lists differ from tuples (another form of similar data type) in that they are mutable - any of the internal elements of a list can be modified or deleted.	Non-scalar
Dictionnary	Data type that represents an unordered set of items,	Non-scalar

	each of them being a key-value pair, where each key is unique. Each item is individually labeled, and does not sequentially related to the other items. A dictionary allows different data types in the key-value pairs such as string, integers and lists.	

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.

   Answer: Tuples are linear arrays that can store multiple values of any type. They're useful for situations where we need to save several values, but naming a new variable for every one of them would be too much hassle. Tuples can't be altered once they're defined, since they're ***immutable***. The only way to change elements in a tuple is to write a new version of the tuple and assign it the original variable name.

   *One small exception to this is when we want to **add** an element in a tuple - it's possible to do so on the original tuple by placing the new value we want to add in a single-element tuple, and then concatenate (merge) the two tuples together.

   List are another type of ordered sequence, similar to tuples. However, lists differ from tuples in that they are ***mutable***, meaning that any of the internal elements of a list can be modified or deleted, which is not the case in a tuple. It's also possible to rearrange or insert new elements in lists.

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.

   Answer: For the Flashcards' creation by users, I would use dictionaries regrouping key-value pairs of strings. The strings would let users write their inputs regarding the vocabulary words, their definitions, and their word category for each flashcard. This structure would allow users to create as many cards as wanted easily. The dictionaries are also flexible, so it would be easy to add potential new key-values pairs / features in the flashcards in the future (with other data types than strings if necessary).

Example:

```
flashcard_1 = {
            'word_name' = 'word',
            'word_definition': ‘definition’,
            'word_category": ‘category’
           }
```

To allow users to quiz themselves, I would create an outer list, which would contain all flashcards created previously by the user. I would then set up the UI so that this list would be used to show each card in a way that allows users to quiz themselves / do some revision.

The list would be more useful than a tuple, since lists are mutable, meaning that any of the internal elements of in it can be modified or deleted. They can also be sorted, which could be great if users want to organize their cards in certain ways. All of these characteristics would make the list a great choice.

Example:

```
flashcard_list = [‘flashcard_1', ‘flashcard_2’, ‘flashcard_3']
```

Exercise 1.3: Functions and Other Operations in Python

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.

```
travel_destination = str(input("Where would you like to travel? "))
travel_destination = travel_destination.capitalize()
```

```
if travel_destination == "Canada":
    print("Enjoy your stay in " + travel_destination + ' !')

elif travel_destination == "Mexico":
    print("Enjoy your stay in " + travel_destination + ' !')

elif travel_destination == "Bolivia":
    print("Enjoy your stay in " + travel_destination + ' !')

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.

   Answer: In Python, logical operators are OR, AND and NOT. The OR and AND are used to check more than one condition at the time and return a boolean value accordingly (**true** or **false**).

   The OR operator checks if at least one condition is **true** in a statement. If it's the case, the code using the OR operator will return a **true** value. The only way a code using the OR operator will return a **false** value is if all conditions are **false**.

   The AND operator checks if at all conditions are **true** in a statement. If it's the case, the code using the AND operator will return a **true** value. In any other situation (if one of the condition is **false** or all conditions are **false**), the code using the AND operator will return a **false** value.

   The NOT operator is used to reverse the result of a logical expression that comes after it. For example, if a condition statement would normally return a **true** value, but the NOT operator is placed before the condition statement, the result will be reversed to **false**.

3. What are functions in Python? When and why are they useful?

   Answer: In Python, there are built-in functions (e.g.: print( ) and append( )) and on-purpose created functions (functions created by developers to answer certain needs). In essence, functions are sets of instructions that process the code in order to achieve certain things. They can have argument(s), which ensure that the functions are running with specific value(s) to process.

   Functions are particularly useful, because they can be defined one time, and then called whenever necessary elsewhere in the code. This allows to reduce significantly the amount of code / repetition that is required to run some actions, since a simple call to an already defined function doing these same actions do the thing.

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.

Answer: So far, I think I mostly progressed on my goal #1, which was to have a good base about core Python principles, in order to be able to discuss about Python with very good level of confidence if someone was to ask me questions such as:

- What is Python?
- What are the main characteristics of Python?
- In which scenarios / projects Python would be the best tool, and why?
- What are the main differences between Python and JavaScript?

Throughout the exercises, I acquired a good knowledge of the different data types in Python, as well as the logical operators, the if-elif-else statements, loops, functions and so on. I also have a better idea of what are the benefits of developing with Python, and how to use effectively iPython Shell, which are all pieces of knowledge that could be useful in any discussion regarding Python, whether on a more technical level, or more general level.

As for my two other goals, I feel like I am progressing, but that more knowledge / practice / reflection / revision would still be necessary / useful to reach them more.

Exercise 1.4: File Handling 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?

   Answer: File storage is important because when we use variables in scripts to assign and keep track of values, these data no longer exists when the script stops running. Data can't be retrieved for later use if they're not stored, which is not practical at all. Data therefore need to be stored somewhere in files in order to be read, updated or modified later, even after the script, the program, or 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?

   Answer: Pickles are useful when we need to store more complex data structures than just strings or numbers (dictionaries for example). Simpler data structures can be stored, for example, in txt. files, but with more complex data, pickles are the best option

   Pickles convert complex data into a packaged stream of bytes, known as a 'pickle', then write this into a binary file. A binary file can store complex information in a way that can be read by a machine but not by humans. Then, when necessary, it is possible to retrieve and read these data

stored as binary file using **pickle.load()** (and some logic to indicate how we want to display specifically the data retrieved).

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?

   Answer:

   To know in which directory you are in, we can run the command: **os.getcwd()**
   To change from a working directory to another, we can run the command: **os.chdir(<path to desired folder>)**

   *Important to note that to run these commands, it is necessary to import the **os module**:
   import os

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?

   Answer: I would implement a **try-except** block. More precisely, I would place the code where I expect a potential error to occur in the **try** section. That way, if no errors are found, the rest of my code would execute as normal. However, if there's an error, a logic implemented in the **except** section (following the **try** one) would be put in use to notify me. That way, if any errors are found, Python won't terminate the entire program, but will highlight the existence of an error so I can fix it.

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.

   Answer: So far so good. I am particularly happy and proud that in general, I am capable to fix the problems I encounter when I code in Python by myself, without having to rely on external resources such as discussions on Github or Stack Overflow for example. Maybe this is because the way Python is (generally very readable and logic at first sight), or it's because I already had a good base in coding before jumping into learning Python (with JavaScript).

   I can't say that I am struggling with something in particular. Of course, I've encountered some problems and bugs so far while I worked with Python, but I've always been able to fix the issue after a certain time.

   That being said, I would like to get more familiar with the **for loops**, the **try-expect blocks** and how to write data into **binary files**, as these elements are very useful in many ways. They are not necessarily complicated to work with, but I would like to figure out more rapidly in which scenario they should be used and could be helpful, and know right away how I could code it / implement logics accordingly.

Exercise 1.5: Object-Oriented Programming in Python

Reflection Questions

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

   Answer: OOP is a programming concept that organizes the code around data (also called objects). It's different from functional programming.

   Most of the objects in Python can be broken down into the data they contain and into the methods that are possible to use to interact with them. Each object also has a corresponding type or class (template describing an internal structure). In OOP, objects are instances of classes (and classes define the structure and behavior of objects). The process of abstracting data and methods into classes is part of OOP. The advantage of pre-defining a class is that the script can then generate objects containing empty/partially data attributes that are waiting to be filled later.

   Inheritance, which allow to create new classes based on existing ones (inheriting their attributes and methods from other classes), and **Polymorphism**, which allow data attribute or method that has the same name across different classes or data types to perform different operations depending on where it was defined, are some key elements of OOP.

2. What are objects and classes in Python? Come up with a real-world example to illustrate how objects and classes work.

   Answer:

   Object: Most of the objects in Python can be broken down into the data they contain and into the methods it's possible to use to interact with the data. Some examples of objects:

   - **Numbers**: 123, 123.45
   - **Text**: "Hello"
   - **Lists**: ['This', 'is', 'a', 'list!']
   - **Tuples**: ('And', 'this', 'is', 'a', 'tuple.')
   - **Dictionaries**: { 'A+': 100, 'A': 90, 'A-': 80, 'B+': 70, 'B': 60, 'B-': 50 }
   - **Instances of custom classes**

   Class: Each object also has a corresponding type or class (a template describing an internal structure). One class may contain multiple corresponding objects. It's possible to define specific classes based on different needs (rather than relying on only in pre-built elements), in a similar way to how it's possible to define your own functions.

When creating a class, some specific elements are required:

- The **class** keyword
- The **class name** (to be chosen - normally in CamelCase)
- The **class object** (the class inheriting from - which is **object** by default)
- The class body, which contain data attribute and procedural methods

A real-world example of objects and classes could be a province and the cities within the province. The province would be the class and each city inside of it would be an object that belongs to this class.

Each city (or object) would hold data attribute such as the city name, date of foundation, square kilometers, population… (store information about the object), as well as methods for interacting with them and perform operations. Such methods could be used to calculate the population median age, the number of green areas compared to the total surface of the city etc.

Using the **class province** would make it easier to create any new **object city**, or update/modify an existing object city.

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 describes to possibility to inherit methods from one class to another, when the properties or methods of a particular class could be useful to another class (instead of making copies of the code).
Polymorphism	Polymorphism is when a given data attribute or method has the same name across different classes or data types, but performs different operations depending on where it was defined.
Operator Overloading	When using operators such as + and - on a custom class, a *TypeError* is thrown, since the operators aren't supported. To use such operators in a custom class, it is necessary to define methods for them, and this process is known as operator overloading. Most of Python's built-in classes however support special operators, so operator overloading is not necessary in those cases.

Exercise 1.6: Connecting to Databases in Python

Reflection Questions

1. What are databases and what are the advantages of using them?

   Answer: Databases have various advantages over regular local storage. They keep data in a standardized format so it's possible to store and access it more easily than with some other methods, they can be made secure through password access and, they can be accessed using applications such as Python and or others. Database Management Systems (DBMS's) offer interfaces which let database user read, store, and modify data with ease, which are all great working advantages.

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
Integer	Standard integers (full number without decimal, whether positive or negative)
Float	Floating-point decimal numbers

3. In what situations would SQLite be a better choice than MySQL?

   Answer: In different situations. For example:

- When we need to work with very simple databases, like a web page for a shop that stores customers' email addresses.

- When we just want to test a database without having to set up an entire database engine.

  In such situations, SQLite -requiring no installation or setup- is a good option, since it lets us store data in simple .db files, and access and modify these files directly from applications such as Python.

4. Think back to what you learned in the Immersion course. What do you think about the differences between JavaScript and Python as programming languages?

   Answer: I feel like it's a bit more simple to code different conditional logics in Python than in JavaScript. The readability of the codes in Python feels a bit easier as well, and I really like the development/testing environment provided by iPython.

   However, both languages remain of interest to me, because both are widely used, and both have a strong online community, which can be very useful.

   I feel like I would gain to learn more about Python in order to be really able to make a good comparison, because I've passed much more time on JavaScript so far, so I know more things about on this language (e.g.: elements related to frontend frameworks) that I have yet to acquire for Python.

   I am very enthusiasm to continue my learning on Python because it's a language I start to appreciate a lot and for which I realize all the logics and processes it's possible to do with.

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?

   Answer: I think the main one is that Python is usually not used for frontend development, meaning that being great with python might not always be sufficient to develop complete web applications, for example. Learning and mastering other frontend technologies or complementary tools to Python is very often necessary to work on the frontend side.

   In opposition, JavaScript can be used both in backend and frontend development fairly easily, and can rely on many frameworks / libraries for that, which is great because once mastered, it represents a very versatile tool.

Exercise 1.7: Finalizing Your Python Program

Reflection Questions

1. What is an Object Relational Mapper and what are the advantages of using one?

   Answer: An object-relational mapper (ORM) converts the contents and structure of a database into classes and objects that can be interacted with directly.

   Thanks to that, an ORM saves developers a lot of time and let them concentrate on more important tasks. The simplification that ORMs offer is part of what is known as *Pythonic style* - a style well suited to Python's features.

To implement ORM, it is possible to use an open-source Python SQL toolkit called SQLAlchemy, which comes with many tools for managing the interface between an application and a database.

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?

   Answer: All in all, it went pretty well. I think I handled particularly well all the conditional logics in the app, using a mix of operators (such as OR, AND, NOT), if-elif-else logics, and for loops and while loops. I think I also handled well the different data types I've been working with, from being able to recognized them to convert them and manipulate them whenever necessary.

   Concerning any possible improvements, knowing what I know now, I think I would change the way I started to code the whole app in the last / final step. I think I would have gained from having a logical schema presenting all the possibilities available for users when navigating in the app. I quickly realize that such app require a lot of conditional statements, and a lot of options to think of / implement to ensure a good and sound navigation between the different parts of the app.

   Having a visual / schematic representing all these possible navigating routes would have helped a lot and serve as a good reference to start implementing codes from scratch.

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.

   Answer: I've been using Python to develop the backend of a recipe app. Basically, this app allows users to create new recipes, view existing recipes, search for recipes by ingredients, update existing recipes and delete recipes.

   In order to do so, I've used many logics to perform the desired actions, including the use of OR, AND and NOT operators, as well as if-elif-else statements, for loops and while loops, and classes and functions definition.

   During the development, I've relied a lot on iPython, which I found it to be a great tool during the development and testing phase.

   Regarding the data shown to users and with which they could interact, they were all stored in an mySQL database. To interact with those data, I've implemented an object-relational mapper. I've used the open-source Python SQL toolkit called SQLAlchemy. That way, all operations on the app data were performed using this ORM, rather than using SQL queries.

   Along the way -and even if it wasn't part of the final app-, I also worked on getting to know the different python data types and how to manipulate them, storing/retrieving data in/from binary files (pickles), handling errors and exceptions in my codes, and code in an object-oriented programming logic.

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?

    Answer: Coding the app in general and using different logics to accomplish different conditional desired behaviors.

    b. What's something you're proud of?

    Answer: I'm particularly proud that for most parts of this Achievement, I've been able to find solutions to my problems / bugs by myself, without relying too much on external sources on the web such as Github or other forums. I feel like I got a good grasp of how Python work so far, and I am particularly happy about this.

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

    Answer: Probably working with SQLAlchemy. It was the first time I heard about this, and the first time I heard about the concept of object-relational mapper (ORM). Even though everything is up and running using this, I think I could use more reading / practice on that topic to get more familiar.

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

    Answer: Yes totally. I feel like I've acquired enough good knowledge to understand how Python work in general, and to be able to code projects on my own using this language (which was one of my goals at the beginning of this phase).

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

    Answer: In the final step of Achievement 2, I'll take a moment to visualize everything that's needed / what are the technical requirements, and I'll draw a schema representing all the app requirements to be coded. I feel like this could be of a great help / great reference to implement all the different functionalities that are going to be expected. This could be particularly useful, if I'm thinking back to the final step of Achievement 1.

Pre-Work: Before Starting Django

- Was your study routine effective during Achievement 1? If not, what will you do differently during Achievement 2?

  Answer: Yes, I think it was pretty effective. First, I was reading all the exercise text content. Then I was taking a small pause, to later jump on the exercise task. This allowed me to take the time to assimilate and think about everything I've read before putting it into practice concretely.

- 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?

  Answer: Most proud of: being able to build the entire project almost all on my own, without relying too much on external help such as Github or Stack Overflow. This has shown me that I understand general Python logics pretty well up to this point. In Achievement 2, I will try to keep continuing on this path, using my troubleshooting technics to fix any problems / issues I might encounter along the way (without however depriving myself of external support resources when necessary).

- 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?

  Answer: At first, in Achievement 1, I had some issues with some nested blocks of code that were not behaving the way I wanted (because of their nested situation within other blocks of code). To avoid this situation again, I'll be aware right away of the influence that some blocks of code can have on other nested blocks of code, and I'll maintain my code well formatted all the time during the development process. This will allow me to have a better visual on what's nested within what (and where), to better fix any issues that might happen related to that.

Exercise 2.1: Getting 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?

   Answer: **Some advantages of using plain vanilla Python:**

   - It provides a lot of flexibility on how the project can be designed. Developing with plain vanilla Python doesn't restrict developers with specific/necessary steps to follow in an exact manner, like it's the case when using Django framework.

- Developers can import and use specific features only when they truly need them for their project, instead of having right away a lot of out-of-the-box solutions like it is the case when using Django. If the project only requires some specific features, it might not be useful to have Django's whole package, especially since Django's use of prewritten code means its more server intensive, making it heavy on low-bandwidth systems.

Some disadvantages of using plain vanilla Python:

- Everything has to be built from scratch, which can make the development process longer.
- Python doesn't specifically rely on the DRY principle like Django does, which could lead to codes that not as much non-repetitive, non-redundant, and efficient as the codes written with Django.

Some advantages of using Django:

- Django's MVT architecture ensures development is fast and easy.
- Django is built with security in mind, allowing secure-by-design implementation with built-in automated encryption.
- Django is open source and has a huge community of contributors. As such, it's usually very easy to get support if needed.

Some disadvantages of using Django:

- Django is a highly structured web framework that does things a certain way. In a Django project, certain steps have to be taken in an exact manner. Django's firm structure can add an element of difficulty when building a project, as there's no room to diverge from the rules.
- Django offers a lot of out-of-the-box solutions. An application that doesn't require certain features—such as database access or file management—may not need Django for implementation.
- Django's use of prewritten code means it's more server intensive, making it heavy on low-bandwidth systems.

2. In your own words, what is the most significant advantage of Model View Template (MVT) architecture over Model View Controller (MVC) architecture?

Answer: The fact that in MVT architecture, less codes are usually required to produce same / similar results. For example, for a database with a list of records that we would like to display to users, we would have to, with MVC architecture, (1) write code to fetch the list from the database, (2) write HTML and CSS to display the list, (3) map the list to the URL of the web application and (4) send the list to the user.

However, with an MVT architecture, we wouldn't have to write code to fetch the data from the database and map it to the URL. We would simply need to specify which items to present to users, and the framework (template) will prepare and send it.

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?

      Answer - 1st goal: I want to have a very good base about core Django principles, in order to be able to discuss on Django with very good level of confidence if someone was to ask me questions such as:

        - What is Django?
        - What are the main characteristics of Django?
        - In which scenarios / projects Django would be the best tool, and why?
        - Why would you use vanilla Python vs Django for a specific project (and vice versa), and why?

    - What do you want to get out of this Achievement?

      Answer - 2nd goal: By the end of this achievement, I want to know enough about Django to be able to identify what personal projects I could build using this framework, and have a good idea of how the different features of such projects could be implemented using Django.

    - Where or what do you see yourself working on after you complete this Achievement?

      Answer - 3rd goal: If my learnings throughout this Achievement confirm me that Django could be a great tool for this, I would like to build web apps for online stores using Django framework after this Achievement. Among other features, I would want these web apps to allow clients to customize the products they would like to buy based on a selected range of options.

Exercise 2.2: Django Project Set Up

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.

   Answer: The four main terms I would point out in this case are: (1) project, (2) apps, (3) configuration files and (4) database.

Project: The whole company's website would be the project in Django term, as it represents the complete application structure consisting of multiple modules (apps), configuration files, and a database.

Apps: Each particular element within the project (website) that allows it to run and accomplish different functions / tasks would be the apps in Django term. If the website has individual modules like login, services display, online events, or a blog for example, they would all be the apps of the project.

Configuration files: Everything related to managing the website from an admin point of view, such as providing pages or interfaces for admin login, setting up global / module-specific variables, creating links between different views and templates, and creating databases, would be considered to be the configuration file in Django terms.

Database: In Django terms, that would represent the technology used to store the project data (ex: mySQL, SQLite, PostgreSQL, MongoDB…).

On top of that, I would add that frequently, there's a high level of interactivity within all these different elements in a project (ex: app–database interaction, app–app interaction, configuration files–database interaction)

2. In your own words, describe the steps you would take to deploy a basic Django application locally on your system.

Answer: I would first run migrations to create a database by using the command `py manage.py migrate`. I would then deploy the project using the command `py manage.py runserver`. To ensure everything is working properly, I would go on the server link generated automatically in my command prompt after running the last command, and I would then be able to confirm that the operation was successful depending on the message shown in my web browser.

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

Answer: Django admin site is free and comes ready-to-use with Django. Along the recipe application development, the Django admin site will be used to :

- Register the models created previously to make them appear in the admin interface
- Based on the models registered/created:
    - Create, view, update, and delete data (from the database) that will constitute the content of the web application, to eventually display them on the public version of the web app when the views and the templates will be created

Exercise 2.3: Django Models

Reflection Questions

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

   Answer: Django models are used to define the structure of database tables and table fields (with different constraints for some table fields when needed). They are the SQL database developers use with Django, and they are a subclass of django.db.models.Model. Generally, each model maps to a single database table.

   To create a model, developers generally have to create an app within the project, and then go inside the *model.py* file of this app to define the model along with his attributes. They can then register the model in the *admin.py* file of that same app.

   Django models provide different benefits, one of the most notable one being the simplicity they bring. They allow storing data in a database conveniently and simply, following specific patterns determined and based on the project needs, which assure data consistency and relevance.

   For example, a "recipe" model and a "users" model would both create one table within a database, and each of these tables would have, as columns / fields, the attributes specified in each model.

   A model for "recipe" could, for example, have attributes such as "name", "cooking_time" and "description", which would all translate as a column / field in the "recipe" table of a database.

   Each row of a table represents a specific object. In the case of the "recipe" table given as an example, one row could be associated with the object "pizza" (and his associated information such as "cooking_time" and "description").

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.

   Answer: Writing tests at the early steps of a project development can help to catch bugs as they occur, and make it easier to fix them rapidly, which could ultimately save time.

   Developing tests early in the process is also beneficial because it encourages developers to think from the beginning about the actual requirements and expected results from the application or project. This helps to prevent making errors when developing the application or the project.

   Finally, having tests ready to use early in the process can also help developers to have more time for the application or project development itself, since they won't have to repeatedly check everything manually as changes are made in the product during the development. It also makes the whole code more reliable.

Exercise 2.4: Django Views and Templates

Reflection Questions

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

   Answer: Django views are responsible for handling web requests and returning web responses. There are two types of views in Django: function-based views (FBV) and class-based views (CBV). FBV involves defining views as Python functions, while CBV uses Python classes. The web responses sent back to users (for both types of views) can take different forms, such as HTML contents of a web page or JSON data.

   The logic to **render** the views is usually put in a file called *views.py* located in an app folder, while the logic to **define** what's going to be rendered on the UI is usually defined in a *templates* folder hosting, for example, an HTML document.

   Here's a real case example:

   1- To render a welcome page for a recipes application built with Django, we would first define a Python function in the *view.py* file of the *recipes* app from the project. This function (named "home") would take two arguments: the request coming from the web, and the path to the template to be returned as a response.

   2- We would then create this template, including everything we would like to display on the welcome page, using HTML.

   3- Once the view and template are in place, we would need to specify the URL that will be used to call that particular view. To do so, some additional logic would have to be implemented in a *urls.py* file in the *recipes* app directory.

   4- As a final step, we would have to register the URL specified in step 3 and the view in the *urls.py* file of the main root project.

   This process would allow creating a view and a template for an app welcome page, and mapped it to a specific URL that would be used to call it.

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?

   Answer: I would use Django class-based views (CBV), for the following reasons:

- CBVs are based on Python classes, and provide a way to create views that, once created, are easy to reuse and extend.
- Due to their class-based nature, CBVs reduce the effort spent duplicating or rewriting code.
- Overall, CBVs are a better option when in need of a generic view that must be reused across several apps, as one can avoid code duplication (using built-in views).
- FBVs wouldn't be the best option, since they are more verbose and harder to use in cases where code needs to be reused or extended.

3. Read Django's documentation on the Django template language and make some notes on its basics.

   Answer: Django template language (DTL) is the language syntax of the Django template system. DTL is the primary way to generate output from a Django application. A template, which is a text file, contains the static parts of the desired HTML output as well as some special syntax describing how dynamic the content will be.

   A Django project can be configured with one or several templates. The main elements in DTL are variables and tags. However, the syntax of the DTL involves four different constructs:

   - Variables
     - When the template engine encounters a variable, it evaluates that variable and replaces it with the result.
   - Tags
     - Tags (ex: if, for) are more complex than variables: some create text in the output, some control flow by performing loops or logic, and some load external information into the template to be used by later variables. The template tags are a way of telling Django that there is something else than plain HTML.
   - Filters
     - Filters can be used to modify variables for display.
   - Comments
     - Comments are used to comment-out part of a line in a template.

In summary, DTL allow developers to create dynamic *and da*ta-driven web pages efficiently.

Exercise 2.5: Django MVT Revisited

Reflection Questions

1. In your own words, explain Django static files and how Django handles them.

   Answer: Static files are used to support HTML Django templates, with files such as CSS or JavaScript. Static files can be used to add more features or more dynamic behaviors to HTML

templates. Once static files have been written, they can they be loaded in the HTML templates via the command {% load static %} (at the top/beginning of the HTML) to ensure that the web page integrate and display the code from the static files.

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

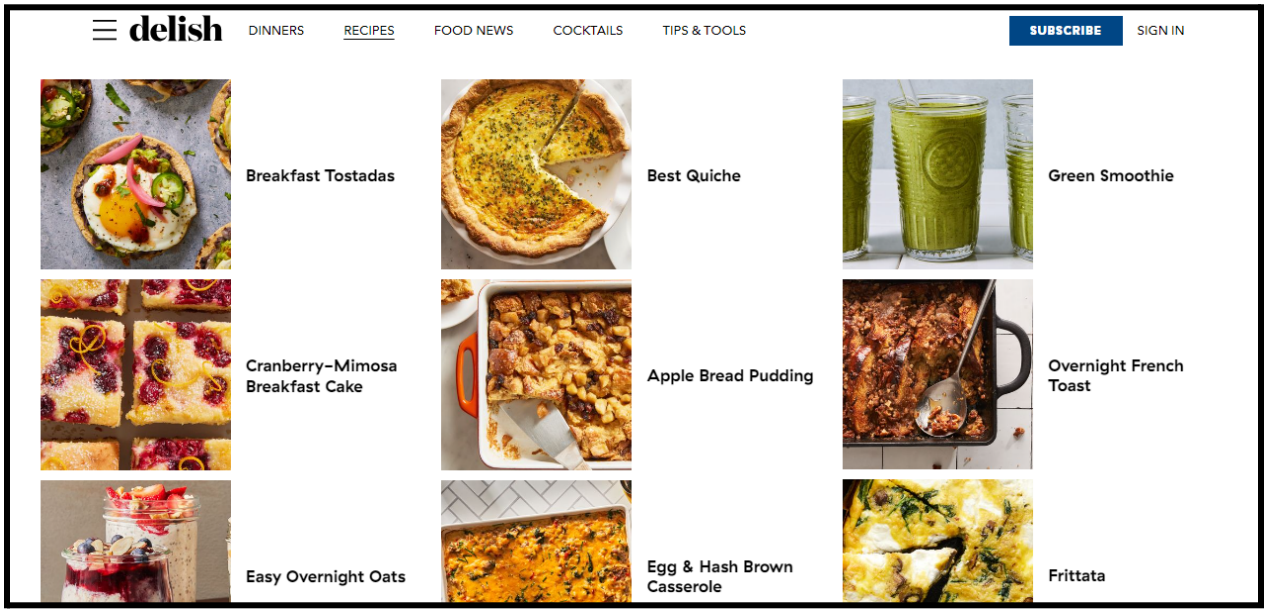
Package	Description
ListView	ListView is a class-based generic list view. It displays a page representing a list of objects, so it is useful to use when we need to show multiple instances of a table in the database.
DetailView	Often used in pair with ListView, DetailView is a way to display a single instance of a table in the database in more details. It is also another Django class-based generic view.

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.

   So far so good. I'm really happy with the required and bonus features I've added in my web app so far, since it makes it similar to real popular recipe apps. I'm also happy to start getting used to the process involving using different files in parallel to make sure each page is rendered correctly (urls.py, admin.py, models.py, etc). However, I think that I could be more familiar with how to write tests to validate the implementation of different features. This is something I've spent a little less time on, although all the tests I've written so far work well. This is something I will work on to improve in the future.

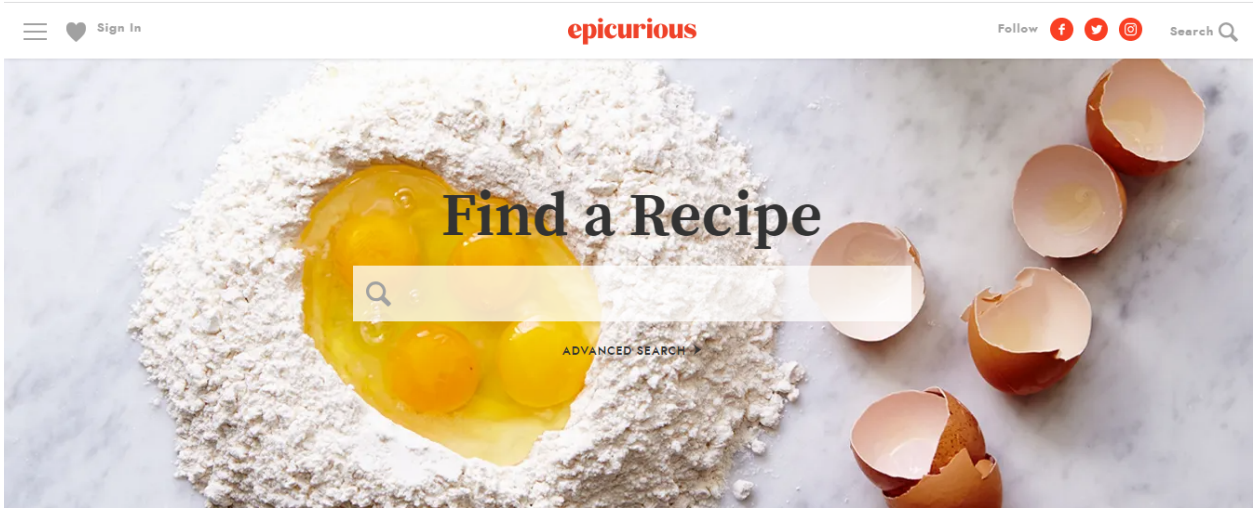
Recipe apps frontend inspirations

First interesting website / web app : Delish



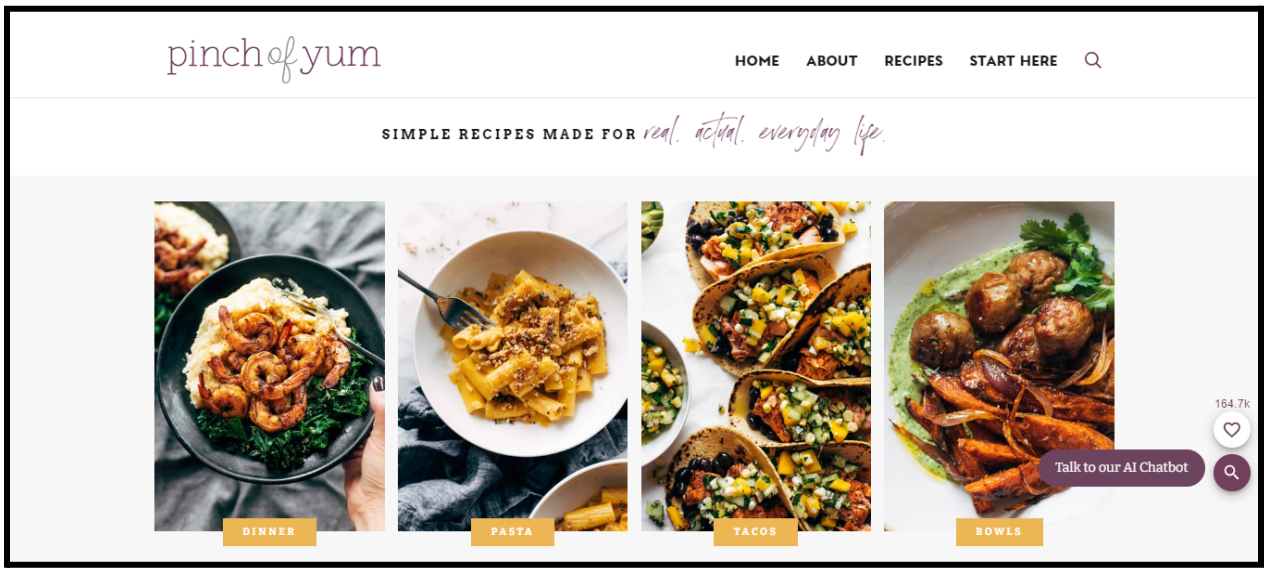
What I like the most: I like the main page, where all the recipes are displayed 3 by row. It's simple and it gives a rapid overview of the available recipes. The page is not too crowded with text either, which is another good point.

Second interesting website / web app : Epicurious



What I like the most: The simplicity of the presenting page. The UI is not too crowded, and a search bar is provided to search directly for recipes. I also like the fact that there is a big focus on the image.

Third interesting website / web app : [Pinch of yum]



What I like the most: The simplicity of the page. The focus is on the images, and the colors are coherent. I also like the different fonts use, since they add a more artistic touch.

Exercise 2.6: User Authentication in Django

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.

   Answer: Authentication is important to implement because it can be used to restrict certain pages or content to specific users. It allows websites to change their display according to whether a user is registered or not. For example, with authentication, a website can display dashboard/views/content that is unique to each user, which can make the UX more personalized, accurate and dynamic.

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

Answer:
1- Go to the project-level folder, and create a view.py file.
2- In this file, define a function handling the logic (e.g.,: username and password) to allow users to login (using some Django built-in features such as Django authenticate libraries and form).
In this step, it's important to ensure to specify the correct path where the successful logged-in users should be brought, as well as the redirecting path / error message for unsuccessful login.
3- Create an HTML view in a template folder, where we print the login form received from the view.py file. This allows users to have a visual on their UI in which they'll be able to login.
4- Register my login URL in my urls.py file at the project-level.
5- In order to ensure that the pages are only available to logged-in users, we have to protect them using the *LoginRequiredMixin* on Class-Based View (CBV) and *login_required* decorator on Function-Based View (FBV).
6- Finally, add some styling, and run some tests to ensure that the login form is working as expected, and that the pages only available to logged-in users are not accessible otherwise.

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()	Django authentication provides both authentication and authorization together and is generally referred to as the authentication system. Briefly, authentication verifies a user is who they claim to be, and authorization determines what an authenticated user is allowed to do.
redirect()	The redirect() function returns an *HttpResponseRedirect* to the appropriate URL for the arguments passed. The arguments could be: • A model: the model's get_absolute_url() function will be called. • A view name, possibly with arguments: reverse() will be used to reverse-resolve the name. • An absolute or relative URL, which will be used as-is for the redirect location. It basically redirects the user to another specified URL.
include()	Whenever Django encounters include(), it chops off whatever part of the URL matched up to that point and sends the remaining string to the included URLconf for further processing. It delegates URL handling to another URL configuration (ex: URL specified in a project' app urls.py file). For example: path('recipes/', include('recipes.urls')) When Django reach recipes/, it will then go look into the *recipes app urls.py* to complete the URL path, based on a specified logic.