Python for Web Developers

Learning Journal

Objective

We find that the students who do particularly well in our courses are those who practice metacognition. Metacognition is the art of thinking about thinking; developing a deeper understanding of your own thought processes. With the help of this Learning Journal, you’ll broaden your metacognitive knowledge and skills by reflecting on what you learn in this course.

Thanks to this Learning Journal, when you finish the course you’ll have a complete and detailed record of your learning journey and progress over time. We really recommend that you take the time to complete this Journal; students do better in CF courses and in the working world as a result!

Directions

First complete the pre-work section before you start your course. Then, once you’ve begun learning, take time after each Exercise to return to this Journal and respond to the prompts.

There will be 3 to 5 prompts per Exercise, and we recommend spending about 10 to 15 minutes in total answering them. Don’t overthink it—just write whatever comes to mind!

Also make sure that, once you’ve started filling this document in, you upload it as a deliverable on the platform. This is so that your mentor can also see your Journal and how you’re progressing over time. Don’t worry though—what you write here won’t affect how you’re graded for the Exercise tasks. The learning journal is mostly for you and your self-evaluation!

Pre-Work: Before You Start the Course

Reflection questions (to complete before your first mentor call)

• 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?

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

• 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.

Remember, you can always refer to Exercise 1.4 of the Orientation course if you’re not sure whom to reach out to for help and support.

Exercise 1.1: Getting Started with Python

Learning Goals

• Summarize the uses and benefits of Python for web development

• Prepare your developer environment for programming with Python

Reflection Questions

• 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 deals with developing the client-side of an app, or the side the user interacts with. Backend web development deals with the server side of an app, or the behind the scenes of an application. If working on backend applications, one might expect to work with things like APIs, form validation, user authentication, etc.

• Imagine you’re working as a full-stack developer in the near future. Your team is asking for your advice on whether to use JavaScript or Python for a project, and you think Python would be the better choice. How would you explain the similarities and differences between the two languages to your team? Drawing from what you learned in this Exercise, what reasons would you give to convince your team that Python is the better option?

*(Hint: refer to the Exercise section “The Benefits of Developing with Python”)*

While Python and JavaScript are similar in that they both are scripting languages, Python may be the better option to get a project started more efficiently. Because of the available libraries and pre-installed operations, Python presents the opportunity to get an app started a lot quicker than JavaScript.

• 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?

I want to learn more about the commands of Python, and how to put them together to create scripts. Knowing the syntax is key to creating my own scripts. I want to understand how Python is used in web applications, and how I can use Python to develop better apps. I eventually want to be able to develop my own scripts that are capable of whatever I can think of.

Exercise 1.2: Data Types in Python

Learning Goals

• Explain variables and data types in Python

• Summarize the use of objects in Python

• Create a data structure for your Recipe app

Reflection Questions

• Imagine you’re having a conversation with a future colleague about whether to use the iPython Shell instead of Python’s default shell. What reasons would you give to explain the benefits of using the iPython Shell over the default one?

The iPython shell is interactive, making it easier to debug code as it is being written. It also allows for testing snippets of code, cutting down on testing time later on.

• 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.

1. Integer: whole numbers, positive or negative; scalar

2. String: a sequence of characters (same as JS); scalar

3. List: a collection of objects in order; non-scalar

4. Dictionary: key-value pair collection; 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.

One of the biggest differences between lists and tuples are their mutability. Lists are mutable, aka their contents can be modified. Tuples, on the other hand, are immutable, aka their contents cannot be modified.

• 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.

I would use a dictionary for this language-learning app. Dictionaries provide key-value pairs, which would make it easy for users to enter information and retrieve it later. A tuple wouldn’t suffice due to the fact it is immutable, users wouldn’t be able to modify flashcards easily. Lists don’t have key-value pairs, making it hard to associate a vocab word with its respective definition. To store multiple flashcards, a list would be the best (a list of the flashcard dictionaries).

Exercise 1.3: Functions and Other Operations in Python

Learning Goals

• Implement conditional statements in Python to determine program flow

• Use loops to reduce time and effort in Python programming

• Write functions to organize Python code

Reflection Questions

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

• The script should ask the user where they want to travel.

• The user’s input should be checked for 3 different travel destinations that you define.

• If the user’s input is one of those 3 destinations, the following statement should be printed: “Enjoy your stay in \_\_\_\_\_\_!”

• If the user’s input is something other than the defined destinations, the following statement should be printed: “Oops, that destination is not currently available.”

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

Destination = input(“where would you like to travel?”)

destination\_1 = “paris”

destination\_2 = “tokyo”

destination\_3 = “new york”

if destination == destination\_1:

print(“enjoy your stay in “ + destination\_1 + “!”)

elif destination == destination\_2:

print(“enjoy your stay in “ + destination\_2 + “!”)

elif destination == destination\_3:

print(“enjoy your stay in “ + destination\_3 + “!”)

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 allow us to combine multiple conditions. The 3 main logical operators are and, which returns true if both conditions are met, or, which returns true if at least one condition is met, and not, which returns true if the condition is false.

• What are functions in Python? When and why are they useful?

In python, functions are reusable snippets of code that perform something. This allows for cleaner code, avoiding repetition and increasing efficiency. You should use functions if your script will be performing the same task multiple times.

• In the section for Exercise 1 in this Learning Journal, you were asked in question 3 to set some goals for yourself while you complete this course. In preparation for your next mentor call, make some notes on how you’ve progressed towards your goals so far.

I’ve learned more terminology for python, and things like functions and logical operators, which are necessary for creating my own scripts. I need to practice how to use these more, so that I am able to eventually use them in my own original scripts. I can create scripts that do basic tasks using logical operators and functions.

Exercise 1.4: File Handling in Python

Learning Goals

• Use files to store and retrieve data in Python

Reflection Questions

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

File storage is extremely important because it allows data to persist even after the program finishes running. Without file storage, any data created would be lost once the terminal closed and the program ended. If you didn’t store local files, you would have to re-enter data every time a program runs.

• 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 python objects that are stored in a binary format. You should use pickles when saving objects like dictionaries, because it enhances performance (pickles are faster than JSON).

• 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?

os.getcwd() and os.chdir()

• 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?

If I was worried there may be an error in a block of code, I would use the try-except block in order to catch any potential errors and prevent the entire script from terminating.

• 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’m really proud of the recipe storage/search system built in this exercise. I still need to improve my python vocabulary, and plan on using flashcards to better remember what they do and when to use them. Im excited to continue learning and improving!

Exercise 1.5: Object-Oriented Programming in Python

Learning Goals

• Apply object-oriented programming concepts to your Recipe app

Reflection Questions

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

Object-Oriented Programming involves centering your programming around objects and classes. OOP makes it easier to reuse code, create complex scripts, and makes an app more scalable and maintainable.

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

Classes are templates that define attributes and methods for objects, and objects are instances of a class. A real world example could be a library; the Book class defines properties like title, author, etc. and methods like read(). Each book, or object, follows the same structure but has unique data.

• In your own words, write brief explanations of the following OOP concepts; 100 to 200 words per method is fine.

**Method**

**Description**

Inheritance

Inheritance is when a new class, the child class, obtains properties and methods from a class that already exists, or the parent class. This allows for code reuse, as well as a scalable way to define relationships between objects. Inheritance reduces redundancy, improves maintainability, and allows for more efficient coding.

Polymorphism

Polymorphism is when methods are allowed to have the same name but different meanings based on the object type, meaning a method can have the same name but behave differently depending on the object. This allows for dynamic behavior and code generalization, without altering the original.

Operator Overloading

Operator Overloading allows built in operators to be customized, allowing objects to behave like built-in data types. This improves code readability and usability. Operator Overloading is useful for math operations, comparisons, and for creating customized behavior for objects while keeping the code clean.

Exercise 1.6: Connecting to Databases in Python

Learning Goals

• Create a MySQL database for your Recipe app

Reflection Questions

• What are databases and what are the advantages of using them?

Databases are collections of organized information. They make it easy to retrieve and update information, especially from large collections of info. They are also useful for ensuring an app’s scalability, by handling growing datasets efficiently.

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

INT: A whole number without decimals

VARCHAR: A string variable length that will allow up to a certain defined number of characters

FLOAT: A number including decimals

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

SQLite would be more beneficial for smaller apps, or for mobile apps. SQLite is serverless, which is why it is great for mobile apps. It’s more efficient, but is not multi-user compatible, and isn’t great for data security or scalability.

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

One major difference between python and javascript is that javascript runs in browsers or node.js, while python runs on the server. Javascript is better for web apps, because it is more efficient and dynamic. Python is better for data-focused apps, where storing, retrieving, and updating data is the main goal of the app.

• 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?

Some limitations of python include the fact that it would not be good for frontend development, and it is slower than other languages like javascript.

Exercise 1.7: Finalizing Your Python Program

Learning Goals

• Interact with a database using an object-relational mapper

• Build your final command-line Recipe application

Reflection Questions

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

An ORM is a tool that allows you to interact with a database using an OOP approach instead of writing SQL queries. You can perform database operations using python objects instead of SQL queries.

• 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?

I was able to implement CRUD operations in my app using python and SQL. My app functions as intended and is straightforward and easy to use. If I were to improve it, I would handle more edge cases to make for a better user experience. I would also add a category to make the app more useful; instructions.

• 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.

I recently created an app using python and SQL that allows users to create, view, search, edit and delete recipes, which are all stored in a SQL database. I was able to make data manipulation easier and more efficient with the use of an ORM. This project strengthened your abilities to work with databases in python.

• You’ve finished Achievement 1! Before moving on to Achievement 2, take a moment to reflect on your learning in the course so far:

• What went well during this Achievement?

I was able to build a python app that uses a SQL database. I was able to practice OOP in python.

• What’s something you’re proud of?

Im proud of implementing the search function for users. I was a bit worried about it before implementation, but I feel accomplished after having successfully coded it.

• What was the most challenging aspect of this Achievement?

I encountered a few errors when downloading SQLalchemy, which proved challenging and frustrating. After some research, I realized I was not the only one facing this issue, and followed the advice of a few fellow developers. This allowed me to successfully download SQLalchemy.

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

Yes! I feel a lot better about my python skills than I did at the beginning of this achievement.

• What’s something you want to keep in mind to help you do your best in Achievement 2?

I want to remember to handle edge cases. There were a few times my app crashed due to unforeseen errors, and I could see how that would be annoying as a user.

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

Pre-Work: Before You Start Achievement 2

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

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

• Reflect on your learning and project work for Achievement 1. What were you most proud of? How will you repeat or build on this in Achievement 2?

• What difficulties did you encounter in the last Achievement? How did you deal with them? How could this experience prepare you for difficulties in Achievement 2?

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

Remember that can always refer to Exercise 1.4 of the Orientation course if you’re not sure whom to reach out to for help and support.

Exercise 2.1: Getting Started with Django

Learning Goals

• Explain MVT architecture and compare it with MVC

• Summarize Django’s benefits and drawbacks

• Install and get started with Django

Reflection Questions

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

Vanilla python allows for more control and flexibility over your app, but you would have to write everything from scratch. Developing in vanilla python would also be a lot slower. Django offers a multitude of built-in components, speeding up development and leading to less errors.

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

MVT architecture provides a clearer separation between, the data, business logic, and presentation layers. It allows for easier updates to the front end without interfering with how data is managed.

• Now that you’ve had an introduction to the Django framework, write down three goals you have for yourself and your learning process during this Achievement. You can reflect on the following questions if it helps:

• What do you want to learn about Django?

I want to learn how to build full web applications with Django, I feel like it is a necessary skill.

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

I want to gain confidence in using Django’s tools/features, like the admin interface, security tools, and user authentication tools.

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

I want to work on more complex Django projects, and continue to learn how to integrate Django into my own applications.

Exercise 2.2: Django Project Set Up

Learning Goals

• Describe the basic structure of a Django project

• Summarize the difference between projects and apps

• Create a Django project and run it locally

• Create a superuser for a Django web application

Reflection Questions

• Suppose you’re in an interview. The interviewer gives you their company’s website as an example, asking you to convert the website and its different parts into Django terms. How would you proceed? For this question, you can think about your dream company and look at their website for reference.

(*Hint: In the Exercise, you saw the example of the CareerFoundry website in the Project and Apps section.*)

The overall site is the project, so for Figma, figma.com is the project. The project contains global settings, URLs, etc. Each app is a component of the site, like the homepage, resources page, and contact/support. Each app has its own models, views, templates, and URLs.

• 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 app on my system, I would first start by creating a virtual environment and installing Django. Using Django-admin startproject I would create the project. Then, I would run migrations and run the server. I would also create a superuser for admin.

• 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 allows developers and staff to easily manipulate data stored in the database without the need for frontend code. I’ll use it to test my models and the relationships between them, as well as managing users/staff/anyone with access to the site.

Exercise 2.3: Django Models

Learning Goals

• Discuss Django models, the “M” part of Django’s MVT architecture

• Create apps and models representing different parts of your web application

• Write and run automated tests

Reflection Questions

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

Django models are python classes that define the structure of database tables. With Django, you can interact with the database using python instead of SQL. Migrations are how you make changes to the database, and they keep track of updates made. Along with these benefits, it makes it easier to organize and understand code.

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

Writing test cases from the beginning of a project helps catch bugs before they become big problems. It notifies you if a new addition to your code breaks other parts of your code. It saves a lot of time in the long run to write tests from the beginning.

Exercise 2.4: Django Views and Templates

Learning Goals

• Summarize the process of creating views, templates, and URLs

• Explain how the “V” and “T” parts of MVT architecture work

• Create a frontend page for your web application

Reflection Questions

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

Django views are the liaison between what the use sees and the app’s data. For example, if the user visits http://sitename.com/about, Django checks the URL, calls the about view, which returns an HTML page.

• 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?

If I know I’ll be reusing a lot of the same logic, I would use class-based views. They are more structured, and make it easier to reuse code across multiple pages, by using inheritance to share logic.

• Read Django’s documentation on the Django template language and make some notes on its basics.

- Use {{ variable }} to display dynamic data

- Use {% tags % } for loops and conditionals

- {% include %} lets you reuse parts of a template

Exercise 2.5: Django MVT Revisited

Learning Goals

• Add images to the model and display them on the frontend of your application

• Create complex views with access to the model

• Display records with views and templates

Reflection Questions

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

In Django, static files are files that aren’t in python, like CSS or Javascript files, as well as images that are not dynamic. {% load static %} is used to configure these files and link them properly. Static files should be kept in a static/ folder at each app level.

• 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 displays a list of objects from a model.

DetailView

A class based view that displays the details of a single object from a model.

• 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.

Im starting to feel a lot more comfortable with Django, especially with how views, templates, and URLs all connect. I’m really proud of the recipe list and home page I currently have, they are starting to look like a real application. I’m looking forward to getting more practice with Django.

Exercise 2.6: User Authentication in Django

Learning Goals

• Create authentication for your web application

• Use GET and POST methods

• Password protect your web application’s views

Reflection Questions

• 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 how an application verifies who a user is. It protects user-specific content and ensures only users can access certain features.

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

1. Use Django’s built-in authentication form

2. Create a view to handle the login

3. Protect views using @login\_required

• 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()

Checks if the username and password provided match a valid user in the database. If the credentials are correct, it returns a user object, otherwise it returns none.

redirect()

Redirects the user to a different URL/view after an action.

include()

Allows you to reference other URLconfs in urls.py. It’s used to keep URL patterns organized by splitting them across multiple apps.

Exercise 2.7: Data Analysis and Visualization in Django

Learning Goals

• Work on elements of two-way communication like creating forms and buttons

• Implement search and visualization (reports/charts) features

• Use QuerySet API, DataFrames (with pandas), and plotting libraries (with matplotlib)

Reflection Questions

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

Spotify collects data on what songs users play, skip, save and add to playlists. It tracks listening times, device types, and user location. This helps Spotify personalize recommendations and target ads.

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

QuerySets don’t hit the database until they are evaluated. Evaluating is done by iteration, conversion to list, slicing, or using the len(), bool(), or .count() functions.

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

QuerySets are built into Django, only queries when needed, and is directly connected to database models. They are not built for advanced data analysis. DataFrames are great for larger scale data processing, support statistical operations, grouping, and pivoting, and are integrated with libraries for visualization. They add an extra dependency and can take up large amounts of memory.

Exercise 2.8: Deploying a Django Project

Learning Goals

• Enhance user experience and look and feel of your web application using CSS and JS

• Deploy your Django web application on a web server

• Curate project deliverables for your portfolio

Reflection Questions

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

In Django, you can use css and javascript by putting the files in a static folder. The {% load static %} tags help to link these files inside the html.

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

- set debug = false and configure allowed hosts

- use a production database

- set up static files

- set up environment variables

- push code

- deploy app

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

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

• What went well during this Achievement?

I steadily improved my Django skills and understanding of models, templates, and views.

• What’s something you’re proud of?

I’m proud that I fully developed a working Django app.

• What was the most challenging aspect of this Achievement?

Troubleshooting deployment issues on heroic and managing static and media files was the hardest part.

• Did this Achievement meet your expectations? Did it give you the confidence to start working with your new Django skills?

Yes! It made Django feel less intimidating.

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