CAREERFOUNDRY

Python for Web Developers Learning Journal

Objective

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

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

Directions

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

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

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

Pre-Work: Before You Start the Course

Reflection questions (to complete before your first mentor call)

1. What experiences have you had with coding and/or programming so far? What other experiences (programming-related or not) have you had that may help you as you progress through this course? I have no prior personal experience with coding or programming. However, I have access to a vast amount of programming-related knowledge and can provide assistance with a wide

range of programming topics and tasks. My training data includes a diverse set of programming and software development content, allowing me to offer guidance, code examples, explanations, and help with troubleshooting code.

2. What do you know about Python already? What do you want to know? I'm just starting out, so I don't know anything about Python yet. I've heard that Python is easy to read and use. I want to learn the basics like how to write code and do simple tasks. As I get more comfortable, I'll explore more advanced stuff. My main goal is to learn and get better.

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. As a beginner, I expect challenges in learning Python,especially as I noticed a lack of information in the exercise. To tackle them, I'll dedicate regular time, use online resources, practice, seek help, stay patient, and keep a productive environment. These steps should help me progress and overcome any gaps in information

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

Exercise 1.1: Getting Started with Python

Learning Goals

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

Reflection Questions

1. In your own words, what is the difference between frontend and backend web development? If you were hired to work on backend programming for a web application, what kinds of operations would you be working on? Frontend is about creating what users see and interact with on a website, like the design and buttons. Backend handles the stuff behind the scenes, such as managing data, handling user logins, and making sure everything works smoothly. If I worked on the backend, I'd be setting up the server, managing the database, writing code for features like user logins, and ensuring the website runs well and stays secure.

2. Imagine you're working as a full-stack developer in the near future. Your team is asking for your advice on whether to use JavaScript or Python for a project, and you think Python would be the better choice. How would you explain the similarities and differences between the two languages to your team? Drawing from what you learned in this Exercise, what reasons would you give to convince your team that Python is the better option? *(Hint: refer to the Exercise section "The Benefits of Developing with Python")* *JavaScript is good for making things happen in web browsers, like interactive buttons and animations. But Python can do that too, and it can do much more. Python is easier to understand, even for new coders. It's like reading a book with clear, simple sentences. Python can be used for both the visible part of the website (frontend) and the behind-the-scenes work (backend). It has lots of ready-made tools, like a toolbox full of helpful gadgets, to make coding faster and easier. Plus, lots of people use Python and are ready to help, like having friends who know a lot about it. So, for our project, Python is a smart choice because it's easy, versatile, and well-supported, making our work smoother and more successful.*

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? My goals for this achievement are simple. First, I want to become really good at the basics of Python. Second, I want to use Python to build useful things, like programs that do tasks for me. Lastly, I see myself working towards full-stack development, so I want to know how Python fits into that. These goals will help me learn and get ready for future projects.

Exercise 1.2: Data Types in Python

Learning Goals

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

Reflection Questions

1. Imagine you're having a conversation with a future colleague about whether to use the iPython Shell instead of Python's default shell. What reasons would you give to explain the benefits of using the iPython Shell over the default one? The IPython Shell offers an improved interactive Python programming experience with features like tab completion, rich help system, and enhanced data analysis capabilities, making it a preferred choice over the default Python 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	Integers are whole numbers without a decimal point	Scalar
Strings	Strings are sequences of characters, such as text or symbols.	Non-Scalar
Float	Floats are numbers with decimal points or in scientific notation	Scalar
List	Lists are ordered collections of items, and each item can be of any data type, including other lists	Non-Scalar

3. A frequent question at job interviews for Python developers is: what is the difference between lists and tuples in Python? Write down how you would respond. Lists in Python are mutable, using square brackets, and are suitable for dynamic collections. Tuples are immutable, with parentheses, ideal for unchanging data. Lists offer more methods, better for modification, but consume more memory. Tuples are memory-efficient, faster for access, and can be used as dictionary keys. Your choice depends on mutability needs and data type.

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. For a language-learning app with flashcards for vocabulary memorization, a list of dictionaries is the most suitable data structure. It offers a structured way to store flashcards and allows for flexibility in handling expanding content and additional features as the app evolves.

Exercise 1.3: Functions and Other Operations in Python

Learning Goals

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

Reflection Questions

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

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

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

```
destination1 = "Paris"
destination2 = "Tokyo"
destination3 = "New York"

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

if user_input == destination1:
    print(f"Enjoy your stay in {destination1}!")
elif user_input == destination2:
    print(f"Enjoy your stay in {destination2}!")
elif user_input == destination3:
    print(f"Enjoy your stay in {destination3}!")
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. Logical operators in Python, such as and, or, and not, allow for combining and manipulating boolean values to create conditional expressions. They're fundamental for decision-making in Python code

3. What are functions in Python? When and why are they useful? In Python, functions are reusable code blocks defined with def. They enhance code organization, reusability, and readability by encapsulating specific tasks. Functions take parameters, return values, and enable modular and independent development. Their abstraction and flexibility contribute to a well-structured and maintainable codebase.

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. I'm happy to note a sense of progress and increased comfort and confidence in acquiring new skills. I feel well-prepared to tackle challenges, whether through independent problem-solving using Google or revisiting resources provided by CareerFoundry. This growing confidence is motivating and reinforces my commitment to continuous learning and skill development.

Exercise 1.4: File Handling in Python

Learning Goals

- Use files to store and retrieve data in Python

Reflection Questions

1. Why is file storage important when you're using Python? What would happen if you didn't store local files? File storage in Python is like saving your work on a computer. It helps you keep information even after you close your program. It's useful for sharing and remembering things. Without it, you might lose your work and find it hard to share or remember important details.

2. In this Exercise you learned about the pickling process with the **pickle.dump()** method. What are pickles? In which situations would you choose to use pickles and why? Pickles in Python are like magic containers for saving and later using pieces of your program. You'd use pickles when you want to remember how your program was or store big things, like machine learning tricks, so you can use them again without redoing all the work. It's like taking a snapshot of your program's memory.

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? To find out which folder you're in, you use os.getcwd() in Python. If you want to switch to a different folder, you use os.chdir(new_directory), where new_directory is the path to the folder you want to go to.

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? To prevent a Python script from stopping entirely due to an error in a specific part, you can use a "try-except" block. It's like saying, "try doing this risky thing, and if it goes wrong, do something else and keep the rest of the script running."

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. Im feel like I'm grasping pretty quick, I just need to work on my time management for these tasks.

Exercise 1.5: Object-Oriented Programming in Python

Learning Goals

- Apply object-oriented programming concepts to your Recipe app

Reflection Questions

1. In your own words, what is object-oriented programming? What are the benefits of OOP? Object-oriented programming (OOP) is a coding approach that structures software using objects, each containing data and related functions. Its benefits include code organization, reusability, security, simplicity, and flexibility through features like inheritance and polymorphism. OOP provides a clear and modular way to design software.

2. What are objects and classes in Python? Come up with a real-world example to illustrate how objects and classes work. In Python, a class is like a blueprint that describes how to create something. An object is like the actual thing you create from that blueprint. For example, imagine a blueprint for a Car. The blueprint says what a car is made of and what it can do. An object would be a specific car you build using that blueprint, like a red Toyota or a blue Ford. The blueprint (class) tells you what every car should have, but each actual car (object) can be a bit different.

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 in programming is like passing down traits from a parent to a child. In Object-Oriented Programming (OOP), it means a new class can inherit properties and actions from an existing class. Think of it as a blueprint where a new version (subclass) automatically gets the features of the original (superclass), and you can customize or add more features to suit specific needs. It's a way to make code more efficient by reusing what already works and building upon it.
Polymorphism	Polymorphism is like having a single remote control that works for various gadgets. In programming, it means using the same name for different things. There's one type where you can use the same name for different tasks (like pressing the "on" button), and another type where different things can be used in the same way (like using one remote for different devices). It makes the code simpler and more flexible, letting you do different things with a common command. It's like having one button on a remote control that does different jobs depending on what you're using it

	for.
Operator Overloading	Operator overloading in programming is like giving a plus sign the ability to add not just numbers, but also words or even images. In Object-Oriented Programming (OOP), it allows you to redefine how operators like +, -, or * work for your custom objects. For example, you can make a "plus" operation mean something different for a car object than for a number. It's a way of teaching operators new tricks based on the context. Think of it as expanding the job description of common symbols to handle different situations, like making a plus sign understand both numbers and strings.

Exercise 1.6: Connecting to Databases in Python

Learning Goals

- Create a MySQL database for your Recipe app

Reflection Questions

1. What are databases and what are the advantages of using them? Databases are organized systems for storing and managing data, offering advantages such as data integrity, efficient retrieval, scalability, and security.

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

Data type	Definition
INTEGER	Used for whole numbers, allowing storage of values without decimal points.
VARCHAR	Variable-length character string, suitable for storing alphanumeric data of varying lengths.
DATE	Specifically designed for storing date values in the format 'YYYY-MM-DD'.

3. In what situations would SQLite be a better choice than MySQL? SQLite might be a better choice than MySQL in scenarios where simplicity and lightweight functionality are prioritized. It's suitable for small to medium-sized applications, embedded systems, or situations where a standalone,

serverless database is sufficient. Additionally, if ease of setup, low maintenance, and minimal configuration are essential, SQLite can be a preferable option over MySQL.

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? JavaScript is primarily used for web development with a C-style syntax, focusing on front-end tasks, while Python, known for readability, supports diverse applications such as web development, data analysis, and machine learning with its indentation-based syntax and versatile ecosystem.

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? Python, while versatile and readable, has limitations such as slower execution speed compared to lower-level languages, challenges in mobile development, and limited capabilities for memory-intensive tasks.

Exercise 1.7: Finalizing Your Python Program

Learning Goals

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

Reflection Questions

1. What is an Object Relational Mapper and what are the advantages of using one? An Object Relational Mapper is a programming tool that allows seamless interaction between a relational database and an application by mapping database tables to object-oriented classes. Its advantages include simplifying database operations, reducing repetitive code, and providing a higher-level, Pythonic interface for developers.

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? I think it went ok. I developed a functional Recipe app with a user-friendly interface and efficient database interactions. However, I believe there's room for improvement in refining certain features to enhance the overall user experience. I think I did well organizing the Recipe app, making it easy to manage recipes. The backend setup using SQLAlchemy and the user interface for adding, editing, and deleting recipes were particularly well done.

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. I made a Recipe app using Python. It lets you add, edit, search, and delete recipes easily. I used a tool called SQLAlchemy to manage data effectively. The app is designed to be simple for anyone to use, even if you're not super tech-savvy. This project helped me get better at Python and create apps that people can actually use.

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? I felt was like the content was easier to grasp
   b. What's something you're proud of? I like that i was able to solve problems on my own
   c. What was the most challenging aspect of this Achievement? The most challenging was maintaining consistency in completing weekly tasks and achieving the goal of two tasks per week.
   d. Did this Achievement meet your expectations? Did it give you the confidence to start working with your new Python skills? Yes, I want to see how else I can apply python. I would like to explore further applications and, in fact, consider starting a career in this field.
   e. What's something you want to keep in mind to help you do your best in Achievement 2? I want to do at least an hour or two of work on some days so I can stay on track. Just difficult sometimes with my job.

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

1. Suppose you're a web developer in a company and need to decide if you'll use vanilla (plain) Python for a project, or a framework like Django instead. What are the advantages and drawbacks of each? **Vanilla (Plain) Python:** Simpler and flexible but requires manual setup for common functionalities. **Django Framework:** Rapid development and built-in features, but comes with a learning curve and some enforced structure.

2. In your own words, what is the most significant advantage of Model View Template (MVT) architecture over Model View Controller (MVC) architecture? The big advantage of MVT over MVC is that MVT keeps the part that deals with how things look separate and focused, making it easier to manage and understand.

3. Now that you've had an introduction to the Django framework, write down three goals you have for yourself and your learning process during this Achievement. You can reflect on the following questions if it helps:
    - What do you want to learn about Django?
    - What do you want to get out of this Achievement?
    - Where or what do you see yourself working on after you complete this Achievement? My goals for learning Django are to build a complete web application, and I want to feel confident using django. So far i've felt more confident in using python.

Exercise 2.2: Django Project Set Up

Learning Goals

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

Reflection Questions

1. Suppose you're in an interview. The interviewer gives you their company's website as an example, asking you to convert the website and its different parts into Django terms. How would you proceed? For this question, you can think about your dream company and look at their website for reference.
   (*Hint: In the Exercise, you saw the example of the CareerFoundry website in the Project and Apps section.*) In an interview, if they want me to turn a company's website into Django language, I'd think of each page like a "view" that shows stuff. For changing things, I'd use "models," which are like organized forms. Templates help make the page look good. For logins and web addresses, I'd use built-in tools from Django. Pictures and styles are managed using Django's simple system for files. Breaking down parts into "apps" and connecting to a database are important, and I'd think about putting the website on the internet. This shows how I'd use Django to make the website work.

2. In your own words, describe the steps you would take to deploy a basic Django application locally on your system. To deploy a basic Django application locally, activate the virtual environment, install dependencies, apply migrations, create a superuser, and run the development server. Deactivate the virtual environment when done. These steps assume a simple app structure and are intended for local development. For production, additional configurations are needed.

3. Do some research about the Django admin site and write down how you'd use it during your web application development. The Django admin site is a handy tool for managing data in web development. You register your models to easily add, edit, or delete data. Customize views, control who can access it, and use helpful features like inline models and actions. It's a quick way to handle database tasks without building a separate admin panel.

Exercise 2.3: Django Models

Learning Goals

- Discuss Django models, the "M" part of Django's MVT architecture
- Create apps and models representing different parts of your web application
- Write and run automated tests

Reflection Questions

1. Do some research on Django models. In your own words, write down how Django models work and what their benefits are.Django models are like blueprints for storing and retrieving data in a

web application. They make it easy to define what kind of information your app will save, help you interact with the database using Python, and ensure that your data stays organized and secure.

2. In your own words, explain why it is crucial to write test cases from the beginning of a project. You can take an example project to explain your answer. Writing tests from the start of a project is like making sure all the parts of a new toy work correctly before you play with it, so you don't find surprises later and can enjoy it without worries.

Exercise 2.4: Django Views and Templates

Learning Goals

- Summarize the process of creating views, templates, and URLs
- Explain how the "V" and "T" parts of MVT architecture work
- Create a frontend page for your web application

Reflection Questions

1. Do some research on Django views. In your own words, use an example to explain how Django views work. In Django, views are like the waiters at a restaurant. They take your order (receive a request), communicate with the kitchen (backend), and bring you the dish (response or webpage) you asked for.

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? If I expect to reuse a bunch of code in different places of my Django project, I'd go with class-based views. They're like well-labeled boxes for my code, making it organized and easier to reuse without creating a mess. It's just a cleaner way to handle things

3. Read Django's documentation on the Django template language and make some notes on its basics.
   Template Variables - Magic Words: In Django templates, you use template variables (those inside {{...}}) as magic words. These are placeholders that get replaced with real content when your page is shown, like filling in the blanks in a story.

   Template Tags - Special Instructions: Template tags (those inside {% ... %}) are like special instructions. They can loop through lists, make decisions, or perform other actions to add dynamic features to your pages.

Filters - Crayons with Different Shades: Filters are like crayons with different shades. They allow you to modify the appearance of your variables. For example, turning text to uppercase or formatting dates.

Template Inheritance - Coloring Book Outline: Template inheritance is like having an outline for your coloring book. You create a main template with common elements, and other templates fill in the details. It ensures a consistent look across your pages.

Template Comments - Whispering in the Margins: Template comments are like whispering in the margins of your coloring book. They don't show up on the final page but allow you to leave notes for yourself or others working on the project.

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

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

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	
DetailView	

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.

Exercise 2.6: User Authentication in Django

Learning Goals

- Create authentication for your web application
- Use GET and POST methods
- Password protect your web application’s views

Reflection Questions

1. In your own words, write down the importance of incorporating authentication into an application. You can take an example application to explain your answer.

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

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()	
redirect()	
include()	

Exercise 2.7: Data Analysis and Visualization in Django

Learning Goals

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

Reflection Questions

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

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

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

Exercise 2.8: Deploying a Django Project

Learning Goals

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

Reflection Questions

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

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

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

4. You’ve now finished Achievement 2 and, with it, the whole course! Take a moment to reflect on your learning:
    a. What went well during this Achievement?
    b. What’s something you’re proud of?
    c. What was the most challenging aspect of this Achievement?
    d. Did this Achievement meet your expectations? Did it give you the confidence to start working with your new Django skills?

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