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?
    - Before this course (Python for developers), the only recent experience with coding has been the preceding full-stack immersion course with CareerFoundry. Before that, I only had a little bit of exposure back in high school when I took the intro to computer sciences course. I don't have much coding related experience, so I plan on working through this course like I am back in school. I have only had 1 online course in college, but I generally did well in classes because I take a lot of organized notes as I follow along with the course, and I turn back to these notes when I come across any questions.
- What do you know about Python already? What do you want to know?
    - Not much at all. I only knew that Python is a very popular coding language and considered one of the easier ones to learn, so it is a good one to start with. Other than that, not much at all. There is a lot I would ideally like to know, but it is very broad and I just want to learn what I can use Python for and how to use it.
- 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.
    - I foresee times when I get stuck and am faced with errors, especially in the terminal, as I am still very uncomfortable with working in the terminal. I am worried if a command was executed incorrectly that it would significantly alter my environment settings or scripts, and I wouldn't know how to reverse the changes. What I can think of at this time to solve these upcoming challenges would be to work on the coding exercises when I have a good amount of time and quiet space to focus. I have a busy schedule, so if I am distracted, I will more likely create more errors. When I feel like I am a little rushed or short on time/energy, I should plan on focusing on reading and note-taking. Then when I have more time to allot to the course, I can sit down and attempt the coding. My mentor, who is also busy with work and family, is mainly available in the evenings (similar to myself), so I can try to work around that.

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 the interfaces that the client/end-user sees, like the webpage or application interface, including any graphics, links, text, displayed data, forms, etc. Backend web development involves the operations that happen behind the scenes. This includes interacting with servers, using APIs to interact with databases, processing user requests and data, etc. If I was hired to work on backend programming, I would imagine myself working with APIs and databases to retrieve and update information stored in databases. I'd be interested in working on creating the databases.

- 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")*
    - Python is a high-level scripting language, has easily understandable keywords, and uses dynamic typing, all similar to JavaScript. Python also boasts amazing readability in its code and has a simple built-in package management system that allows users access to a range of packages that simplify complex operations into condensed functions. The readability along with the interactive shell allows for simple debugging and reduced errors, streamlining the development process. Python frameworks also come pre-installed with common web operations that can help developers start working quickly and more efficiently.
- 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 be able to feel more comfortable navigating the terminal/command line interface
    - I want to build a strong foundation in Python by mastering the essential concepts like variables, data types, different loops, and how to build functions in Python.
    - I want to get a better sense of how I can apply what I will be learning about Python to real-world projects (see where I can apply it to).

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 overall a more user-friendly option, as it makes code easier to read due to the syntax highlighting and auto indentation. Its immediate execution of code line by line also makes it a great tool for testing out small pieces of code, which simplifies the debugging process. It also has a nice tab complete feature that allows us to see all possible commands for a given object.

❖ Python has a host of different data types that allow you to store and organize information. List 4 examples of data types that Python recognizes, briefly define them, and indicate whether they are scalar or non-scalar.

Data Type	**Definition**	**Scalar or Non-Scalar?**
int	Integer - whole numbers, both negative and non-negative with no theoretical limit	scalar
dictionary	Unordered set of items store as key-value pairs, indexed by identifiers (keys) and the values can be of any type	non-scalar
tuples	Linear arrays that can store multiple values of any type in an array-like structure	non-scalar
bool	Stores either of two values - 'true' or 'false' - useful for storing output of a checked condition	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.

➢ Lists and tuples are similar in that they store data across linear sequences; each element is indexed; and similar operations can be performed on them. They differ in that tuples are immutable, whereas lists are mutable. So while the contents of tuples cannot be modified once created, the internal elements of a list can be modified or deleted. This offers more flexibility when dealing with lists, but on the other hand this makes tuples faster to read and access as they take up less memory and overhead, especially when there is large amounts of data involved.

❖ 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 choose the dictionary data structure, because it would allow me a more flexible and structured way of organizing the information (in key-value pairs). There would be words, definitions, and categories stored, and if I stored these as individual tuples or lists, it wouldn't make much sense as they are not linked to each other properly (a list of vocabulary words wouldn't be much use if we don't know which definitions match with them). Dictionaries are also mutable, so as we continue to develop the app, we can modify the flashcards to include more information.

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("Enter where you would like to travel to (city): ")
if destination == "Seoul":
    print("Enjoy your stay in Seoul!")

elif destination == "Tokyo":
    print("Enjoy your stay in Tokyo!")

elif destination == "Hong Kong":
    print("Enjoy your stay in Hong Kong!")

else:
    print("Oops, that destination is not currently available.")
```

- ❖ Imagine you're at a job interview for a Python developer role. The interviewer says "Explain logical operators in Python". Draft how you would respond.
    - ➢ Logical operators in Python are used to combine and check for multiple conditions at the same time. They can check for whether all mentioned conditional statements are met (and operator), check if at least one of the conditional statements are met (or operator), or reverse the result with the not operator.
- ❖ What are functions in Python? When and why are they useful?
    - ➢ Functions in Python are sets of instructions that manipulate code to achieve certain tasks, and they can be built-in (like print(), and append()) or custom created by the developer. Functions are useful because they can be used to condense code where certain steps are being repeated. This reduces repetition of code, which makes the codebase cleaner and more concise while also saving the developer time.
- ❖ 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 am developing a good basic understanding of variables, data types, and functions (including loops) in Python, so that part is going well.
    - ➢ Regarding the terminal and shell, I am more comfortable with it now as compared to when I was setting up the coding environment, but I would say I am still a little anxious when using it. I keep worrying about somehow messing up the settings and the coding environment/system if I input an incorrect command. As an example, in my previous course (Full Stack Immersion), I was trying to generate documentation with TSDoc, but it wasn't quite working. As I experimented with different commands to see if something would, my whole project folder was deleted somehow. I couldn't find a way to reverse the action, so I had to clone my project from my GitHub repo and rewrite the newer portions.
    - ➢ I am still trying to understand how the command line interface/shell can be applied to real-life projects, as I don't recall people using apps or tools that work from a terminal.

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 important because after a Python script has finished running, the data and values that were stored in variables are not saved and cannot be retrieved for later use. That would make something like a recipes app useless, because we will not be able to access the stored recipes.

- ❖ 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 packaged streams of byte that are converted from complex data and written into a binary file. We would use pickles when the data we are handling have more complex data structures, like dictionaries, and text files can no longer store this data while retaining the structure of the data. These get converted in the pickling process and stored in binary files that can be read by machines and accessed when needed.

- ❖ In Python, what function do you use to find out which directory you're currently in? What if you wanted to change your current working directory?

    - ➢ I would use the os.getcwd() function to find out which directory I am currently in. The tree command is also useful in seeing a more visual representation of the directory I am working in. If I want to change my current working directory, I can first confirm the folders and files available with the os.listdir() command before using the os.chdir() to navigate to the desired location.

- ❖ Imagine you're working on a Python script and are worried there may be an error in a block of code. How would you approach the situation to prevent the entire script from terminating due to an error?

    - ➢ I would utilize the try-except block to handle potential errors. The block of code that I am worried about an error occurring in would be placed into the try block, and the following except block(s) would provide hints to what caused the error. I would also include an else block to contain code to be run if no errors occurred. With the try-except block in place, it prevents the entire script from terminating but rather highlights the existence of an error so it can be amended.

- ❖ 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 would say it is going pretty well. Python concepts and syntax have been sitting well with me as it is very logical and intuitive. The part of the lesson that mentioned using a lambda function as a key for sorting didn't make a lot of sense in the beginning. After reviewing that section a few times, I think I understand, but it's still a little fuzzy in my mind. Overall I think I just need more practice with logical problems.

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 (OOP) is a programming model that is based on the concept of objects rather than functions. In OOP languages like Python, everything is an object, and each object contains data and associated methods to interact with that data. These individual objects are created through classes, which are like reusable blueprints that represent a broader category of objects. The advantages of OOP include code reusability, which leads to simplification of complex code; code organization due to hierarchical structures; flexibility of creating class-specific behavior with help of polymorphism; and easier debugging process due to the modular nature (classes) and tools like inheritance.

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

- ➢ Classes are the reusable code templates (blueprint) for creating individual (but similar) objects. Objects are then instances of the class they were created from, and they contain data and functions used to interact with or manipulate the data. A real-world example of a class would be "Puzzle", which can have attributes like a name, size, and suitable user age. Some subclasses might be "jigsaw puzzle", "word puzzle", and "3D puzzle", which would all inherit the attributes and methods from the "Puzzle" class and then have their own differentiating attributes and methods. One object created from the "3D puzzle" subclass may be "Rubik's Cube". It would have the attributes defined in the "3D puzzle" class and then its own unique attributes as well as functions that involve its own rules and actions that can be done.

❖ 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 a concept that involves classes building off of other classes that are broader in range (encompassing the new class). It is like a family tree, where child classes (or subclasses) are created from parent classes. These child classes can inherit properties and methods from the parent class (copied over), so this information does not need to be repeated in the newly created classes.
Polymorphism	Polymorphism is the concept where an attribute or method can have the same name across different classes but can perform different operations based on where it was defined. This allows for additional flexibility, as there will be no conflict as long as the attribute or method was defined separately and exclusively.
Operator Overloading	Operator overloading is a process of defining your own methods for operators (+, -, >, <, ==, etc) in custom classes. In Python's built-in classes, the operators are supported and work in different ways depending on the data type. With custom classes though, operator overloading is needed for the operators to be supported.

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

Data type	Definition

- In what situations would SQLite be a better choice than MySQL?
- Think back to what you learned in the Immersion course. What do you think about the differences between JavaScript and Python as programming languages?
- 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?

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?
- 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?
- 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.

- 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?
    - What's something you're proud of?
    - What was the most challenging aspect of this Achievement?
    - Did this Achievement meet your expectations? Did it give you the confidence to start working with your new Python skills?
    - What's something you want to keep in mind to help you do your best in Achievement 2?

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?

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

- 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?

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.*)

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

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

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.

- 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.

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.

- 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?

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

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.

- 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	

- 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

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

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

- 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

- 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.

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

- 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

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

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

- (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?
  - What's something you're proud of?
  - What was the most challenging aspect of this Achievement?
  - 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.