

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?
 - a. I have completed the intro to frontend development and the immersion portions of the course. I think both provide a foundation to be successful in the specialization course.
2. What do you know about Python already? What do you want to know?
 - a. I know a little about it's uses, like development of AI, but I'd like to know more about it's application to specific fields or where it's commonly used.
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.
 - a. As I've already experienced in the previous courses, there can be times when I face roadblocks to progress or get stuck on a task. I need to keep persevering through the challenges to find the solutions I need to move forward.

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?
 - a. Front end development is focused on what the user sees, such as the user experience and interface, whereas backend development is focused on the server side of things, like management of a database, business logic, and configuration of a server. If I was hired to work on backend programming for a web app I would expect to work on user authentication, maintaining databases, and keeping the flow of data 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")
 - a. I would explain that Python is like JavaScript in that commands are executed line by line via standardized syntax. It's readability makes it easy to learn and understand, it has simple built-in package management, and has strong community support so there's always a solution.
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?
 - a. I want to get a better understanding of Python's applications and how it can be used.
 - b. I want to become more familiar with Python scripts and how they progress from this introduction.
 - c. I want to learn why Python may be more desirable to use than another language and when it's less desirable.

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?
 - a. iPython Shell is much more user friendly. The text is visibly clearer with syntax highlighting, you don't need to indent manually, and each command is executed immediately so you can test out code.
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?
Tuples	Linear arrays that can store multiple values of any type.	Non-Scalar
Lists	Similar to a Tuple, but are mutable, which means any of the internal elements of a list can be modified or deleted.	Non-Scalar
Strings	An immutable array of characters. They can be composed of alphanumeric characters as well as symbols and are surrounded by either single or double quotes.	Scalar
Dictionaries	Stores values and objects within itself indexed by identifiers, or keys. An unordered set of items, each of them a key-value pair, where	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.
 - a. Tuples and Lists are types of ordered sequence in Python. They differ in that Lists are mutable, or able to be modified or deleted, whereas Tuples cannot. They also differ in

syntax where Tuples use parentheses and Lists use square brackets. Tuples are more suited to situations where data does not change frequently and Lists are more flexible and can be updated.

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.
 - a. A dictionary data structure would be useful for a language-learning app that uses flashcards. It can store the words as the key and the corresponding definition as a value. This would improve the time it takes to lookup the appropriate word to study and its definition. Dictionaries are dynamic, so they can easily handle changes in entries, whether it be adding new ones or modifying existing ones. Depending on the features, other types of data structures would need to be incorporated to handle different types of data and their relationship to each other.

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!*)

```
destination = input("Where would you like to travel to? ")

if destination == "Hawaii":
    print("Enjoy your stay in Hawaii")

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

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

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.
 - a. Logical operators in python check for multiple conditions at the same time using operators such as 'and' and 'or'. In the case of 'and', an expression would return a value of 'True', when checking two conditions, if both conditions are met, or 'False' if any of the conditions are not met. The 'or' operator checks whether any of the conditions are met and only needs one condition to be met in order to return a value of 'True', otherwise it returns 'False' if no conditions are met.
3. What are functions in Python? When and why are they useful?
 - a. Functions are used when needing to repeat a specific task multiple times and saves time from having to repeatedly write the same code. They are called on to perform a specific task and can manipulate data from the script.
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.
 - a. After my initial call with my mentor I have a little more understanding of the types of applications Python can be used for. I'm gaining more of an understanding of how Python works and am excited for the next exercises as we get into building the app and learn more about Python. The exercises so far have been informative as to showing examples of how Python is used.

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?
 - a. File storage is important because data is destroyed once the file is closed if only stored in a script. Storing data in a .txt or .bin file allows data to be accessed regularly across multiple scripts.
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?
 - a. Pickles are a packaged stream of bytes. They can be used to write or read from external files. The files typically contain data types such as lists and dictionaries and are useful for reading data from binary files, which cannot be read by humans.

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?
 - a. The function `os.getcwd()` tells you which directory you're currently in and `os.chdir()` allows you to change the directory.
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?
 - a. I would use a try-except-else-finally block to 'try' a block of code where I believe the error would occur. 'Except' can tell the user what type of error occurred. 'Else' can run additional code if the 'try' block is successful. 'Finally' continues the code regardless if a 'return' statement occurs before it.
5. You're now more than halfway through Achievement 1! Take a moment to reflect on your learning in the course so far. How is it going? What's something you're proud of so far? Is there something you're struggling with? What do you need more practice with? Feel free to use these notes to guide your next mentor call.
 - a. I find myself referring back to the reading quite a bit. I am able to follow along during, but when it comes time to try it myself with the instructions in the task, I find I don't immediately know what to do or how to start. This exercise's task seemed to build on the previous one, so that was helpful to refer to the previous task and be able to understand what needed to be done. I'm sure with more practice I'll get a better hang of it. I have come across errors, but am able to quickly diagnose and fix them. They are typically silly syntax errors, which caused me trouble in the immersion course.

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?
 - a. Object-oriented programming abstracts data and methods into classes that can be reused throughout a script without needing to repeat code blocks for different methods.
2. What are objects and classes in Python? Come up with a real-world example to illustrate how objects and classes work.
 - a. An object is made up of data types that can be used in methods. Classes are similar to templates that help to quickly create multiple objects and methods. An example of a class would be the menu at a coffee shop and each drink item would be an object that contains attributes and methods for interacting with it.

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

Method	Description
Inheritance	Using attributes or methods from a parent class in a sub class
Polymorphism	An attribute or method with the same name across different classes, but performing different operations
Operator Overloading	Defining your own method for using an operator in a class

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?
 - a. Databases can store data externally rather than on your local machine. Data is stored in a standard format that is easier to access, more secure, and easy to modify.
2. List 3 data types that can be used in MySQL and describe them briefly:

Data type	Definition
INT	Standard integers
FLOAT	Floating point decimal numbers
DATETIME	Datetime values

3. In what situations would SQLite be a better choice than MySQL?
 - a. SQLite would be better when you're dealing with simple databases, such as storing customers' email addresses for an online store. SQLite does not require installation and allows you to store data in .db files.
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?

- a. Python's syntax is much simpler than JavaScript's. Python does not require variable prefixes, which keeps things simpler and easier to read. JavaScript also requires the use of semicolons to end an action whereas Python does not.
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?
 - a. From what I have learned so far, I can't say there are a lot of drawbacks or limitations to Python. I would need to continue learning more to discover what else can or can't be done with the language. I like the fact that I can write scripts and use the iPython shell to test small blocks of code. I am more in the habit of writing scripts because it's familiar relative to the work in the immersion course, so I want to continue trying out the iPython shell to get more comfortable using it.

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?
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?
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.
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?
 - 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 Python skills?
 - e. 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

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

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?

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.)
2. In your own words, describe the steps you would take to deploy a basic Django application locally on your system.
3. 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

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

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.
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?
3. 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

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.