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?

Prior to *the entire CF course*, I had limited experience with HTML and CSS thanks to my current role – mostly in WordPress and Adobe Experience Manager. Before beginning *the Python speciality course*, I completed courses in frontend and full-stack development. These covered HTML, CSS, JavaScript, React, Angular, Bootstrap and more.

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

   I know that Python is a very popular language because it's relatively easy to learn. I know that it's used by data journalists to do things like analyze large datasets and scrape websites. I also know that it's a popular language in AI/ML applications. Just getting a working knowledge of Python and a basic comfortability with it will be fantastic and beneficial.

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.

   I think that I have my studying process down pretty well, so I don't foresee any issues with that. I think the main challenge will just be readjusting my thinking to a new language in general. I will have to regularly remind myself that I'm still learning and it's OK to not know everything. I was also very comfortable talking to my tutor from earlier sections of the course, so familiarizing myself with a new tutor will be a process as well.

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 development deals with what a user sees and interacts with. It's the text and layout of the web page or app, forms, buttons, animations, etc. It's the visual and interactive elements. Backend development relates to everything that goes on in the background, from API and database development and management, to user authorization, dynamic site updates, processing user requests, communicating with servers and more.

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")*

   Python in similar to JavaScript in several ways, including its line-by-line execution and dynamic typing, which allows for variables to assume any kind of value during runtime. So the switch from JavaScript to Python would not be a difficult one to make. But Python brings with it several values that help it edge out JavaScript. These include built-in package management for things like math, password encryption and JSON formatting. It's also highly readable, which keeps code resistant to errors and easier to debug. Plus, it's strong community support means we'll have access to resources if we run into issues.

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?

   1. I want to understand the types of projects that Python is most suitable for and the types of roles looking for this coding knowledge.

   2. I want to be more comfortable navigating the Command Line Interface on my computer by gaining a better understanding of the Python commands.

   3. I want to really focus on reading the additional resources for each section, adjusting my notes and code as I learn more about Python.

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 is a user-friendly, practical shell for reading code. Syntax highlighting makes it much easier to see what is happening and code doesn't need to be manually indented, like in the built-in default python shell. iPython shells also allow you to test small bits of code easily and quickly – each command is executed immediately after typing it. This is a good way to test code before writing out and running entire script files.

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 (int)	Integers can be whole negative or positive numbers (-20, 23, 178, -182939, etc.) from zero to infinity. They do not include decimal points and can be used to represent things like counts and quantities.	Scalar
Boolean (bool)	Boolean data types store one of two values – True or False – and are useful for checking condition outputs.	Scalar
Dictionary (dict)	A dictionary stores values and objects within itself using key-value pairs. These are essentially a list of items, where each item has a name (the key) and a description (the value). Values have no immutable data type restrictions, they can even be another dictionary.	Non-Scalar
Tuple	Tuples are linear arrays that can store multiple values of any type (integers, strings, etc.). They're useful when you need to save several values, but naming a new variable for every one of them would be too much work. They are similar to lists but are immutable - you can't change, add, or remove items from them once they're created.	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 and tuples are similar in that they are essentially a sequence of items. They can include a variety of data types, like integers, floats, and strings.

The main difference is that tuples are immutable and you cannot add, change or remove items from them once they are created. In order to “add” a new item to a tuple, you’d have to place the new value in a single-element tuple, and then concatenate (i.e., merge) the two tuples together. Tuples are faster to read and access, especially when large amounts of data are involved.

Lists, on the other hand, are mutable - any of the internal elements of a list can be modified or deleted. You can also rearrange or even insert new elements in lists. They are useful in situations where reordering or modifications may be necessary.

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.

   I would choose a dictionary for the language learning app. This would be ideal because flashcards could be set up with key-value pairs that utilize the word as the key and its definition and category as the values. This structure would allow quick lookups and easy updates to add/delete new words to the dictionary and add new values (synonyms, sentence examples) down the line without altering the entire structure. Since tuples are immutable, I believe they’d be too restrictive for this app. Lists are mutable and could be useful for storing things like synonyms, but the larger set of data for a single word is complex enough that a dictionary would make more sense (with lists making up some of the values).

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 do you want to travel? ')

if destination == 'New York' or destination == 'Los Angeles' or destination == 'Seattle':
   print('Enjoy your stay in', destination)
else:
   print('Oops, that destination is not currently available!')
```

2. Imagine you're at a job interview for a Python developer role. The interviewer says "Explain logical operators in Python". Draft how you would respond.

    Logical operators in Python include things like **and**, **or** and **not**. They return a Boolean value of True or False. They are useful for checking multiple conditions at once and performing an action based on the results. Logical operators are used for controlling the flow of a program, particularly when making decisions with conditional statements like **if**, **while**, or **for** loops.

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

    Functions are sets of instructions that process or manipulate your code. There are built-in functions in Python like like print(), and len(), but you can also create your own functions. They are useful for repeating steps, allowing you to condense the code and save time. Functions also help you keep your code clean and concise.

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 really happy with my Python progress so far. The command line interface has become much more familiar and I'm getting more comfortable using it to navigate and issue commands. I also feel like I'm gaining a good understanding of Python basics and troubleshooting.

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?

   Files are used to store Python data permanently. While just working with data in the python or iPython shell is useful for learning or testing small bits of code, the data is deleted as soon as the code is finished running. Python uses text and binary files to store variables, functions and data on your machine so that it can be accessed again later.

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 convert complex data into a packaged stream of bytes, known as a "pickle," then write this into a binary file. These files can easily be read by a machine, but are not human-readable. They are useful when the data structure is more complex than a simple list of string or numbers, which work well in a plain text file. Once the complex data structure (like a recipe dictionary) has been converted to a pickle using the dump() method, it can then be converted back to the original, human-readable version via the pickle.load() method.

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 complete either task, you first need to import the os module. Once that's done you can use the command os.getcwd() to determine your current directory. To change directories, you'd use the command os.chdir(). The syntax for the second command is os.chdir("<path to desired folder>").

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?

   If you're worried that a block of code could produce an error, a try-except block can help. You'd place the code you think might cause an error in the try block. Then, in the except block(s), you can write code for what to do if an error is encountered, like a file not found or invalid characters entered by the user. This could include printing an error message for the user or including a return statement to stop the function from going any further. You could then include an else block to include the code to run if the try block does not encounter an error. Lastly, you can use a finally block to write code that should run no matter the result of the try block.

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.

   I'm feeling pretty good so far in the Python Achievement 1. I'm really proud that I was able to write nearly all of the recipe_input.py and recipe_search.py scripts before looking at student submissions and other resources for help. I was also able to troubleshoot issues in the code relatively quickly. I think I need more practice using loops. For example, given the instructions, I want more practice determining how a loop should be structured to get the desired results.

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?

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

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

Method	Description
Inheritance	
Polymorphism	
Operator Overloading	

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?

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

Data type	Definition

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

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?

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.