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 completed the front end and full stack immersion courses for careerfoundry. I have taken some coding classes in high school, where I learned some C++.

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

I know almost nothing about Python, except that many people love using the language because it's "simple" and "easy to read".

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 I will have technical difficulty, during the full-stack immersion course specifically, there were many points where the reading didn't match what I had to do exactly and it made for very frustrating experiences. I was eventually able to get through it, but I hope there are not as many differences in this python course. I am hoping my mentor is encouraging and active so I am able to come to them with the problems.

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 what the user sees, it is more about the design of a website and users can see what a frontend developer is doing to the website. Backend makes me think of people who work on the servers and it feels like the structure of the site. Backend is

also harder for a user to see because it's a lot of communicating with things unseen. If i was hired for backend programming I would be working on things like SQL and MongoDB to work on servers and databases.

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?

There are many reasons to choose both options, but my pitch for Python would start at how readable it is, which is good especially if not everybody knows JavaScript or Python. It would be easier for everybody to learn Python than JavaScript. Python also comes with many packages and it has a very simple packaging system. Like the Node Package Manager, Python has virtual environments that can be used on individual parts of an app so the app can run faster.

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?

I want to become proficient enough in Python to use it for my own projects. I want to be able to put it on a resume and work in the field with Python because I do enjoy backend, so having a well-known and simple language to start working with sounds very appealing. I hope to start helping people make their own websites soon and eventually move on to bigger things like making games with it, if possible!

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 much more readable for coders and even non-coders. The iPython shell can also let you test out snippets of code. The default shell is viewed like how you would read a book, but the iPython shell numbers each line, spaces them out, shows what you typed with **In** and what the output is with **Out**. The iPython shell also automatically indents. Basically, it's legible and simple to use, compared to the default shell which becomes a chunk of text fast with no spacing or simplicity.

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?
Float	A number with decimal points. Can be extremely small or extremely large, positive or negative	Scaler
String	A string is used for a character or characters. Can be letters, numbers, special characters, spaces, in quotes	NS
List	A list is showing a collection of items. Can be anything in the list including other lists	NS
Dictionary	A dictionary is similar to a list, except this shows key-value pairs so each item is matched with a key	NS

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 are mutable and Tuples are immutable. So lists can be changed easily, while a Tuple you can never change (you can copy the information into a new tuple, but you don't change the original). Lists are more practical, even if you aren't sure whether you will be changing anything, if there is a chance you should always use a list. A tuple is best for a collection of things that you don't want being changed by anyone.

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.]

The most obvious solution is just using a Dictionary, since key-value pairs make the most sense. For example, the key will be the word and the item will be the definition. If not a dictionary, I would do a list since the list will need to keep growing for the new flashcards being created. You could create two lists, one for words and one for definitions. Then you just go through the list of definitions and try to match it with the list of words. I would least likely use a tuple since it is immutable, but I could see some use cases for it. For instance, if it was an app created for the user where all the words were in many tuples for different levels and everything was already created for the student, nothing needs to be added.

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!)*

```
place = input('Where are you traveling?')
if place == "Berlin":
   print('Enjoy your stay in Berlin!')
elif place == "Canada":
   print('Enjoy your stay in Canada!')
elif place == "Antarctica":
   print('Enjoy your stay in Antarctica!')
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 are "and", "or" and "not", which in some other languages are "&&", "||" and "!" which I'm more familiar with and are mostly used in conditional statements. Use "and" to describe ALL things that need to pass in a conditional, like x>8 and y<2. BOTH statements have to be true, x has to be less than 10 and y has to be greater than 2. Use "or" to describe at least ONE thing that needs to pass, like a<100 or b>10. Here, if a is less than 100 then the statement passes before even checking if b is greater than 10 because only 1 of the statements needs to be true (both could be true. Use "not" to describe something that cannot be true for the statement to continue, like if a!=b. This means that if a does NOT equal b then the if statement is true and will continue.

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

Functions are EXTREMELY helpful in Python. A function is used to create a simpler way of doing a task. For example, if you have to add the prices of all the ingredients you buy, you can create a function where you enter in all the prices and the function

adds them all up. It feels like a machine that you create the directions for and then you put something in it that will follow the directions you set the function to follow, usually returning something back to you. They are great for efficiency when doing the same thing over and over again, but they are also good just for legibility. A function like “monthlySpending” could have many strings, lists, ints, and all sorts of things within it, but the name itself lets you know it will be looking for how much you spent in the month.

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 already feel comfortable with Python, in the sense that I could start a project and look up more tutorials to complete them. I don’t think I could start creating websites with it yet, but I do feel like I will be able to soon. It’s a very intuitive language and I will be able to put it onto my resume soon I’m sure. Also, I’ve been enjoying it so much I have been looking at more backend jobs because of it. Very excited for the logical thinking that comes with backend!

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?

You need files to be saved in storage for you to read and write with. You need to get files, you need to add to files, you need to create new ones, all of this is necessary to do anything online. Python storage is important also for testing, since when you create an app and test it, if you don’t store it all that data is lost so when trying to work on data produced after you close and reopen the app, it’s not possible.

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?

Pickling is used when you need to use data over multiple sessions. For instance, I created a recipes list in an exercise I've already completed, but when I want to reference the recipes I made in that last I cannot because I am not still in that session. Pickling lets you turn files into binary and lets you read binary so you are able to save/store data and read/write in it as well.

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?

You'll have to import os, but after doing that you can use os.getcwd() to see where you are and os.chdir() to change the current working 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?

You can use try-except blocks to prevent the entire script from terminating. So you put the code that could cause the error into the try block and if no error occurs, it will skip the except block/s. However, if an error does occur it will lead into the except block linked to that error or linked to all errors. It will then complete the except block, which is usually just printing what the error is, and then continue with the rest of the code.

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 definitely need more help with "for loop" practice and I need to start getting more comfortable using repetitive lines of code. For instance, the except block with except Exception as e: and other things that are common, but not common to me yet. I am proud that I even made it to python and I will be excited when I feel myself getting more used to this coding using logical thinking, compared to the front-end creative thinking.

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 creates a much more organized way of coding. You create a class holding all of the variables and methods you create, then you can call on them to do whatever tasks you need to do, all within that class. You can have subclasses inherit things from the parent class, but besides that most classes are separate so it feels like you're typing an entire file of code into these smaller classes. Compare this to how I created the recipes in ex 1.4, all the code was put in a specific order on 1 file and now it's all within a class that you can just call on and reuse constantly.

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

Classes are what you use to create objects. You create the class to hold on the information and create the methods and/or variables that the object will be able to use. Then you create an object that can use those limits that the methods and variables describe. An example would be like a Book class where you can create an object called 1984. Book could have methods like get_author() and get_title(). It will also have an __init__ section where you can have all the information about the book when you enter it in.

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

Method	Description
Inheritance	A subclass can inherit things from the parent class. This helps decrease redundancy in code. So an animal class can give its subclass the name and age, but the animal itself can have its own method to speak.
Polymorphism	Going along with the animals speaking, you can have the same method mean different things for different classes, this is polymorphism. So 2 different subclasses could both have speak, but lets say it's 2 different people who speak different languages then Matt.speak() could say "hello" while Sam.speak() could say "hola"
Operator Overloading	Operator overloading is when you want to use an operator with two different objects so you make it a method inside the class. For instance, you can't just add two different objects even if they are only made up of numbers, you would have to create an __add__() method so you could add height1 + height2.

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.