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

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.

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 web development involves the parts of an app that users can see or interact with. Backend development involves servers and any data that needs to be passed from the user to the servers or from the servers to the user. A job in backend programming would include working with APIs and databases, managing endpoints to share with frontend developers, and focusing on how user data is stored and used.

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 few similarities between JavaScript and Python are that they both are scripting languages with easily understandable keywords to implement commands and perform tasks. They also both use dynamic typing which allows for flexibility in that variables do not have to stick to a certain type of value. Finally, both also use packages and libraries to increase efficiency for developers.

   One major difference is readability. Python was created with ease of readability as one of its key features, allowing for easy and quick understanding of code. This also helps with avoiding errors

and easy debugging. Python also includes an interactive shell. Python may be a better option for a team of developers because of the readability and ability to address errors more quickly. The simplicity of package management, particularly around version management, and the pre-existing essential operations, Python would allow a team to get started and prototype more quickly than JavaScript.

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?

   First, I want to focus on some of the details of how to talk about this new language—I'd just been getting more comfortable with JavaScript and React which I think may be helpful in wrapping my mind around new terminology.

   Second, I want to seek out solutions consistently and proactively when I'm confused or stuck before I ask for help.

   Third, I want to stay on top of updating the readme for this project specifically for the purpose of being confident in talking about what I'm doing in different phases of the project so that I'd be comfortable talking about it in interviews.

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?
   - It's easier to read code in the iPython shell
     - Syntax highlighting helps to more easily identify different parts of code
     - Indentation is automatic
     - The auto-complete view is easier to browse quickly
   - Able to run code more quickly–commands are executed immediately

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	Decimal numbers	scalar
bool	Stores true or false	scalar
string	Array of characters (letters, numbers, symbols) surrounded by single or double quotes	non-scalar
tuple	Linear array of values (different from lists because not mutable)	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.

   They look similar, but what helps me to tell them apart is thinking about the different operations that can be performed on each. Unlike with tuples, with lists, you can modify each of the elements. Tuples allow you to access different segments through slicing and to add elements to the list. Lists also allow that, but they also allow for changing the values of different elements of the list, sorting the list, and deleting elements from the list. A strength of lists is that they are overall more flexible, but tuples make it easier to view larger amounts of data more easily.

4. In the task for this Exercise, you decided what you thought was the most suitable data structure for storing all the information for a recipe. Now, imagine you're creating a language-learning app that helps users memorize vocabulary through flashcards. Users can input vocabulary words, definitions, and their category (noun, verb, etc.) into the flashcards. They can then quiz themselves by flipping through the flashcards. Think about the necessary data types and what would be the most suitable data structure for this language-learning app. Between tuples, lists, and dictionaries, which would you choose? Think about their respective advantages and limitations, and where flexibility might be useful if you were to continue developing the language-learning app beyond vocabulary memorization.

   For flashcards, I think the best data structure for each flashcard is a dictionary for the same reasons that it was a good fit for recipes: dictionaries allow for key-value pairs and for the keys and values to take any data structure. User's individual flashcards could be kept in lists to allow for accessing all flashcards. These things allow flexibility for users to add flashcards as needed, remove ones, or modify flashcards as their knowledge increases.

   I think a potential format to consider if the app continues to develop is to create standardized dictionaries stored in tuples, maybe by level, to be able to quickly access a

list of vocab words. Words could be added to these lists but not taken away to make sure there is some form of standardization. Then users could add elements from the tuple to their own lists to make notes or personalize info on the flashcards.

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 to?")

if destination == "Hawaii":
  print("Enjoy your stay in Hawaii!")
elif destination == "Zanzibar":
  print("Enjoy your stay in Zanzibar!")
elif destination == "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 basically evaluate what conditions are in place in order to take an action. For example, if there are two statements, that and operator only returns true if both statements are

true. The or operator returns true if only one or both of the statements are true, and the not operator returns the opposite of what the output would be without it there.

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

   Functions are a set of steps taken to do something to the code in Python. They are useful because they can be written one time and then used in multiple places. For example, if multiple strings needed to be turned into a list and then sorted, instead of running those multiple steps multiple times, the function can be called on that string in just one line.

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 had forgotten what my goals were, but I'm actually really happy to see that I feel like I'm making progress on them! I'm feeling comfortable and confident with the new skills and debugging issues, and when I'm stuck, I feel really well equipped to figure out the answers (either from referring back to the CareerFoundry materials or googling). Lastly, I'm working on consistent readme updates.

Exercise 1.4: File Handling in Python

Learning Goals

- Use files to store and retrieve data in Python

Reflection Questions

1. Why is file storage important when you're using Python? What would happen if you didn't store local files?

   File storage allows the data created in scripts to be stored and accessed later on (rather than all the data disappearing when the script stops running).

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 store data structures that are more complex. In this task, pickles were used to store and access dictionaries. They convert dictionaries or other complex structures into bytes and store them in binary files. They can read and write to binary files

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

To find the current directory or change to new directories, you should use the os module. First the os module has to be imported. Current directory: print(os.getcwd()). Move to a new director: os.chdir(<path to new 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?

   This is where try and except blocks are useful. The try block is where the desired code would go, except blocks can specify the exact type of error that may terminate the script and includes actions for the script to take besides terminating.

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 really enjoying Python! I feel like I'm grasping it really quickly (it's helped to do all the technical interview practice). The one thing that feels a little bit challenging is that I'm finding myself wanting quicker feedback on whether or not my scripts have errors (for example, I had some type errors around printing the recipes or in how I was using the pickle modules, and I had to just keep initiating python in my terminal, and I found myself wanting quicker feedback).

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?
   OOP uses classes to group data and methods that can form a sort of template for individual objects. Benefits are that it reduces duplication of data and functions and allows for multiple objects that have consistent types of attributes and methods.

2. What are objects and classes in Python? Come up with a real-world example to illustrate how objects and classes work.
   Classes create a frame for objects with different data attributes and methods. Objects are specific instances of those objects. For example, for a business review site like Yelp, restaurants may be a Class that include name, location, general cost, and reviews, but one specific restaurant would be an object in that class with its own name, location, cost, and reviews.

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

Method	Description
Inheritance	Passing attributes or methods from one class to another (using example above: a Restaurant class may pass on attributes and methods to a subclass like FastFood which would then have objects like McDonalds or Burger King)
Polymorphism	Methods that are named the same way will not always return the same thing depending on which object is calling it. (speak example from the exercise: dog.speak() may return woof while cat.speak() may return meow)
Operator Overloading	This is defining operators like +, -, >, <, etc. in special methods in a class in order to use them outside of a class. They are defined with double underscores and a specific name (add, sub, gt, lt, etc.)

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.