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 recently completed CareerFoundry's Full Stack Immersion course. I have learned HTML, CSS, and JavaScript over the last 20 weeks. I also have familiarity with SQL and NoSQL as well as database structure fundamentals.

2. What do you know about Python already? What do you want to know?
   I knew very little about Python prior to this Specialization course other than being aware that it was a programming language affiliated with machine learning and had been around since the 90s. At such an early point in my exposure to its concepts, I would like to explore more about object-orientated programming as well as the implementation of package libraries within Python's functionality.

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.
   With this course acting as my introduction to Python as a programming language, I expect each task to be a challenge in itself. I believe that this grants me the opportunity to approach each challenge with a broader mindset in an effort to try new problem-solving strategies.

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?

   The difference comes down to structure and purpose. The front end is built with the intention of creating a visual and interactive experience for the user, while the back end provides the functionality and logic behind the application. For instance, the backend would feature database storage and configuration as well as security measures to protect proprietary information.

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")

   In this scenario, I would provide data supporting Python as the most user-friendly approach due to its readability and flexibility related to its Dynamic Typing features. It comes down to ease of doing business for all parties involved. With features like Built-in-Packaging management, Python allows its developers to utilize automated functions, ultimately allowing for speedier and more precise web development.

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 understand the structure of virtual environments more
   > I want to explore more built-in-packaging options and understand their practical uses
   > I want to understand how to work on database management with 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?

   Aside from offering extensions and customization, iPython is more user-friendly, allows for rich output to allow for complex visual tasks, and includes comprehensible history management.

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?
Dictionaries	Unordered set of items where each of them a key-value pair, where each key is unique	Non-Scalar
Lists	Mutable character sequence wrapped in [ ]	Non-Scalar
Strings	Immutable sequence of characters wrapped in single or double quotes	Non-Scalar
Tuples	Linear arrays that can store multipole values of any type	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.

   The difference comes down to being able to change their content. Lists offer the ability to change their contents within the square brackets. Tuples on the other hand are immutable and do not allow any modification to the content stored inside parentheses.

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.

   Based off what I learned off of building the recipe structure, I would utilize Dictionaries as they allow for their content to be modified. This would be useful to customize more accurate definitions or tracking progress on the flash card data.

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?")
If destination == "Aruba":
Print("Enjoy your stay in Aruba!")
Elif destionation == "Hawaii":
Print("Enjoy your stay in Hawaii!")
Elif destination == "Bermuda":
Print("Enjoy your stay in Bermuda!")
Else
Print("Oops, that destination is currently unavailable")
```

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.

    Python consists of three main logical operators, 'and', 'or', and 'not'. These operators are used in conditional statements such as 'if' and 'elif' to control the flow of a program. This could involve combining or negating conditions based on the 'and' 'or' statements.

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

    Functions are a fundamental building block that allow a programmer to organize code into modular components that can be reused throughout the code. Functions specifically in Python are essential at building complex and scalable applications.

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 want to understand the structure of virtual environments more *(feeling more confident on this but still need some practice)*
    > I want to explore more built-in-packaging options and understand their practical uses *(need practice)*
    > I want to understand how to work on database management with Python *(need practice)*

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 is a fundamental part of any software application. Python is capable of working with a versatile group of files and libraries which ultimately allows for a wide range of data storage and interoperability with data sharing. The inability to store local files would result in data loss, a breakdown in collaboration, as well as major security concerns regarding sensitive data.

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 a way to convert objects such as lists or dictionaries into a format that can be save to a file for future use. Pickles are useful to preserve complex data structures as well as saving the state of an object for object serialization purposes.

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?

   The function to find the absolute path to the current directory is:

   Import os
   Current_directory = os.getcwd()
   Print("Current Directory:", current_directory)

   The function to change directories is:

   New_directory = '/…..(path to new directory)
   Os.chdir(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?

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.

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?

   Plain Python makes sense for a project with unique requirements or the necessity for full control. Plain Python comes with a lot of flexibility and simplicity with its code writing, however the disadvantages come in the form of security risks. Without a framework like

Django to handle input validation and user authentication, the developer takes a risk by having to implement SQL and cross-site scripting.
Django works for a project that requires a quick, secure and structured development. The benefits of Django come in the form of built-in security features, ORM features, and an abundance of extensions to work with so the developer doesn't need to reinvent the wheel. Django doesn't offer as much flexibility as plain Python and may not make sense for smaller application development.

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

   With MVT (model, view, template), the template provides a cleaner and more direct approach to handling user interface. This is achieved through the separation of concerns associated with improved readability, code reusability, and design flexibility.

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?
     How to implement its template structure

   - What do you want to get out of this Achievement?
     Improving my confidence with server-side web development

   - Where or what do you see yourself working on after you complete this Achievement?
     Codepen exercises, Web seminars, Coding practice groups through Meetup

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.*)

   Teksystems is a company known for hiring junior web developers. In Django terminology, their website utilizes multiple views, contains multiple URLs with mapped configurations, and different apps to manage separate destinations.

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

   - Install Python
   - Create Virtual Environment
   - Install Django within VE
   - Create Django project (Django-admin startprojectname)
   - Run Migrations within manage.py (python manage.py migrate)
   - Create Django App (python manage.py startapp appname)
   - Create superuser (python manage.py createsuperuser)
   - Run Development Server (python manage.py runserver)

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

   The Django admin site offers user and data management tools during web development. I would utilize Django's admin site to update any models I was working on by having clear visibility to the models.py file of my app. This is a very user-friendly option for managing database records.

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.