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?

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")

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?

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?

   Answer: iPython Shell is like a supercharged version of the regular Python shell, offering interactive coding, cool visuals, quick docs, and smooth testing, perfect for scientific stuff and data fun.

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, object like data structure with key-value pairs where each key is unique	Non-Scalar
Tuples	Linear arrays that cannot be modified. Can store multiple values of any type	Non-Scalar
Lists	Mutable character sequence wrapped in [ ]	Non-Scalar
Boolean	Represents 2 possible values: true or false	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.

Answer: Lists and tuples in Python both store collections of items, but lists are mutable (modifiable), enclosed in square brackets [ ], while tuples are immutable (unchangeable), enclosed in parentheses (). Use lists when you need flexibility, and tuples when you want unchanging data.

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.

Answer: I think dictionaries would be suitable since they're flexible. They allow for anything to be stored as a value and are mutable so the app could be updated beyond vocabulary memorization in the future

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 == "Toronto":
  Print("Have a wonderful time in Toronto!")
Elif destination == "Berlin"
  Print("Have a wonderful time in Berlin!")
Elif destination == "Berlin"
  Print("Have a wonderful time in New York!")
Else
  Print("Travel location not found!")
```

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.

Answer: Logical operators in Python helps combine conditions. 'and' checks if both are true, 'or' checks if at least one is true, and 'not' flips the result. They're like tools to build smarter decision-making in our code."

3. What are functions in Python? When and why are they useful?
   Answer: Functions in Python are blocks of reusable code that perform specific tasks. They help break down complex tasks, promote code reusability, enhance readability, and simplify testing and debugging

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.
   Answer: So far I have made some progress in learning the language python, however one of my goals for example was to learn OOP which I haven't made any progress yet, I believe I'll learn it further down the course. Overall I am excited and eager to learn more to achieve my goals.

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?
   Answer: File storage is important in Python for storing and managing data persistently. Without local file storage, you would lose the ability to save and retrieve data between sessions, leading to data loss and lack of continuity in your programs.

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?
   Answer: Pickles are like magic jars for saving stuff. You can turn complex things, even special objects, into a kind of code that you can keep or send easily. Imagine using pickles to remember and bring back Python stuff - it's like storing game levels, saving where you left off, or passing secrets between friends.

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?
   Answer: I would use the 'os' module to work with directories and paths.os.getcwd() for example would tell me what the current directory is.

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?
   Answer: I would use the try and except block to handle potential errors. This way, even if there's an error in a specific block of code, the entire script won't stop abruptly, and I can provide alternative actions or error messages to ensure smoother execution.

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.
   Answer: I would say so far I am making rapid progress. Though, it is getting complicated going down each exercise, I believe I would have to keep re-reading and re-learning the things I learn in order to fully grasp everything.

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?
   Answer: Object-oriented programming (OOP) is a way of writing code that's like organizing a toolbox. Instead of thinking about code as a list of instructions, you create "objects" that bundle data and actions together. The benefits of OOP include better organization, reusability of code, easier maintenance, and modeling real-world concepts more intuitively.

2. What are objects and classes in Python? Come up with a real-world example to illustrate how objects and classes work.
   Answer: In Python, objects are instances of things, and classes are blueprints to create those things. Like how a "Car" class makes "my_car" and "friend_car" objects, each with unique data and actions, similar to real cars sharing a design but having individual features and functions.

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

Method	Description
Inheritance	Inheritance in OOP is like passing down traits from parents to children. Imagine a "Vehicle" class as a parent and "Car" and "Bike" classes as its children. The children inherit characteristics like "wheels" and "engine"

	from the parent. It's efficient as you can make changes in the parent class, and all its children get updated automatically. For instance, if you add a new method "turn_on_lights" to the "Vehicle" class, both "Car" and "Bike" objects can use it without rewriting the code. It encourages reusability and a structured code hierarchy.
Polymorphism	Polymorphism in OOP is like different things sharing the same interface. It's about treating objects of different classes in a unified way if they have common methods. Imagine a "Shape" class with a method "area." Both "Circle" and "Rectangle" classes inherit from "Shape" and provide their "area" implementations. Now, you can call "area" on any shape object, and Python figures out which version to use. This makes code more flexible and versatile. You can extend functionality by adding new classes that adhere to the same interface, making it easy to accommodate future changes.
Operator Overloading	Operator overloading is about giving your objects the ability to use familiar operators in custom ways. For instance, in a "Vector" class, you can define how + or - should work between instances. It's like extending the capabilities of objects to match human-like expectations, making code more concise and intuitive.

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?
   Answer:
   - Vanilla (Plain) Python: Offers control but requires manual web development tasks, suited for smaller projects.
   - Django Framework: Provides rapid development, built-in features, and scalability but may have a learning curve and can be overkill for simple projects.

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

Answer: The most significant advantage of the Model View Template (MVT) architecture over Model View Controller (MVC) is its inherent separation of the template layer, making it more focused on the presentation and user interface, which simplifies front-end development and promotes cleaner, maintainable code.

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?
      I am interested in the similarities between javascript frameworks and python frameworks.
    - What do you want to get out of this Achievement?
      I want to be proficient in django
    - Where or what do you see yourself working on after you complete this Achievement?
      I can see myself working on other projects, as my interest lies more in front end actually, however, I believe it'll still be super beneficial to have backend skills, languages and frameworks under my belt for my future career

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.*)
   A: (Discord) I would identify multiple sections, to name a few: Homepage, user profiles, servers and channels and messages.

2. In your own words, describe the steps you would take to deploy a basic Django application locally on your system.
   A: To deploy a basic Django application locally, I'd install Django, create a project, run initial migrations, start the development server, and finally create a superuser for admin access.

3. Do some research about the Django admin site and write down how you'd use it during your web application development.
   A: The Django admin site is like having a personal assistant for managing your web application's data. It offers a user-friendly interface to create, edit, and organize your information without needing to write custom code. Plus, you can fine-tune it to fit your project's specific needs, which saves you a lot of development time and effort.

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.
   Answer: Django models are Python classes that define the structure and behavior of a web application's data. They work by abstracting the database, providing an object-oriented way to interact with data. Benefits include data consistency, code reusability, easy relationships between data, an admin interface, and enhanced security, making it easier to develop and maintain web applications.

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.
   Answer: Writing test cases from the beginning of a project is crucial because it helps catch and fix issues early, ensures that new features don't break existing functionality, and provides documentation for how the code should work. For example, in a recipe app project, writing tests for creating and editing recipes ensures that these features work as intended and continues to work as the project evolves, preventing bugs and saving time and effort in the long run.

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.
   A: Django views are Python functions or classes that process web requests and generate web responses. They define what data to show and how to display it in web pages. Views are crucial for building interactive web applications in Django.

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?
   A: I would choose class-based views for better code organization and reusability.

3. Read Django’s documentation on the Django template language and make some notes on its basics.
   A: Django's template language is used for dynamic content in web apps. It has variables, tags, and filters for logic and display. You can use control flow tags like {% if %} and create reusable templates with inheritance. It also supports custom tags/filters and auto-escaping for security.

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.
   A: Django static files are assets like CSS, JavaScript, and images. Django stores them in a "static" directory within apps, collects them centrally, and serves them in production. You use {% static %} tags in templates to reference these files.

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	ListView is a class-based view used for displaying a list of objects from a database. It's commonly used for implementing features like paginated lists, blog post archives, or any situation where you want to present a collection of items to users.
DetailView	DetailView is another class-based view in Django that is used for displaying detailed information about a single object from a database. It's often used for building pages that show the full details of a specific item, such as a blog post or a product.

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.
   A: So far I have learned a lot about python and django, however, it is still complicated and hard to wrap my head around the MVT architecture and how exactly it works. I guess if I learn and practice more I'll learn it eventually.

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.
   A: Authentication is vital for verifying user identities, protecting sensitive data, and ensuring users have the right permissions. Without it, applications risk security breaches, privacy violations, and unauthorized access. Incorporating authentication builds trust and enhances user experiences.

2. In your own words, explain the steps you should take to create a login for your Django web application.
   A:
   - Install Django.
   - Define a User Model.
   - Create URLs, Views, and Templates for login and registration.
   - Configure authentication in settings.
   - Secure passwords.
   - Test thoroughly.
   - Deploy and monitor for security.

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()	The authenticate function is used to check a set of credentials (usually a username and password) and verify whether they are valid for a user in the Django authentication system. If the credentials are valid, it returns a user object; otherwise, it returns None.
redirect()	The redirect function is used to create an HTTP redirect response. It takes a URL as an argument and generates an HTTP response that redirects the user's browser to the specified URL. This function is often used when you want to direct users to a different page or view.
include()	The include function is part of Django's URL routing system. It's used to include other URL patterns from separate Python modules into the project's main urls.py file. This allows you to organize your URL patterns into smaller, reusable components.

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.
   A:
   - Personalization: It offers personalized product recommendations for a better user experience.
   - Inventory Management: Efficient stock management based on purchase history.
   - Targeted Marketing: Identifies target audiences for effective promotions.
   - User Experience: Enhances website usability based on behavior analysis.
   - Security: Detects and prevents fraudulent activities.
   - Content Creation: Generates content aligned with user interests.
   - Supply Chain: Predicts demand and manages stock efficiently. Customer Support: Improves support by addressing common issues.

2. Read the Django official documentation on QuerySet API. Note down the different ways in which you can evaluate a QuerySet.
   A:

- Iteration
- Slicing
- Conversion to Lists
- Boolean Context (e.g., in an if statement).
- get() method (for retrieving a single object).
- count() method (to count objects).

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.
   A: DataFrames are often better for data processing because they are designed for tabular data, offer rich functionality, support indexing and labeling, handle data cleaning tasks, provide flexibility in data types, and optimize operations for better performance. However, they can be memory-intensive for large datasets.

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.
   A: You can use CSS for styling and JavaScript for interactivity in your Django web application by including them in your HTML templates. CSS can be linked via <link> tags in the template's <head>, and JavaScript can be added using <script> tags, typically at the bottom of the <body> or via external JavaScript files included with the {% static %} template tag for efficient asset management.

2. In your own words, explain the steps you'd need to take to deploy your Django web application.
   A:
   - Choose a hosting provider.
   - Prepare your app for production.
   - Set up a server/platform.
   - Configure a production database.
   - Collect static files.
   - Secure the app with HTTPS.
   - Deploy your code and configure a web server.
   - Apply database migrations.
   - Monitor, test, and scale as needed.

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.