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 studied Mechatronics in the past, for this reason I am not so new to programming.

2. What do you know about Python already? What do you want to know? I know Python is a very flexible programming language, and I would like to learn as much as I can to grow together with AI and Machine Learning.
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.
   Probably some kind of lack of information, since learning  a language is a long journey.

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 is what the user sees, Backend is what is behind the scenes. I would be working on databases, server requests architecture, security like authentication & authorization, APIs, etc.

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? Because of its simplicity to deal with data. *(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?

**My Goals for Learning Python:**

1. **Master Basic Python Concepts:**
    - **What I want to learn:** I want to thoroughly understand Python syntax, basic data structures, and control flow mechanisms like loops and conditionals.
    - **Outcome I want:** Gain confidence in writing basic Python scripts and solving simple programming problems.
    - **Future vision:** I see myself working on small projects and solving coding challenges to reinforce my understanding of Python fundamentals.
2. **Develop Problem-Solving Skills:**
    - **What I want to learn:** I want to improve my ability to break down complex problems into smaller, manageable parts using Python.
    - **Outcome I want:** Be able to tackle more challenging tasks and write efficient, clean code.
    - **Future vision:** I aim to contribute to open-source projects or collaborate on coding projects with others to hone my problem-solving skills.
3. **Build Real-World Applications:**
    - **What I want to learn:** I want to learn how to apply Python to real-world scenarios, such as web development, data analysis, or automation.
    - **Outcome I want:** Create projects or applications that demonstrate my skills.
    - **Future vision:** After completing this Achievement, I hope to work on more significant projects, potentially landing an entry-level position as a Python developer.

# 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 is convenient to use for easy and quick test of small chunks of code, and better visulization.

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? |
|---|---|---|
| Int | Represent whole numbers | scalar |
| float | Represent numbers with decimals | scalar |
| str | Represent characters, like words | non-scalar |
| bool | Represent true and false values | scalar |

3. A frequent question at job interviews for Python developers is: what is the difference between lists and tuples in Python? Write down how you would respond.
   Lists in Python are mutable, meaning they can be changed, while tuples are immutable and cannot be altered once created.
4. In the task for this Exercise, you decided what you thought was the most suitable data structure for storing all the information for a recipe. Now, imagine you're creating a language-learning app that helps users memorize vocabulary through flashcards. Users can input vocabulary words, definitions, and their category (noun, verb, etc.) into the flashcards. They can then quiz themselves by flipping through the flashcards. Think about the necessary data types and what would be the most suitable data structure for this language-learning app. Between tuples, lists, and dictionaries, which would you choose? Think about their respective advantages and limitations, and where flexibility might be useful if you were to continue developing the language-learning app beyond vocabulary memorization.
   I would choose dictionaries because they let you store each vocabulary word with its definition and category, making it easy to look up and organize the information.

# 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 == "Paris":
    print("Enjoy your stay in Paris!")
elif destination == "New York":
    print("Enjoy your stay in New York!")
elif destination == "Tokyo":
    print("Enjoy your stay in Tokyo!")
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, like and, or, and not, are used to combine or invert conditions in if statements. They help control the flow of the program based on multiple conditions.

3. What are functions in Python? When and why are they useful? Functions in Python are blocks of reusable code that perform a specific task. They help organize code, make it more readable, and reduce redundancy. You use them when you need to perform the same operation multiple times or break down a complex problem into smaller, manageable parts.

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.

   **Progress Towards Goals:**

   - **Goal 1:** Understand basic Python syntax and structure.
     - **Progress:** I've completed exercises on conditionals and loops, gaining a solid understanding of how to control program flow.
   - **Goal 2:** Learn to write clean, reusable code.
     - **Progress:** Practiced writing functions to organize code, making it more modular and easier to debug.
   - **Goal 3:** Build small projects to apply what I've learned.
     - **Progress:** Started creating small scripts, like a simple travel app, to apply the concepts of conditionals and functions in real scenarios.

# 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 important because it allows you to save data for later use, making it possible to access and manipulate that data even after your program ends. Without storing local files, you'd lose all the data each time the program closes, making it hard to maintain any kind of persistent information.

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 are a way to serialize and save Python objects to a file so you can load them back later. You'd use pickles when you need to save complex data structures, like dictionaries or custom objects, and easily restore them without having to manually parse and reconstruct the data.

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 out the current directory, use `os.getcwd()`. To change the current working directory, use `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? Use a `try-except` block to catch and handle errors gracefully. This way, if an error occurs in the block of code, the script can continue running or provide a useful error message instead of just stopping.

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.
   **How is it going?** It's going well; I'm learning a lot about Python basics and file handling.
   **Something you're proud of so far?** I'm proud of understanding and applying concepts like loops, conditionals, and file handling.
   **Something you're struggling with?** I'm finding it a bit challenging to remember all the syntax and functions.
   **What do you need more practice with?** I need more practice with writing and organizing functions and using file handling effectively.

# 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 (OOP) is a way of designing software by organizing code into "objects." Objects are like little building blocks that combine data (like variables) and functions (like methods) into one unit. These objects are created from "classes," which are like blueprints or templates.
   **Benefits of OOP**:

   - **Organization**: It helps keep code organized and manageable by grouping related data and functions together.
   - **Reusability**: You can reuse code from one class in other parts of the program or in new programs.
   - **Modularity**: Each object is a self-contained unit, which makes it easier to fix problems and update code.

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

   In Python, a **class** is a blueprint for creating objects. It defines a type of object and its behavior. An **object** is an instance of a class. It is created using the class blueprint and contains data and functions defined in the class.

   **Real-World Example**:

   - **Class**: Think of a `Car` class as a blueprint for creating cars. It might have attributes like `color`, `model`, and `speed`, and methods like `accelerate()` and `brake()`.
   - **Object**: An individual car, like a red Toyota Corolla, is an object created from the `Car` class. It has specific attributes and can perform actions defined by the `Car` class.

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 is a way to create a new class that is based on an existing class. The new class, called a "child" or "subclass," inherits attributes and methods from the existing class, known as the "parent" or "superclass." This helps avoid code duplication because the child class can use or override the parent class's features. For example, if we have a `Dessert` class that inherits from a `Recipe` class, the `Dessert` class would get all the attributes and methods of `Recipe` but can also add its own unique features. |
| Polymorphism | Polymorphism allows different classes to be treated as instances of the same class through a common interface, typically using methods. It means that the same method name can be used in different classes but can work differently depending on which class is calling it. For example, a `print_recipe()` method might show details differently for a `DinnerRecipe` class than for a `DessertRecipe` class. Polymorphism makes code more flexible and easier to extend. |
| Operator Overloading | Operator overloading lets you define how operators (like +, -, *, etc.) work with objects of a class. By overloading operators, you can make them perform custom operations for your objects. For instance, if you have a `Recipe` class and you want to add two recipes together to combine their ingredients, you can overload the + operator to perform this task. This makes your classes more intuitive and integrates them better with Python's syntax. |

Just for “me” to remember well:

**Method**: A method is a function that is defined inside a class and is meant to operate on the data within that class. Methods can access and modify the object's attributes and can perform actions related to the object. For example, in a `Recipe` class, a method like `add_ingredients()` might add new ingredients to a recipe's list. Methods help objects perform tasks and interact with their data.

# 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?

   Databases are organized collections of data that are stored and managed electronically. They help keep data structured and easily accessible. Here are some advantages of using databases:

   - **Organization:** Databases organize data in tables, which makes it easier to find and manage.
   - **Efficiency:** They allow for fast retrieval and manipulation of data using queries.
   - **Consistency:** Databases ensure that data is accurate and up-to-date through features like constraints and transactions.

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

| Data type | Definition |
|---|---|
| INT | Stores integer numbers (e.g., 1, 2, 3). Useful for counting or IDs. |
| VARCHAR | Stores variable-length text strings (e.g., names or descriptions). You specify the maximum length. |
| DATE | Stores dates (e.g., 2024-08-05). Useful for keeping track of events or deadlines. |

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

   SQLite might be a better choice than MySQL in the following situations:

   - **Single-user Applications:** When only one user is interacting with the database, such as in a desktop application.
   - **Small to Medium-Sized Data:** When the application doesn't need to handle large volumes of data or high traffic.
   - **Simplicity:** When you need a simple, self-contained database with minimal setup, such as for a small project or a prototype.

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?

   JavaScript and Python are both powerful languages but serve different purposes:

   - **JavaScript:** Primarily used for making web pages interactive. It runs in the browser and can dynamically change content on websites.
   - **Python:** Known for its simplicity and readability, it's used for a variety of tasks like web development, data analysis, automation, and more. Python has straightforward syntax, which makes it easier for beginners to learn.

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?

   Some limitations of Python may include:

   - **Performance:** Python is slower compared to languages like C or Java because it is interpreted rather than compiled.
   - **Memory Usage:** Python can consume more memory, which might be an issue for memory-constrained environments.
   - **Mobile Development:** Python is not as commonly used for mobile app development as languages like Swift (for iOS) or Kotlin (for Android).

# 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.