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 have about one year of coding experience through CareerFoundry. Some of the languages and frameworks I've learned so far include HTML, CSS, SCSS, React, and Angular, among others. These experiences have helped me build a strong foundation in web development and problem-solving, which I believe will be valuable as I continue learning Python.

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

I know that Python is one of the most widely used programming languages and has been around for a long time. I'd like to dive deeper into it and understand how I can apply it to real-world projects in the future — especially for web development, data analysis, or automation.

---

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 or week that might be favorable to your facing challenges and growing. Plan for how to solve challenges that arise.**

I might struggle a bit with learning Python's syntax and debugging at first. To overcome these challenges, I plan to use the student forum, reach out to my tutor, and take advantage of learning resources available online. I'll also make sure to set aside dedicated study time in a quiet space each week. Most importantly, I'll remind myself to ask for help early whenever I feel stuck.

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 users see and interact with — like buttons, layouts, and visuals — using tools like HTML, CSS, and JavaScript.
   Backend is what happens behind the scenes — managing data, servers, and user logins.
   If I worked on the backend, I'd be handling things like building APIs, connecting to databases, and making sure data flows smoothly to the frontend.

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")***

   Both Python and JavaScript are great, but Python feels simpler and easier to read.
   JavaScript is great for interactive frontends, while Python is better for backend work and data-heavy projects.
   I'd recommend Python because it's clean, beginner-friendly, and has strong frameworks like Django and Flask that make building web apps faster and more secure.

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?**
   - Get more comfortable writing clean Python code.
   - Build small projects to practice backend development.
   - Feel confident using Python for real-world web apps and add it to my skill set.

## 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?**

   I'd say iPython is just easier and more fun to use. It has color highlighting, auto-complete, and better error messages. You can also run system commands and test small bits of code quickly—it's like a more powerful and user-friendly version of the regular Python shell.

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? |
|---|---|---|
| Integer (int) | whole numbers like 10 or -3 | scalar |
| Floating-point number (float) | numbers with decimals like 3.14 | scalar |
| String (str) | sequences of characters like "hello" | scalar |
| List (list) | an ordered collection of items like [1, 2, 3] | 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.**

   I'd say the main difference is that lists are changeable, and tuples aren't. You can add, remove, or update items in a list, but once a tuple is created, it can't be changed. Lists are more flexible, while tuples are faster and good for storing data that shouldn't be modified.

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'd use a dictionary because it lets me store each word as a key and its details—like definition and category—as values. It's organized, easy to update, and perfect for looking things up quickly. If I expand the app later, dictionaries make it simple to add features like tracking progress or grouping words by topic.

# 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.lower() == "paris":
   print("Enjoy your stay in Paris!")
elif destination.lower() == "tokyo":
   print("Enjoy your stay in Tokyo!")
elif destination.lower() == "new york":
   print("Enjoy your stay in New York!")
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 help combine or compare conditions. There are three: and, or, and not. and means both conditions must be true, or means at least one must be true, and not reverses the result. They're super useful when checking multiple things in an if statement.

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

   Functions in Python are reusable blocks of code that perform a specific task. They're useful because they help make code cleaner, organized, and easier to debug. Instead of repeating the same code, you can define a function once and call it whenever needed — saving time and reducing errors.

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'm getting more comfortable with Python and the IPython shell, and I'm starting to see how useful they can be in real-world applications. However, I still have some work to do to become truly proficient in Python.

# 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 in Python because it lets you save data, settings, and progress so you can use them later. Without storing local files, every time you run your program, you'd lose everything — no saved data, no history, and you'd have to start from scratch each time.

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 in Python are a way to save objects like lists or dictionaries so you can load them later without rebuilding them. I'd use pickles when I need to store data between program runs, like saving user preferences or model data, because it's quick and keeps the original structure intact.

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

   You can use `os.getcwd()` to find out which directory you're currently in. If you want to change your working directory, you can use `os.chdir("path/to/directory")`. These functions are part of the built-in `os` module, which helps you interact with your computer's file system.

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?**

   I'd use a `try-except` block to handle the error safely. The code that might cause an error goes inside `try`, and if something goes wrong, the `except` block runs instead of crashing the program. This way, the script keeps running even when an error happens.

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.**

   So far, the course is going really well! I'm becoming more confident with Python and how the different concepts connect. I'm proud that I'm understanding Python faster than in previous achievements—it's really boosted my confidence as a coder and reassures me that I'll be ready for the job once I graduate. I still struggle a bit with debugging and remembering syntax, but with more practice and guidance, I know I'll get there.

# 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 writing code by organizing it into objects that represent real things, each with their own data and actions. It makes programs easier to understand, reuse, and maintain. With OOP, you can build flexible, organized code using features like classes, inheritance, and encapsulation.

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 that defines the properties and behaviors of something, while an object is a specific instance of that class. For example, if “Dog” is a class with attributes like name and breed, then “my_dog” could be an object representing a real dog named Max.

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 lets one class take on the properties and methods of another. It’s like passing traits from a parent to a child. For example, a `Car` class can inherit from a `Vehicle` class and reuse its features like `start_engine()`. Thi<br>s helps keep code cleaner and easier to manage. |
| Polymorphism | Polymorphism means different objects can use the same method name but act differently. For example, a `Dog` and `Cat` class might both have a `speak()` method — one barks, the other meows. It makes your code flexible and easier to expand without changing what already works. |
| Operator Overloading | Operator overloading lets you change how built-in operators work for your custom classes. For instance, if you have a `Point` class, you can make + add two points together instead of just numbers. It keeps your code simple and more readable. |

# 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.