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](https://careerfoundry.com/en/steps/your-cf-team#receiving-support) 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”)*

1. 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?
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?** |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |

1. 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.
2. 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.

### 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!)*

|  |
| --- |
| # Script for a Simple Travel App  # Ask the user where they want to travel  destination = input("Where do you want to travel? ")  # Define the three destinations  destination\_1 = "Paris"  destination\_2 = "New York"  destination\_3 = "Tokyo"  # Check the user's input and print the appropriate message  if destination == destination\_1:  print("Enjoy your stay in Paris!")  elif destination == destination\_2:  print("Enjoy your stay in New York!")  elif destination == destination\_3:  print("Enjoy your stay in Tokyo!")  else:  print("Oops, that destination is not currently available.") |

1. 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:** In a job interview, I would explain logical operators in Python as follows:

Logical operators in Python are used to perform logical operations on conditional statements. They allow you to combine multiple conditions and control the flow of your program based on those conditions. The three primary logical operators in Python are:

**and**: This operator returns True if both operands are true. It is used to combine two conditions where both must be true for the overall condition to be true.

**or**: This operator returns True if at least one of the operands is true. It is used to combine two conditions where either can be true for the overall condition to be true.

**not**: This operator returns the opposite of the operand's boolean value. It is used to invert the truth value of a condition.

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

**ANSWER:** Functions in Python are reusable blocks of code designed to perform a specific task. They allow you to organize your code into modular, manageable, and readable sections. A function is defined using the def keyword, followed by a function name, parentheses (which may include parameters), and a colon. The code block within the function is indented.

Functions are useful because they:

1. **Encapsulation**: They allow you to encapsulate a task in a single unit, making the code more modular.
2. **Reusability**: You can call a function multiple times, reducing redundancy and promoting code reuse.
3. **Maintainability**: By breaking down complex tasks into smaller functions, the code becomes easier to read, debug, and maintain.
4. **Abstraction**: Functions allow you to abstract away complex logic, providing a simpler interface to the rest of your code.
5. 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:** In preparation for the next mentor call, here are some notes on my progress towards my goals:

* **Goal 1: Understand Python Basics**:
  + Progress: I have successfully completed several exercises that cover the basics of Python, including data types, conditional statements, and loops.
  + Challenges: Initially, I found it difficult to understand list comprehensions, but with practice, I am becoming more comfortable.
* **Goal 2: Develop Problem-Solving Skills**:
  + Progress: I have worked on multiple problems and exercises that required logical thinking and problem-solving skills. I am getting better at breaking down complex problems into smaller, manageable tasks.
  + Challenges: Sometimes I struggle with optimizing my solutions for better performance, but I am actively working on this.
* **Goal 3: Build a Solid Foundation in Python for Web Development**:
  + Progress: I am currently learning about Python libraries and frameworks that are essential for web development, such as Flask and Django. I have also started working on small web projects to apply what I have learned.
  + Challenges: Integrating different technologies and managing dependencies can be challenging, but I am gaining confidence as I progress.

### 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:** Without file storage, you’d lose all data when the program ends. For instance, user inputs, configurations, or results would be lost, requiring the user to re-enter data or the program to recompute results every time it runs. This would make it impractical to handle large or complex datasets.

1. 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 a way to serialize and deserialize Python objects using the pickle module. Serialization (pickling) converts a Python object into a byte stream, which can be saved to a file or transmitted over a network. Deserialization (unpickling) converts the byte stream back into the original Python object.

**When to use pickles:**

* **Storing Complex Data Structures:** Use pickles to save and load complex data structures (like lists of dictionaries) that would be cumbersome to handle with text-based formats.
* **Efficiency:** Pickling is faster and more compact for complex objects compared to manually converting objects to JSON or XML.

**Why use pickles:**

* **Ease of Use:** The pickle module handles the conversion automatically.
* **Python-Specific:** It is best suited for Python-specific data and objects that might not be easily translated into text formats.

1. 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:  
Current Directory:** Use os.getcwd() to get the current working directory.

**Change Directory:** Use os.chdir(path) to change the working directory to path.

1. 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?

To prevent the entire script from terminating due to an error, you can use **try-except blocks**. This allows you to handle exceptions gracefully and continue running the script or provide informative error messages.

**try:** Contains code that may raise an exception.

**except:** Handles specific exceptions.

**else:** Executes if no exceptions were raised.

**finally:** Executes regardless of whether an exception was raised or not.

1. 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?

* So far, I’ve been enjoying the course and making good progress. I’m particularly proud of how I tackled the pickling exercises and successfully implemented file I/O in Python. I’ve been struggling a bit with handling exceptions effectively, so I plan to spend more time practicing error handling and debugging.

### 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 programming paradigm that organizes software design around data, or objects, rather than functions and logic.  
**Benefits of OOP:**

* **Modularity**: Code is divided into discrete objects, making it easier to manage, debug, and reuse. Each object operates independently, encapsulating its own data and methods.
* **Reusability**: Once a class is created, it can be used to create multiple objects. Inheritance allows for new classes to be created based on existing ones, promoting code reuse.
* **Flexibility**: Polymorphism allows objects to be treated as instances of their parent class rather than their actual class. This makes it easier to interchange objects and extend code.
* **Maintainability**: OOP makes it easier to modify and maintain existing code. Changes to a class do not affect other parts of the program, as long as the interface remains the same.
* **Data Encapsulation**: Objects keep their state private inside a class. Other objects do not have direct access to this state. Instead, they can only call a list of public functions – known as methods.

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

**Answer:**

In Python, a **class** is a blueprint for creating objects. It defines a set of attributes and methods that the created objects can use. An **object** is an instance of a class. While the class itself is a concept or template, an object is a specific realization of that class, holding actual data and methods that operate on that data.

**Real-world Example:**

Consider a class Car. A car has attributes like color, model, and year, and methods like start(), stop(), and drive(). The class defines these attributes and methods, but it doesn't represent a specific car.

class Car:

def \_\_init\_\_(self, color, model, year):

self.color = color

self.model = model

self.year = year

def start(self):

print(f"{self.model} is starting.")

def stop(self):

print(f"{self.model} is stopping.")

# Creating objects

car1 = Car("Red", "Toyota Corolla", 2021)

car2 = Car("Blue", "Honda Civic", 2020)

car1.start() # Output: Toyota Corolla is starting.

car2.start() # Output: Honda Civic is starting.

1. 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 fundamental OOP concept where a new class (called a subclass or derived class) is created based on an existing class (called a superclass or base class). The subclass inherits all the attributes and methods of the superclass, allowing for code reuse and the creation of a hierarchical relationship between classes. Inheritance allows the subclass to add its own attributes and methods or override existing ones. |
| Polymorphism | **Polymorphism** is the ability of different objects to be treated as instances of the same class through a common interface. It allows for methods to be used interchangeably without knowing the specific type of the object. This is achieved through method overriding and interfaces. Polymorphism enhances flexibility and integration in code, making it possible to write more general and reusable code. |
| Operator Overloading | **Operator overloading** allows custom implementations for the behavior of operators (such as +, -, \*, ==) for user-defined classes. It enables objects to interact using natural operators, making the code more intuitive and readable. This is achieved by defining special methods in a class. |

### 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?

**Answer: Databases** are organized collections of data that are stored and managed in a structured way. They enable efficient storage, retrieval, and manipulation of data. Databases use a Database Management System (DBMS) to handle these operations.  
**Advantages of using databases:**

* **Data Integrity and Consistency**: Databases enforce rules and constraints to ensure the accuracy and consistency of data.
* **Efficient Data Retrieval**: Indexing and querying capabilities enable quick access to specific data.

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

|  |  |
| --- | --- |
| **Data type** | **Definition** |
| VARCHAR | Stores variable-length character strings. Suitable for columns where the length of the text can vary, such as names or descriptions. |
| INT | Represents integer values. Used for whole numbers, such as IDs or quantities, without any decimal places. |
| DATE | Stores date values in the format YYYY-MM-DD. Used to represent dates, such as birthdates or event dates. |

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

**Answer:**

**Embedded Applications**: SQLite is ideal for applications where a lightweight, self-contained database is required, such as mobile apps or desktop applications.

**Development and Testing**: It’s useful for local development and testing because it doesn’t require a server setup.

**Single-User or Low-Concurrency Applications**: SQLite is suitable for applications with minimal concurrent access needs or where the database is used by a single user.

**Simplicity and Minimal Configuration**: SQLite does not require complex configuration and is easy to set up and deploy.

1. Think back to what you learned in the Immersion course. What do you think about the differences between JavaScript and Python as programming languages?

**Answer:**

**JavaScript** and **Python** differ in several ways:

* **Use Cases**: JavaScript is primarily used for web development (client-side scripting), while Python is used for a variety of applications including web development, data analysis, and automation.
* **Syntax**: Python’s syntax is known for being clean and readable, which makes it easier for beginners. JavaScript’s syntax can be more complex, especially with its use of callbacks and asynchronous programming.
* **Concurrency**: JavaScript uses an event-driven model for concurrency (e.g., with Node.js), while Python uses threads and asynchronous programming (e.g., with asyncio or multithreading).
* **Type System**: JavaScript is loosely typed (dynamic typing), whereas Python is also dynamically typed but often has more strict guidelines and tools for type hints and checking.

1. 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?

**Answer:**

**Limitations of Python** include:

* **Performance**: Python is an interpreted language, which can make it slower compared to compiled languages like C++ or Java.
* **Global Interpreter Lock (GIL)**: The GIL can be a bottleneck for multi-threaded programs, limiting the performance of CPU-bound tasks.
* **Memory Consumption**: Python’s memory consumption can be higher compared to some other languages, which might be a concern for memory-constrained environments.
* **Mobile Development**: Python is not commonly used for mobile app development compared to 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.

1. You’ve finished Achievement 1! Before moving on to Achievement 2, take a moment to reflect on your learning in the course so far:
   1. What went well during this Achievement?
   2. What’s something you’re proud of?
   3. What was the most challenging aspect of this Achievement?
   4. Did this Achievement meet your expectations? Did it give you the confidence to start working with your new Python skills?
   5. 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](https://careerfoundry.com/en/steps/your-cf-team#receiving-support) 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.*)

1. In your own words, describe the steps you would take to deploy a basic Django application locally on your system.
2. 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](https://docs.djangoproject.com/en/3.2/ref/templates/language/#templates) 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 |  |

1. 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](https://docs.djangoproject.com/en/3.2/ref/models/querysets/). 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:
   1. What went well during this Achievement?
   2. What’s something you’re proud of?
   3. What was the most challenging aspect of this Achievement?
   4. 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.