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 had no prior experience in coding prior to the full stack immersion course. This is my first interaction with python. As a project manager my past experience working to define

deliverables, define timelines, and utilize critical thinking to solve complex issues have helped me in the completion of this course so far.

2. What do you know about Python already? What do you want to know? Nothing as of the start of this course. I would like to learn how to build full applications and ready myself for a career utilizing python.
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. The constant change of programming tools can sometimes make the lessons "outdated" in terms of the steps you need to follow as they may change with new updates to the tools we are using. Understanding that there may be some inconsistencies and using tools such as stack overflow, google, and other message boards, will help find the answer to any questions that may arise. If I cannot solve the problem on my own, having access to my mentor will allow me to ask questions that I was unable to solve.

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 development focuses on the user interface and visual elements for a website. Backend development focuses on the server-side programming and infrastructure. A backend developer would be working on operations which allow the handling user input, implementing business logic, server configurations, performance optimization and database management.

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")
Python has several factors which would make it a more desirable choice over JavaScript.
- Python prioritizes readability with its syntax which helps with maintaining the program and its general readability.
- Python simplifies package management with its built-in management tool pip.
- Python's strong community support ensures access to abundant documentation and resources.

3. Now that you've had an introduction to Python, write down 3 goals you have for yourself and your learning during this Achievement. You can reflect on the following questions if it helps you. What do you want to learn about Python? What do you want to get out of this Achievement? Where or what do you see yourself working on after you complete this Achievement?
   - I want to better understand object-oriented programming.
   - I want to learn how to write scripts to automate tasks.
   - I want to learn more about backend development.

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? iPython provides multiple upgrades over the standard python shell. iPython comes with syntax highlighting (providing more readable code), it is more user-friendly (improved tab-completion, command history, shortcuts) and provides auto indentation for code blocks. You can more easily test smaller code snippets as each command is executed upon entering and the output is immediately displayed.

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	Whole numbers only	Scalar
Floats	Numbers with fractions. (Decimal)	Scalar
String	Characters enclosed in either single or double quotes	Non-Scalar
Boolean	True or False Statement	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. While lists and tuples both allow you to store groups of data in Python, lists allow you to alter the data once it is entered while tuples are unchangeable once you create them. Lists therefore provide greater flexibility while tuples are faster and require less space in your memory.
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 to use dictionaries for this task as the key-value pairing it provides would allow the storing and accessing of data for each associated attribute. The vocabulary word would be the key with the definition and category acting as the value.

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!)*

```
def travel_destination():
    destination = input("Where do you want to travel? ")

    if destination == "Berlin":
        print("Enjoy your stay in Berlin!")
    elif destination == "London":
        print("Enjoy your stay in London!")
    elif destination == "Miami":
        print("Enjoy your stay in Miami!")
    else:
        print("Oops, that destination is not currently available.")

travel_destination()
```

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 are used to manipulate Boolean values. The 'and' operator returns true if both operators are true, otherwise it will be a false statement. The 'or' operator will return true if either of the operators are true and false only if neither operator is true. The 'not' operator is used to flip the result of a condition. Turning true to False and False to True. These operators are used in conditional statements to help control the program flow based on entered values.

3. What are functions in Python? When and why are they useful? Functions are reusable blocks of code tat will perform certain tasks. They help with code organization, readability, and reducing redundancy of code by making the overall code more modular.

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. So far, I feel like we are still only in the beginning stages of what python can do. It is very similar to what I learned in my full-stack course though I have enjoyed the process so far. I am interested in getting into the section where we learn to read from files and parse data. I am also mostly looking forward to the python for backend lessons as this is where I think I will most enjoy working.

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 allows you to store information persistently so that you do not lose your data after a program terminates. File storage allows you to maintain your data between sessions and share information with other 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? Pickles are a module which converts objects into a byte stream to be stored and transmitted. Pickles allow easy storage and retrieval of objects, simplifies data sharing between different programs and facilitates communication between processes.

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? In Python, you can use the **os.getcwd()** function to find out the current working directory. It returns a string representing the current directory path. To change the current working directory, you can use the **os.chdir(path**) function, where path is the desired directory path.

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? Exception Handling in Python allows us to stop the script from terminating in case of an error. Try blocks with except blocks in them allow the handling of specific exceptions that may occur. This way, if an exception is raised within the try block, the script will continue executing from the except block onwards, allowing you to handle the error gracefully and continue with the rest of the script's 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. Currently I have enjoyed this course as I am learning a lot. I had more questions when trying to set up this exercise, however nothing that couldn't be figured out after some researching.

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? OOP is a programming method that organizes code around objects, which are instances of classes. This allows for better code organization, easier maintenance and debugging, enhanced code reusability, and increased productivity through inheritance and polymorphism.

2. What are objects and classes in Python? Come up with a real-world example to illustrate how objects and classes work. Objects are instances of classes which act as the blueprint. A real world example would be cars. Cars have a make, model, and color which act as the object properties. The Methods would be the cars ability to start, stop, and drive.

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 family relationship between classes. The Parent and child classes are the 2 parts of this relationship, with the child class inheriting from the parent. The child class has the ability to add its own attributes as well. This helps with avoiding repetitive code through multiple classes and allows us to create specific class from general ones. Inheritance will only work in one direction from the parent to the child class.
Polymorphism	In Python, polymorphism is when multiple objects of the same class all have the same methods, but they can behave differently. Sort of like the ability of superheroes to perform different actions with the same method name based on their individual powers. For example, imagine a "Superhero" class with a method called "attack." Different superheroes, like "Superman" and "Spider-Man," can inherit from this class and override the "attack" method to perform their unique attacks, such as "Superman" using laser vision and "Spider-Man" using web-slinging. Despite having the same method name, each superhero's attack behaves differently. Polymorphism allows objects of different classes to be treated as interchangeable, enabling flexibility and dynamic behavior based on their specific implementations.
Operator Overloading	Operator overloading in Python is the ability to define how operators behave with user-defined classes. It allows customizing the behavior of

	operators such as +, -, *, /, and others for objects of a class. For example, by overloading the + operator, you can define how two objects should be added together. This enables more intuitive and meaningful operations with custom classes, making them behave like built-in types. By implementing special methods like add or mul, you can define the desired behavior of operators when applied to instances of your class, extending the language's capabilities to suit specific object interactions.

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 allow for easier access and management of data. Using them in coding allows for better data integrity, consistency, security, and enables more complex searching. This allows for more simplified data handling and scalability in applications.

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

Data type	Definition
INT	A whole number data type. This can be used to store integers, for instance, the age of a person or the number of items in a stock.
VARCHAR	A variable-length character string data type. This is typically used to store text information, like names or addresses. The length parameter indicates the maximum number of characters it can hold.
DATETIME	A data type for storing date and time values. This is useful for recording events, like when a record was created or updated.

3. In what situations would SQLite be a better choice than MySQL? SQLite is best when you need a lightweight, file-based database for small-scale applications, local/single-user storage, or for development/testing. It doesn't require a server setup

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? Python in simpler and easier to learn and use in my opinion. Javascript would be better for making dynamic websites whereas Python allows for building smaller programs with cleaner code.

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? Pythons data types can use more memory than languages with statically typed variables. Python can be slower as it is interpreted rather than compiled. Python also relies on third party libraries which can be a challenge to stay up to date with.

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?
An Object-Relational Mapper (ORM) is like a translator between a database and a programming language like Python. It helps you work with databases in a more convenient way by treating database tables as objects. It makes your code simpler and easier to understand, and also protects against security issues. Overall, it makes working with databases in Python much easier and safer.

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? Everything went ok building the app. I would add the ability to back out of an option if I were to select the wrong one if I were to rebuild this project.

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. I have previous experience in creating Python applications that involve database interaction and CRUD operations. In particular, I have worked with the SQLAlchemy library to manage databases. I have developed applications that allow users to create, read, update, and delete data entries. These applications follow an object-oriented approach, utilizing classes and methods to handle various operations on the data. Input validation and data integrity are key considerations in these applications to ensure accurate and reliable data handling. Overall, my

experience with Python includes developing database-driven applications with a focus on functionality, efficiency, and data management.

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? Initial learning of python has been smooth.
    b. What's something you're proud of? Understanding how to use Python and OOP.
    c. What was the most challenging aspect of this Achievement? Just getting used to OOP and SQL app building in general.
    d. Did this Achievement meet your expectations? Did it give you the confidence to start working with your new Python skills? Yes, I am eager to start a new python project to branch out.
    e. What's something you want to keep in mind to help you do your best in Achievement 2? Just to make sure to keep code clean and readable.

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? Using vanilla (plain) Python means you have complete control over your project and can make it exactly how you want. However, it requires more work because you have to write everything from scratch and might miss out on useful features that frameworks provide. On the other hand, using a framework like Django gives you pre-built tools and features that make development faster and easier. You don't have to reinvent the wheel, but you might have less flexibility to customize things exactly the way you want.

2. In your own words, what is the most significant advantage of Model View Template (MVT) architecture over Model View Controller (MVC) architecture? The most significant advantage of the Model View Template (MVT) architecture over the Model View Controller (MVC) architecture is the clear separation between the presentation logic and business logic. In MVT, the template handles the presentation layer, while the model handles the business logic. This separation allows for better code organization, maintainability, and reusability. Additionally, MVT simplifies the development process by eliminating the need for explicit controller components, resulting in cleaner and more concise 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?
   - 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 three goals for learning Django are as follows: 1) understanding Django's features for URL routing, views, forms, validation, and authentication, 2) gaining the ability to create visually appealing applications using Django's template system, and 3) being able to independently develop a Python project with Django, such as building a database to track watched movies or TV shows.

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.*) I pulled up the Disney World website for this. I would start by identifying the major components and functionalities of the website. This would include breaking down the website into Django models, views, and templates. For example, the homepage could be a Django template that displays various sections and featured content. Each section, such as Careers, What We Do, Life at Disney, and Events, could be Django models with corresponding views and templates. I would also consider implementing user authentication and registration using Django's built-in authentication system. Lastly, I would leverage Django's URL routing to map different URLs to appropriate views and templates.

2. In your own words, describe the steps you would take to deploy a basic Django application locally on your system. To deploy a basic Django application locally:
   - Set up a virtual environment to isolate dependencies.
   - Use pip to install Django and other required packages.
   - Create a new Django project using the "django-admin startproject" command and configure the database settings in the project's settings.py file.
   - Next, create Django apps within the project using the "python manage.py startapp" command, and define models, views, and templates within the apps.
   - Run database migrations with "python manage.py migrate" and start the development server using "python manage.py runserver".
   - Finally, access the application locally through the provided localhost address and port number.

3. Do some research about the Django admin site and write down how you'd use it during your web application development. The Django admin site is a powerful tool that can be used during web application development. It provides an easy-to-use interface for managing the application's data and performing administrative tasks. Developers can quickly create, update, and delete database records through the admin site without writing custom views. It automatically generates forms based on the models, simplifying the process of managing data. Additionally, developers can define custom admin actions, permissions, and filters to customize the admin site's functionality based on specific project requirements, making it a valuable tool for efficient 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. Django models are Python classes that represent database tables. They define the structure of the data, including fields and relationships. Models handle the creation, retrieval, updating, and deletion of records (CRUD) in the database. The benefits of Django models include automatic schema generation, database abstraction, built-in validation, and support for complex queries. They simplify database operations, promote code organization, and enhance code reusability, making it easier to develop and maintain data-driven web applications.

1. 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. Writing test cases from the beginning of a project is crucial for several reasons. Firstly, it helps ensure the correctness and stability of the codebase by identifying and preventing bugs early on. For example, in a banking application, test cases can verify that deposits, withdrawals, and transfers are accurately reflected in account balances. Secondly, tests act as documentation, providing clear usage examples and ensuring future modifications don't break existing functionality. Lastly, early test coverage facilitates continuous integration and deployment, enabling efficient development iterations and maintaining a high-quality codebase.

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. Views handle the logic behind processing user requests and generating responses. Incoming HTTP requests are received and then return corresponding HTTP responses.

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? I would choose class-based views in Django. Class-based views provide a more organized and modular approach for code reuse. They allow us to define common functionality in base classes and inherit from them in different views. This promotes code reusability, reduces duplication, and improves maintainability.

3. Read Django's documentation on the Django template language and make some notes on its basics.

    1. Syntax: The Django template language uses curly braces ({}) and percent signs (%) to enclose its tags and variables, respectively. For example, {{ variable_name }} represents a variable, while {% tag_name %} represents a template tag.

    2. Variables: Variables in templates are placeholders that display dynamic content. They are surrounded by double curly braces. For example, {{ article.title }} would display the title of an article object.

    3. Tags: Template tags control the logic and flow within templates. They are surrounded by percent signs. Tags can perform various actions, such as loops, conditional statements, and including other templates.

    4. Filters: Filters modify the output of template variables. They are appended to a variable using the pipe symbol (|). Filters can format data, truncate strings, apply date formats, and more. For example, {{ variable_name|filter_name }} applies a filter to the variable.

    5. Comments: Comments in Django templates can be added using {# comment text #}. They are useful for adding notes or disabling certain sections of code without affecting the template rendering

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.