Python for Web Developers: Learning Journal

Pre-work

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 a friend who went to school for coding initially, and he tried to teach me the basics of Python in Visual Studio Code over online calls. I also had a digital Python textbook from another friend who also went to school for coding, so I tried to use that material to learn as well. I did not get to far with this initial attempt at learning, but it did set me up for pursuing coding in the future, and my prior exposure to the syntax and structure of Python aided in my learning of JavaScript in the Full-Stack Immersion course as I noticed the similarities between the two languages.

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

   I am already pretty familiar with working with strings, variables, conditional statements, and some functions. It has been a couple years since I was last learning any Python material, so I am certainly rusty now.

   What I would like to learn is how to use it for building apps, frontend and back, similarly to how I have learned to use other languages and frameworks to build apps.

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.**

   I anticipate I will have different challenges caused by errors on my part in following instructions from the Exercise readings and tasks. I do my best to follow the instructions to the letter, but there can always be something I overlook or misunderstand. In any case, I know I can always look to online forums like Stack Overflow, ask ChatGPT, contact my mentor, or reach out to the Slack community whenever I am stuck with a project.

Achievement 1: Introduction to Python

Exercise 1.1 Getting Started with Python

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 client-side of an application, which consists of the design, layout, and routing, whereas backend development pertains to maintaining the functionality of URL endpoints, authentication/authorization, accessing data from external sources, and storing data in databases.

   As a backend developer, I would be hired on to implement authentication/authorization logic to endpoints that require it using different models, testing the functionality of each endpoint using a program like Postman, maintaining an external database for storing application or user data, maintaining a hosting service for the application, or monitoring application performance through a service like Atatus.

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?**

   On the surface, JavaScript and Python have some similarities, particularly when it comes to their syntax and code structure. Compared to JavaScript though, Python is a much more user-friendly with its simplistic syntax, making the code easy to read and understand. It also has a simple package management system that comes built-in, making it easy to access a wide array of functions and operations without having to install them ourselves. Like JavaScript, Python also has strong community support.

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 learn more about how Python can be used to create apps and a backend, similar to my past projects. I want to get a better understanding of how this language works similarly to others I have learned while being a simpler syntax. I am not sure what I see myself working on after I complete this achievement, but I know I want to get better at documentation and code structure.

Exercise 1.2: Data Types in Python

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's shell is more user-friendly than the regular Python shell because will display certain things in different fonts and colors, making them stand out more where it counts. It allows for testing pieces of code quickly as each command is executed right after typing and pressing Enter. It also automatically indents when writing nested statements, similarly to regular coding.

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)	All whole numbers, positive and negative	Scalar
Float	All decimal numbers, positive and negative	Scalar
String (str)	Array of characters, alphanumeric and symbol	Scalar
Boolean (bool)	Conditional values, True or False	Scalar

3. **A frequent question at job interviews for Python developers is: what is the difference between lists and tuples in Python? Write down how you would respond.**

   They are both considered arrays that can store any data type, but lists are mutable and can be changed. Tuples cannot.

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.**

   Vocabulary

   Each word would be their own dictionary in the following structure:

   vocab_word = {

   "name": <"name">,

   "category": <"noun, verb, etc.">,

   "definition": <"definition">

   }

   Each category could also be selected from a tuple containing the different options, since there should not be a need to modify this data.

   Language

   Each coding language would be a list containing an array of their respective vocabulary word dictionaries. This way, the words can be accessed quickly and easily modified if necessary.

   language1 = [

   vocab_word1,

   vocab_word2,

   ...,

   ]

   language2 = [...]

Exercise 1.3: Functions and Other Operations in Python

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 == "Venice":
    print("Enjoy your stay in Venice!")
elif destination == "New York City":
    print("Enjoy your stay in New York City!")
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 consist of and, or, and not, and they are used in conditional statements. and checks if two statements are both true, and if they are

not, the result is False; otherwise, the result is True. or checks if at least one of two statements is True, and if at least one is, the result is True; otherwise, the result is False. not results in the opposite of a conditional, so something that would be True becomes False, and vice versa.

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

   Functions in Python are defined manually and can be used to create variables that are only used within itself, while also being able to access variables defined outside. They are useful when needing to perform a certain task multiple times within a script file, removing the need to reenter the same code repeatedly.

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.**

   The course material hasn't gotten too into app building or backend yet, but I know the backend part is in the next Achievement. I'm enjoying becoming refamiliarized with this language from my past experience with it. It is certainly similar to Python in many aspects, but it is also very different. I actually like that this course is requiring more use of README files to act as a breakdown of each Exercise Task. It helps me become more comfortable with writing them, especially given the unique format in which they have to be written in a code editor.

Exercise 1.4: File Handling in Python

1. **Why is file storage important when you're using Python? What would happen if you didn't store local files?**

   File storage in Python is important since, when using data in a script that needs to be accessed multiple times, information that is only stored in the script is destroyed when the file is closed. Having a separate .txt or .bin file that can store data that needs to be regularly accessed, such as lists or dictionaries, prevents data loss and allows the data to be used repeatedly across multiple scripts.

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 packaged stream of bytes converted from complex data. They are useful for writing to and reading from external files consisting of such data structures as lists and dictionaries. Also useful for reading data from binary files.

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?**

   The function used to determine the current working directory is os.getcwd(). To change to a different directory, os.chdir() would be used.

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?**

   To avoid this situation, a try-except-else-finally block can be used. "Try" is used to attempt some code that could result in an error. "Except" can be used to account for different types of errors and inform the user what went wrong. "Finally" can be used to continue code in the block, even if a return statement occurs before it. "Else" can be used to run some more code only if the Try block runs successfully.

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.

This chapter involved a bit more unfamiliar material compared to what I learned prior to CareerFoundry. Wrapping my head around pickles and things like the try-except block took me a little bit to get used to, and I can't say that I'm comfortable with it yet. I would say the thing I'm struggling with the most is getting a script started, or at least knowing where to start. I get a pretty good idea of how something is supposed to work when I see it in front of me, but when I need to write something from scratch, I can have a hard time knowing exactly what to write first. But it helps to revisit examples from the reading.

Exercise 1.5: Object-Oriented Programming in Python

1. **In your own words, what is object-oriented programming? What are the benefits of OOP?**

   OOP is the process of building custom objects and classes that can be reused multiple times, reducing the need to repeat code blocks multiple times for different methods.

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

   An object consists of data types that can be used in methods, and classes act as a template that consists blueprints for creating multiple objects and methods. An example of objects would be multiple food items on a menu for a restaurant, while an example of classes would be a collection of attributes food items would consist of and methods for preparing the food items.

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

Method	**Description**
Inheritance	When using the attributes or methods of one class (parent) in another (sub)
Polymorphism	When using a part of a class, attribute or method, in multiple classes for different operations while still retaining its name
Operator Overloading	Using a unique function definition (e.g., __add__()) to define methods in a class that uses operators (+, -, *, /)

Exercise 1.6: Connecting to Databases in Python

Coming soon