CareerFoundry Learning Journal

Python for Web Developers

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?

    a. I have completed the CareerFoundry Full Stack Web Development Immersion course, in which I became familiar with HTML, CSS, JavaScript, and the MERN stack (MongoDB, Express, React, and Node), plus other frameworks and libraries. I also dabbled in Python a year before beginning CareerFoundry, building small, simple applications for my boss' video game (automating an attack outcome, possibilities, etc.)
    I think having completed the previous CareerFoundry course will help me in this course, programming aside. I understand how many hours I need to put in each week in order to complete assignments, how to stay organized and focused, and the general flow of CareerFoundry fitting into my current life and job.

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

    a. I know that it is an easy to read/understand language, and that many jobs specifically ask for Python. I want to know more about the frameworks and libraries that come with Python, especially compared to JavaScript now that I have experience with that language as well.

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.

    a. I think the biggest challenge that might arise in this course (which I sincerely hope does not come to fruition) is having to change mentors. This occurred four separate times in my Immersion course, which led to me waiting for a new mentor so I could move on in the course and caused me to get behind in the program/my timeline. Another issue could be me falling behind on my own, but I believe I am diligent and will keep on track.

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?

   a. Frontend web development is everything that the user sees and interacts with, whether that be a browser or application interface or client, on the computer, a mobile device, or another device. The backend of web development is essentially everything else that makes the frontend work: servers and APIs, databases, files, compiling, processing, requests, and communicating with servers.
   If I was working on the backend programming for a web app, I would be working on data and user requests (such as HTTP requests), configuring databases (relational/non-relational), and other server and API configurations.

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?

   a. A big difference between Python and JavaScript is that Python is used to develop only the back-end of an app, whereas JavaScript is used to develop both front-end and back-end. Assuming this is a back-end project, Python would be a better choice. Python is also extremely simple to read and maintain, whereas JavaScript's flexibility makes it a little more complicated. Python also offers built-in package management through pip, which makes integrating external resources far more simple than would be with JavaScript.

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?

a. I want to learn Python well enough to write my own programs easily. I want to go into an interview for a job that requires Python and feel good completing the technical interview, and be ready to contribute to a team working with Python. After this Achievement, I'd like to add more Python projects to my portfolio to make myself competitive in Python-required job opportunities.

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?

    a. The default Python shell is less user-friendly, especially when it comes to reading code. The text is all the same color which makes it difficult to distinguish individual lines of codes, and you have to manually indent when writing functions and other nested statements. The iPython shell has syntax highlighting with contrasting fonts and colors. It also automatically indents for you. Additionally, each command is executed immediately after you type it in, which helps the user test small chunks of code more quickly and easily than the default 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 (positive and negative) without any fractional parts	Scalar
Decimal Number (float)	Numbers (positive and negative) with fractional	Scalar

	parts/decimals	
String(str)	Sequences of characters enclosed in quotation marks (single or double quotes)	Non-Scalar
Boolean (bool)	Truth values; either True or False	Scalar

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

   a. A tuple is a linear array that can store multiple values of any type. A list is similar to a tuple, but is mutable - you can modify, rearrange, insert, and delete internal elements of a list, unlike elements in a tuple. Tuples are faster to read and access than lists, but lists are better for situations where reordering or modifications may be necessary.

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 favorite 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.

5.
   a. I would use a dictionary in this case, as well. The vocabulary word would have key-value pairs, such as its definition and category, and more keys could be added as the user advances.