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.

    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.

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:
    a. The script should ask the user where they want to travel.
    b. The user's input should be checked for 3 different travel destinations that you define.
    c. If the user's input is one of those 3 destinations, the following statement should be printed: "Enjoy your stay in _____!"
    d. 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."

```
acceptable_destinations = ['Spain, Japan, Mexico']

preferred_destination = input('Enter the destination to which you want to travel: ')

if preferred_destination == 'Spain':
    print('Enjoy your stay in', preferred_destination, '!')
elif preferred_destination == 'Japan':
    print('Enjoy your stay in', preferred_destination, '!')
elif preferred_destination == 'Mexico':
    print('Enjoy your stay in', preferred_destination, '!')
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

    a. Logical operators in Python include "and", "or", and "not", and return a boolean - True or False. The "and" and "or" operators take two or more conditions. The "and" operator will only return true if both conditions are true. If either one or both are false, it will return False. The "or" operator will only return False if both conditions are false. "Or" only needs one condition to be true to return True (but if both conditions are true, it will still return True). Unlike "and" and "or", the "not" operator can be used with just one condition. It reverses the logic result of the expression that comes after it, making a True statement False, and a False statement True.

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

    a. A function is a set of instructions that process code to achieve certain things.. There are built-in functions, such as input(), range(), enumerate(), len(), append(), and many others, that can be used without being defined in the script.

You can also define your own custom function, which is useful when you need to repeat certain steps throughout your code. With a function, you can just call the function instead of rewriting that code every time it needs to be used.

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.

    a. I am incorporating the Python tools I'm learning into my daily code practice/challenges, and trying to solve each problem with both Python and JavaScript to the best of my ability. I feel confident with the work on this Achievement's project, and feel like this is coming very naturally and that I'm picking up the concepts well. I also feel like this project is making me want to do more projects in Python, as the logic is fun to interact with and feels like I'm writing math expressions.

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?

    a. When a script stops running in Python, the data no longer exists, and cannot be retrieved for later use. File handling allows the data to remain stored on the machine, even after the program has finished executing.

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?

    a. A "pickle" is a packaged stream of bytes, converted from a more complex data structure. You would want to use pickles in situations when retaining the structure of data is too difficult in the form of regular text (for example, a recipe dictionary!). The pickle is then written into a binary file, which can store the information in a way that is readable by a machine, but not by humans.

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

    a. The Python function to find out which directory you're currently working in is os.getcwd(). To change your current working directory, you would use the os.chdir() function.

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?

    a. In this case, I would insert a try-except block. The try block holds the code where the expected error would occur. If no errors arise, the code continues as usual. If an error does occur, the except block appears with an error message and a guide to fix it. With the except block, the code will continue to run afterward, instead of a return and the termination of the script.

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.

    a. I really feel like Python is still coming naturally to me. However, I also feel like this Achievement has WAY more hands on learning, exercises, and practice tasks, which is helping to drill the information, as opposed to the Immersion course which felt like it just threw information at me and hoped it stuck. I also appreciate my mentor's genuine feedback on my code with each submission. The feedback makes complete sense, and isn't cookie cutter copy and paste responses! So helpful!!

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?
2.
    a. Object-oriented programming, or OOP, allows you to assign specific methods and attributes to a specific subset of objects. This is beneficial because it keeps code non-repetitive, non-redundant, and efficient.

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

    a. In Python, everything is an object. They are usually organized by the type of data that they contain, such as numbers, text, sequences, and dictionaries. Classes are an overarching template of the type of object.

    A real world example of this would be students. Every student has a name, a birthday, an ID number, a grade level, a grade point average, etc. These are different for each student, but each student has these attributes.

4. 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 concept that allows you to use methods from one class in another. Instead of copying the code from the first class (also called parent or base class), the second class (also called subclass or inherited class) can “inherit” it. You call which class to inherit properties from in the parenthesis after the naming of the new class. It is important to note that inheritance only works in one direction: from parent to subclass. Also, if you define a method for a subclass, it will use that method instead of inheriting the parallel parent method.
Polymorphism	Polymorphism is where a given data attribute or method has the same name across different classes or data types, but has different functionality depending on where it was defined. An easy example is assigning the method speak() to different classes of animals. speak() for the class

	Cow might return “MOO”, while speak() for the class Pig could return “OINK”, and so on. The len() method is a built-in function that is already an example of polymorphism; len() returns different results for different types of objects, like lists, dictionaries, and strings.
Operator Overloading	Python has built-in operators, including +, -, >, <, >=, <=, ==, and !=. However, when using an operator on a custom class, you will get a TypeError - unless you use operator overloading. This involves defining your own methods for these operators. There are specific method names for the operators, making it easy to write operator overloads. + is add(), - is sub(), < is lt(), > is gt(), <= is le(), >= is ge(), == is eq(), and != is ne(). Once you define the special operator method in your class (similar to the initialize method, it is formatted as __add__()), you can use the operator symbol as normal within the method.