CAREERFOUNDRY

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 went to a technical high school (HTL in German) where learning C++ and a little bit of HTML was part of the curriculum. I also did a couple tutorials on REACT on scrimba.

2. What do you know about Python already? What do you want to know?
   Python is one of the most popular programming languages out there.
   It's syntax is different from javaScript of course, but quite simple.
   It supports a wide range of applications, from web development to data analysis, with vast array of libraries.

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 really don't know, as I am completely new to python, but I managed to complete each part of this course so far, so I know I will complete this part as well. If I am stuck, I know my mentor will be there for me.

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 visual and interactive aspects of a website that users interact with directly, while backend development deals with the server-side logic, apis and databases that power the website behind the scenes.

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 and JavaScript are both very popular dynamically typed scripting languages, making them easy to write and read. Python's extensive package management, built-in web development operations, and strong community support can streamline our workflow and speed up deployment, making it a more efficient choice for our project.

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 the basics of programming with python. The syntax, best practices, it's object oriented nature and how to use all of that to build a website.
   After this achievement I am going to work on achievement 2 ;-)

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 shell has better readability, due to highlighted syntax and auto indentation. Also, it's faster and handier for testing small pieces of code, because each command is executed immediately.

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	negative and non-negative numbers (from zero to infinity	Scalar
Boolean	Is either true or false	Scalar
Tuple	Linear array for storing multiple values	Non-scalar
Dictionary	Stores values and objects within itself indexed by identifiers, or keys. It's an unordered set of items, each of them a key-value pair, where each key is unique.	Non-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.

   Besides their syntax (Tuples using (), lists using []) list are mutable whereas tuples are not, so lists can be used, when you need a collection of item that can be modified and tuples can be used when the collection of item should not change.

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 use a list of dictionaries (flashcards) to structure the data of this app.
Something like:

```
      flashcards = [
                  { "word": "apple", "definition": "A fruit", "category": "noun" },
                  { "word": "run", "definition": "To move swiftly", "category":
"verb" },
      ]
```

The flashcards list is mutable so the user can easily add or remove a flashcard dictionary.
Using dictionaries for flashcards offers flexibility for future adaptations and also makes it easy to search for flashcards with specific keys. Also, dictionaries can be edited easily whereas tuples can't.

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."

```
destinations = ["Salzburg", "Berlin", "New York"]

user_destination = input("Where would you like to travel? ")

if user_destination in destinations:
        print(f"Enjoy your stay in {user_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.

    There are 3 logical operator ins Python, which are essential in decision making and control flows:
    1. and: returns true, if both inputs are true, otherwise returns false
    2. or returns true, if one or both inputs are true, otherwise returns false
    3. not inverts the truth value of the input

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

    Functions are sets of instructions in your code in order to achieve certain goals.
    If you have some steps for a specific task or operation in your code that you need multiple times throughout your script, you can create a custom function (consiting of those steps) so you don't need to repeat the same code over and over again. Using functions saves time, makes your code more readable and easier to debug.

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.

   I wanted to learn Pythons bascis, it's sytanx, it's object-oriented nature and how to use all of that to build websites.
   My progress so far:
   I now know about the data types in python, which is absolutely essential.
   I am getting comfortable with pythons sytanx.
   I learned how to write and use custom functions while keeping the scope of variables in mind.
   I learned how to use while loops, for loops and control flows.

   So, I reached my goals in covering the bascis and the sytanx. (At least in part).

   The Object-Oriented side of python will will be covered 1.5 and the final goal with have to wait a little longer ;-)

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?

   Because variables used to keep track of values no longer exist, once the script stops running. Python uses text files and binary files to read and write data.

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 used to converts complex data into a packaged stream of bytes, which are written into a binary file.
   If the data is more complex than just simple text (e.g. Dictionaries), I would use pickles.

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?

   Get current directory: `os.getcwd()`
   Change directory: `os.chdir()`

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?

```
Try:
      some error prone code
      some error prone code
except:
    notify the user of the error, or
    probably fix the error
else:
     will run only if the try block does not encounter any errors
finally:
     will run no matter what happens, even if there was a return block before it

move on with the rest of your code
```

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.

   I am starting to get a feel for Python and I have to say, I like working with Python.
   Exercise 1.4 went quite smooth, so it feels like I am progressing and that I am currently on a good path forward.
   Nevertheless, I need more practice using Python to remember function names and the syntax at times.