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?**

Before the course, I didn't know anything specific about programming. In the Intro and Immersion course, I learned a lot about HTML, CSS, JavaScript, Node.JS, React, Angular, React Native, and a lot of packages that I used within them.

Programming is challenging at times, but I like challenges. For example, I like to play games on the hardest difficulty possible and earn all achievements in them. I used this kind of approach, to go through the course this far, and I'm enjoying it.

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

I knew Python is a programming language, and that it is the best language to start to learn coding. Then, in the Immersion course, I learned, that Python is mostly used at the backend of an application because it needs to be transpiled for the browser to be able to read and utilize it.

I'd like to know how exactly Python works and what it is capable of.

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 the week that might be favorable to your facing challenges and growing. Plan for how to solve challenges that arise.**

The first challenge will be to get used to the new language we learn here, and how differently it will be used. My curiosity will possibly help me through this course, and the thought, that I would like to work as a developer.

At challenges in the tasks firstly I will check the documentation to find a solution to my problem. If it doesn't help, I will search for a similar problem someone had at Stack Overflow. If none of them works, I will ask my Mentor for some advice.

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 deals with the client side of an application. They work on how the application looks and feels to a user, how to send requests to the backend, and how to read the data returned from a request.

Backend development deals with the server side of an application. Their job is to elaborate on what data the application will use, how and where will the data be stored, and how will the client side communicate with the database.

I would be working on a REST API, which will communicate between the client side and the database, create models of how the data should be stored in the database (if the project works with a NoSQL database), validate requests, and let only valid data get stored, and last but not least, creating the logic, which allows only specific origins to communicate with the API.

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?**

It is one of the easiest languages to learn and understand, and much easier to debug or document it due to its readability. It uses simple built-in package management, which is useful in the way we don't need to look for third-party libraries in our project and the most common web operations are available in it as well. It has one of the largest communities, and it's the best language for fast prototyping.

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 do you see yourself working on after you complete this Achievement?**

   1: Learn how to use Python effectively.
   2: Progress through the Exercise
   3: Continue with Achievement 2.

Exercise 1.2: Data Types in Python

1. **Imagine you're having a conversation with a future colleague about whether to use 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 uses color schemas to distinguish line of codes, suggestions, etc.
    - It gives more readability to the shell
- iPython has an autocompletion tool
    - It writes all the possible functions usable with the sequence, considering its data type. It can be chosen by pressing 'Tab' navigating to the right function with the arrow keys, then pressing 'Enter'

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?
string	can store alphanumeric data with special characters surrounded by single or double quotes	Non-Scalar
boolean	can store either true or false, which can be used for example as statements to execute specific code lines	Scalar
integer	can store whole numbers in positive and negative directions as well	Scalar
list	Lists are able to store every kind of data type, even other lists, surrounded by square brackets '[ ]', and they are mutable.	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 response.**
   Tuples are ordered, immutable data storages surrounded by parentheses.
   Lists are ordered, mutable data storages surrounded by square brackets.

Tuples are processed and read faster, but they have the disadvantage of immutability. If we would want to modify its content, we have to add all the information again. They are most useful at working with a huge database.

Lists, on the other hand, have larger memory usage and are slower to read, but their content can be reassigned, modified, and new data can be added to it easily.

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.**

   Since flashcards don't usually need changes, tuples would be the best choice in my opinion. Due to its fast processing behavior, the application would run faster, and when a user would like to change something in a word's data, the frontend would fill out every input field, so the user can update data with minimum effort. When the development of the application continues, the small memory usage will come handy, for example, if a lot more languages are added to the application.

Exercise 1.3: Data Types 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 practive 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."**

Answer:

```
destinations = [ Berlin, Prague, Budapest ]
user_destination = input(“Write your wished travel destination: “)

if user_destination in destinations:
        print(“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 operations in Python”. Draft how you would respond.**
   Answer: Logical operators are ‘and’, ‘or’, and ‘not’. They will return the statement ‘True’ or ‘False’.

   The ‘and’ operator will only return true if the statements before and after the operator are true. Otherwise, it will return false.

   The ‘or’ operator will return true, if either the former or latter statement is true. If none of them returns true, the operator returns false.

   The ‘not’ operator will return the opposite result, what it normally would be. For example, if the statement is ‘3 + 2 == 5’, normally it would return true. But if we write the ‘not’ operator before the statement, it will return false.

3. **What are functions in Python? When and why are they useful?**
   Answer: Functions can read and modify our data types. There are two function types: built-in and custom.

   Built-in functions are usable for everyone and almost everywhere in our code (it depends on the data types we are using the functions on). We can do every kind of thing with them. For example, converting data to another data type, adding extra content to a variable, modifying, or removing our data.

   Custom functions on the other hand are made manually by the developer, and it will execute the code block that is related to that function after that has been called.

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.**

Answer: I fell behind schedule a bit, but I'm on the track to complete the goals I was set. They were:

1: Learn how to use Python effectively.
2: Progress through the Exercise
3: Continue with Achievement 2.

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?**
   Answer: It is important to store data in files, because otherwise when the script you are running is finished, all data the script created and worked with would be terminated. But if you store the created data in local files, you can access them later.

2. **In this Exercise you learned about the pickling process with the `pickle.dump()` method. What are pickles? In which situation would you choose to use pickles and why?**
   Answer: Pickles are data, that have been disassembled into bytes and will be stored in a file with the '.bin' extension. It is very useful if you would like to store more complex types of data (for example, dictionaries), otherwise, the more complex data should be converted to strings and stored as texts, or it would not be possible to store them.

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?**
   Answer: You have to import 'os' module, then you can use os.getcwd() to see in which directory you're currently working. If you would like to change the directory, you have to use 'os.chdir("<path to desired folder>").

4. **Imagine you're working on 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?**
   Answer: I would use 'try-expect-else-finally' statements. The questionable block of code would go to the 'try' block, I would print a message at the 'except' block, to let me know if there is an error, and then to the 'else' block, I would add what the code should execute if the try block is successful. The 'finally' block can have code lines, what I want to execute anyway, and/or print a "Script end." text.

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 mentor call.
Answer: I feel proud, that I know more and more about Python after every Exercise. Sometimes I struggle to find the error because the error message is often 'blended' with the data I wanted to print. I feel like I have to use more of the functions to really get to know them and use them effectively (possibly after the course as well).

Exercise 1.5: Object-Oriented Programming in Python

1. **In your words, what is object-oriented programming? What are the benefits of OOP?**
   Answer: In object-oriented programming, we strive to create custom classes, and sort every data as instances into one of these classes. The goal is to reduce code repetition and make the data creation more manageable. The benefits are reusable functions and initiations, and the inheritance possibility through classes. Inheritance is to pass defined functions from the parent class to the subclass, so we don't have to repeat our functions.

2. **What are objects and classes in Python? Come up with a real-world example to illustrate how objects and classes work.**
   Answer: Objects are the data, that we are adding as instances to a class. Classes are like a container of objects, where we can define custom functions of what we can do with the data.

   If we didn't use classes, we had to use dictionaries, tuples, or lists to add key-value pairs to every data we wanted to store. With a great amount of data, it would be really hard and time-consuming to handle. Using classes, we can pass all data as attributes when we are creating a new instance of a class, and it will automatically add the attributes as values to the corresponding keys.

3. **In your own words, write brief explanations of the following OOP concepts; 100 to 200 words per method is fine.**
   Inheritance: It is a way to pass functions and initiation from the parent class to the subclass. For example, if we have a parent class called 'Animals' with the function 'get_age()', to get the age of a specific object the Animals class has. If we create a subclass of the 'Animals' class called 'Cat', we can use the 'get_age()' function on every instance the 'Cat' class has. But, if we define a function in the 'Cat' class, we won't be able to use it on objects in the 'Animals' class. The inheritance works only one way. Creating subclasses

under the ‘Cat’ class will inherit the defined methods both from the ‘Animals’ and ‘Cat’ classes.

Polymorphism: Polymorphism means we can use the name of predefined or already defined functions in a class, and it won’t have the same result unless we don’t define it ourselves. For example, we can define the ‘sort()’ function, but it will add predefined integers together if we want it to (but it would be a bad practice to do so). Or we can have two classes, that use the same function name, but the end result is different. For example, the ‘Cat’ and ‘Dog’ classes can have the same ‘speak()’ function, the ‘Cat’ class will return ‘Meow’, but the ‘Dog’ class will return ‘Bark’.

Operator Overloading: Normally, we can’t use any operation on a class, so we have to define them ourselves, using the corresponding name of the operation (for example ‘add’ for ‘+’), surrounding with two underlines at both sides (‘def __add__’), and we have to write all the functionalities we want it to do. After defining it, we can add two objects of this class together (for example: ‘class_instance_1 + class_instance_2). We have to use the same method to compare objects within the class, we can then modify them accordingly. For example, if we would like to sort the instances of a class, we have to check which instance is less than the other, and then sort them appropriately.