Exersice 1.1

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 had no knowledge of coding before registering for the Full-stack developer course with careerFoundry. I worked as an Account Executive for FMCG companies which is not related to programming.

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

I have heard that Python is a popular programming language that is beginner-friendly for non-programmers to learn.
I would like to know how to use Python to build a game because I would like to create one.

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 biggest challenge for me is that I have only 3 weeks to complete this course. (If not then I need to pay extra money and void the job guarantee.) Moreover, I am not so good at learning anything new with reading only texts and this course is full of heavy texts. My plan is to submit at least 5 tasks per week. There is no space for failing.

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 web development focuses on creating the user interface and experience of a website or application, involving technologies like HTML, CSS, and JavaScript to design and implement the visual elements and interactivity that users interact with directly.
Backend web development, on the other hand, deals with server-side logic and database management, using languages like Python, Ruby, or Node.js to handle data, user authentication, and the overall functionality that supports the front end.

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?
Python and JavaScript are both high-level programming languages with versatile applications, yet they serve distinct purposes. They share similarities in being interpreted, dynamically typed, and having active communities.

Python stands out as the superior choice due to its unparalleled readability, versatility, and robust ecosystem. Its clean and intuitive syntax fosters efficient collaboration and reduces development time, making it an ideal language for teams.

Write down 3 goals you have for yourself and your learning during this Achievement.

1. Be able to code with Python effectively
2. Have Python projects on my portfolio
3. Convince the HR and IT team lead that I am well-versed in Python

Exersice 1.2

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?

The iPython Shell offers numerous benefits over the default Python shell due to its enhanced features and capabilities. It provides an interactive and user-friendly environment with features like tab completion, syntax highlighting, and history navigation, making code exploration and debugging more efficient.

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, both positive and negative, without any decimal points	scalar
Float (float)	numbers with decimal points, allowing for representation of real numbers	scalar

String (str)	sequences of characters, such as text	non-scalar
List (list)	ordered collections of items that can be of any data type, including other lists	non-scalar

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.

The key difference between lists and tuples in Python lies in their mutability. Lists are mutable, meaning their elements can be modified, added, or removed after creation, making them suitable for situations where data needs to change. Tuples, on the other hand, are immutable, meaning their elements cannot be modified after creation, providing stability to the data. This makes tuples suitable for cases where data should remain constant throughout the program. Lists are defined using square brackets [], while tuples are defined using parentheses ().

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.

The necessary data types would be String('str') and Dictionary('dict') to represent the information

String: To store vocabulary words, definitions, and categories. These textual data can be stored using the string data type.

Dictionary: Dictionaries would be a suitable data structure to store flashcards. Each flashcard can be represented as a dictionary with keys like "word," "definition," and "category," where the corresponding values hold the actual word, definition, and category information. Dictionaries allow you to organize and access flashcard data efficiently using meaningful keys.

Exersice 1.3

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."

```
available_destinations = ["Bangkok", "Tokyo", "London"]

user_input = input("Where do you want to travel? ")

if user_input in available_destinations:
    print(f"Enjoy your stay in {user_input}!")
else:
    print("Oops, that destination is not currently available.")
```

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 are used to combine and manipulate boolean values (True or False). There are three main logical operators: "and", "or", and "not".

The "and" operator returns True only if both operands are True.

The "or" operator returns True if at least one of the operands is True.

The "not" operator is a operator that negates the boolean value, turning True into False and False into True.

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.

Today I gained a deeper understanding of conditional statements in Python, mastering the art of making decisions within my code. Explored the realm of loops in Python programming. I feel more confident in my grasp of these fundamental concepts, ready to apply them creatively to a variety of coding challenges.