Python – CF – Learning Journal

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 web development is related to development of programs that are interacting directly with the User through the Web browser (Chrome, Safari etc…). Resulting Apps are executed by these browsers and rely on HTML, CSS and JavaScript.

Backend web development on the other hand is related to some code that is run on a server instead of a web browser. It is usually used for more complex programs that often are interconnected with a database. Main backend languages include JavaScript, C, C++, Ruby etc…

If I was hired to work on backend programming for web app, I would like to work on operations that make use of API and databases that make use of mongoDB technology (or similar NoSQL technology) as I found it very flexible technology (scale up, adjust etc…).

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 is easy to learn and understand. Like JavaScript, it is a scripting language however from the beginning it was developed with the idea of a quick development in mind in particular with a great readability (indentation to separate blocks of code for instance). Simple built in package management as well as strong community support are also great benefits that support the development of the App using Python.

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 would like to learn how to develop a full stack App using Python (Front and Back end) as we did using Node.js in order to get to understand better on real project the advantages of each languages
- I would also like to learn more about Python as I understand it can be used for data science
- In the future, I believe Python would surely be an option for some of the Apps I would be working on and therefore I would need to be able to contribute to the discussion around the choice of Python or another language

1.2: Data Types in Python

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 is much more **user friendly**, in particular when it comes to **reading the code** (it will display different features of the code in contrasting fonts and colors) but also, iPython Shell will display some **guidance** (in particular a suggestion of possible methods that can be applied to the program variables for instance).

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?
int	This data type represents integers, including both negative and non-negative numbers (from zero to infinity)	Scalar
strings	Strings in Python are an immutable array of characters	Non-Scalar
tuples	Tuples are linear arrays that can store multiple values of any type.	Non-Scalar
dictionaries	A 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.

Although both datatypes are ordered sequences, lists are much more flexible than tuples. This comes from the fact that lists are mutable while tuples are immutable. This allows to modify the different elements inside the list as well as to re-order the elements (sort etc….) which cannot be done with tuples.

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.

This app will need some sort of sequence of words to be studied. For such a sequence we may use either list data type or tuples. We can imagine that the sequence of words may need to be extended by the user over time. Also we may want to give the possibility to sort the element of the list (randomly for instance). Due to the fact that tuples are immutable, tuples don't seem to be the right choice and list datatype would be best in this case to store the words to be studied.

When it comes to each vocabulary word, as each of them includes various properties (definition and categories in particular), we would need to use a dictionary data type where definition would be a string and category would be a string as well.

1.3: Operators & Functions 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 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."**

Write your script here. (Hint: remember what you learned about indents!)

```
# Defines travel destinations
travel_destinations = ["Lisbon", "Oslo", "Roma"]

# Asks user to enter a destination
user_destination = str(input("Where do you want to travel?: "))

# Creates a flag to check whether there is a match between the
destination entered by the user and the destinations in the
travel_destinations list
destination_match = False

# Loops through the destination elements available in the
travel_destinations list
for destination in travel_destinations:
  # If the destination entered by the user is equal to one of the
destination available in the travel_destinations list it will print
out "Enjoy your stay in ... !"
  if (user_destination == destination):
    destination_match = True
    print("Enjoy your stay in " + destination + "!")
    # Exit the loop in case there is a match
    break
  else:
    continue

# If at the end of the code after iteration on all the elements of
the travel_destinations list there was no match then the
destination_match flag will remain False and therefore we can print
out "Oops, that destination is not currently available."
if destination_match == False:
  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.

Logical operators in Python include **and**, **or** as well as **not**.

All of these logical operators return a Boolean. Either True or False.

And logical operator returns True in case all statements are True. Otherwise it will return False.

Or logical operator returns True in case at least one statement is True.

Not logical operator returns the opposite of the result (ie. False in case the result is True).

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

Functions in Python are lines of code grouped together and that can be called at some point in the program.
It is useful when we want to use some built-in function (len, sort etc…) for instance and manipulate our code.
It is also helps when we need to repeat some operations several time. It would save some lines of code.
It also make our code much more readable as it allows to group the code and keep it clean.
In addition, it can also prevent the user from modifying some variables in the program (scope).

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 believe I could already progress in my general understanding of Python and how it compares to JavaScript in particular (syntax for loops, else if, etc….)