Python
Learning Journal

Pre-Work: Before Starting

Reflection questions

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?

   Answer: Prior to this specialization course on Python, I've completed a frontend and a full-stack development course at CF. Throughout these courses, I built a website, several web apps and a mobile app. In order to do so, I've used different technologies / tools such as HTML, CSS, JavaScript, Bootstrap, Node.js, MongoDB, PostgreSQL, React, React Native, Angular, and more. I also learned how to develop different coding technics such as the test-driven development (TDD) or the behavior-driven development (BDD) methods.

   Before enrolling into CF, I also had different working experiences that were not related to coding, but related to data and technologies, which could help me in this course and my coding learning in general.

   For example, I worked as a Senior analyst in data and technology policies, where my main focus was oriented towards data governance, data lifecycle and R&D on sensitive data sharing across organizations. All the work carried on while I was occupying this position was done in an Agile environment. Before that, I also worked as a Professional researcher on a project related to supporting decision-making in a municipal infrastructure development context, using GIS and non-GIS data. During this project, I've been exposed to some Python codes, but only as an observer, not a developer.

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

   Answer: I don't know much about Python so far, except that it's a widely used language across different domains, and that it's one of the most popular coding technology among developers.

   I am very enthusiastic about learning Python. I would like to learn everything useful about this language, from how to set a Python coding environment, to how Python codes are generally structured, what are Python specificities and what makes it a better tool than another language on different types of projects. In general, I would like to acquire a good Python knowledge, both on theory and practice, allowing me to use Python whenever it would be the best tool for X project.

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.

   Answer: As for the challenges that may arise, the main one I am thinking of is the ability to distance myself from the coding concepts, patterns, and structures of other coding languages

I've learned so far (mainly JavaScript), if necessary. I am wondering if, in order to learn Python, it is necessary to distance myself from JavaScript concepts and principles, or if, in the opposite, both languages are close to each other and so knowing one makes it easier to learn the other.

Learning a completely new language also represents a good challenge, since it requires more time and effort in comparison to developing additional knowledge in a language for which the basis is already well-know. This learning from the scratch on Python is however extremely important in order to have a solid foundation on which to build further knowledge.

In order to face these challenges and allow me to successfully acquire a good Python knowledge, I will apply several methods I've developed over my coding learning journey so far. They could be breakdown as follows:

- Take the time to read each text content on Python correctly, and read it back when something isn't clear.
- If something in the course material is still not clear (even after a second read), go online to search for that same topic and see if complementary information / examples are available to help my understanding.
- Always do each task at the best of my capacities – the goal not being to simply get the task accepted, but to understand everything I've done, and why.
- In each task, when something isn't working as expected, always try to troubleshoot the problem by myself first via all the debugging technics I could think of.
    - This not only allow me to develop my debugging skills, it also helps me to remember how to fix that specific problem if I ever encounter it again in the future.
- Compare my Python codes to some other codes I've written in the past (e.g.: in JavaScript) if I think it could be useful to better understand the differences and the functioning of each language respectively.
- Ask my mentor for any questions I have not been able to find the answer by myself.
- Continue learning about Python even after the course end, by reading theory books/articles and by creating other apps by myself.

1.1 - Getting Started 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?

   Answer:

- **Backend**: Backend is related to all the codes and structures that are not seen by the users on their UI, but that are still essential to ensure that all the elements the users can interact with / see on their screen actually work as expected.

  For example, a backend developer could work on setting up the different API endpoints that allow users to perform different operations in the web app (or other product) database. If a user creates an account, he'll be using a form displayed on the frontend. However, as soon as he submits the form, the API endpoint created to handle user creation account in the backend (POST request) will be used, and the database (which is also part of the backend) will be updated accordingly, by registering the new user info.

  On top of endpoints and database, other elements could also be implemented by a backend developer, such as codes to hash users' passwords, a CORS policy to protect the web app, or codes to define database schemas / models.

  In the MERN and MEAN tech stacks, the backend is represented by Node.js, Express and MongoDB for example.

- **Frontend**: Frontend is related to all the codes and structures that are seen by the users on their UI. The frontend is closely related to the backend, as those two work together to ensure the web app (or other product) is fully functional.

  For example, if the backend of a web app has been developed to allow users to create an account and delete it, the frontend will display a user registration form on the users UI, as well as a delete account button. When these elements are used by users, the backend logic is called and perform the required action.

  Important considerations for frontend development usually include easily navigation through the web app, accessibility, great visual, responsiveness, good communication with the backend, and so on.

  In the MERN and MEAN tech stacks, the frontend is represented by React (MERN) and Angular (MEAN) for example.

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?

   Answer: Both languages could be interesting, as both of them share some similarities. For example, both are a high-level scripting language and uses easily understandable keywords to make commands and perform tasks, which is a great plus. Python, as well as JS, uses dynamic typing, an approach that allows variables to assume any kind of value without producing errors, which is another interesting element.

   However, I would suggest to use Python for the following reasons:

   - Python frameworks come with pre-installed common web operations such as URL routing, form handling and validation, template engines, etc. This helps developers to move quickly with application development.
   - Python code is easily readable, which help to understand documentation and performing debugging more easily. Python also comes with its own shell called REPL, which is fairly easy and simple to use, and can be of a great help to test python code syntax and perform debugging quickly.
   - Finally, Python is widely used, so there's a big community of developers around it, a lot of documentation, discussions, GitHub topics and so on. This is very helpful to find different important information over the development process.

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?

   Answer:

   1- By the end of this achievement, I want to have a very good base about core Python principles, in order to be able to discuss on Python with very good level of confidence if someone was to ask me questions such as:

   - What is Python?
   - What are the main characteristics of Python?
   - In which scenarios / projects Python would be the best tool, and why?
   - What are the main differences between Python and JavaScript?

   2- By the end of this achievement, I want to know enough about Python to at least be able to identify what personal projects I could build using Python, and have a good idea of how the different features could be implemented.

3- By the end of this Achievement, I would like to improve the way I developed my documentation. More precisely, I would like to have a professional level README related to my project, to show my Python technical communication skills in the best way possible.