**Getting Started with Python for Data Scientists**

**What you should learn when starting out**



Photo by [Uriel SC](https://unsplash.com/@urielsc26?utm_source=medium&utm_medium=referral) on [Unsplash](https://unsplash.com?utm_source=medium&utm_medium=referral)

A key question for any Data Science beginner is which language to use. There are plenty of options out there and making a choice can be difficult, especially when you see arguments on Twitter (or another reputable source) that one language is better than another. I can be confident in saying that no matter which…

Nevertheless, most people end up starting with Python. The reason for this is because of the deep ecosystem that has developed around Data Science in Python, including libraries such as sklearn, statsmodels, pandas, matplotlib, tensorflow and many more. This means that any workflow that you would expect to come across in a Data Science career, you can probably do in Python. The other benefit of picking Python, however, is that it has wide applicability beyond Data Science as well. This includes being used in web development through frameworks such as Django and Flask and soon to be [Pyscript](https://pyscript.net/) as well, its use in automation and testing due to its ease and simplicity and packages such as Beautiful Soup, and the broad usage in general Software Engineering circles for building products. Thus, learning Python can not only set you up for learning Data Science, but also for a much broader career in Software Engineering as well.

**Python fundamentals**

One of the first steps of learning any coding language is to learn the fundamentals. As part of this you also need to set up your computer so that you can actually write and run code. In Data Science, this is often done through the use of Anaconda and Jupyter Notebooks, a common environment used for Data Data Science workflows. The benefit of this for beginners is that you can run small individual pieces of code clearly and Anaconda can help you navigate the often messy reality of package conflicts in Python. While many move onto using actual Python scripts later on and using virtual environments, Anaconda and Jupyter notebooks are good places to start.

Learning the language itself then often starts with understanding how variables and Data Types work. In the case of Python, and in most languages, variables are used to store information that allows you to call and use that information later on in your program. This is simply done with the = operator in Python, which assigns the information to a variable. The second thing to then learn is what data types the language supports. In the case of Python the four main basic data types include int , float , str and bool which represent an integer (a integer value with no decimal place), a float (a numerical value with a decimal place), a string value (typed words) and a boolean value (which can only take on True and false ). While there are other Data Types you will likely encounter, these are the basic building blocks to getting you started on your journey.

The next thing then is to learn about operators in the language. This is the notation that is used to perform operations such as mathematical or comparative operations. In the former, we use notation such as + for addition, - for subtraction, \* for multiplication and / for division as we would expect. However we can also perform comparison operations, which then form the basic of control flow. In Python this can include comparisons such as == for checking if values are equal, != for not being equal and < , > for less then and greater than respectively.

**[UCL Data Science Society: Python Fundamentals](https://towardsdatascience.com/ucl-data-science-society-python-fundamentals-3fb30ec020fa" \t "_blank)**

**[Workshop 1: Jupyter notebook, variables, data types, and operations](https://towardsdatascience.com/ucl-data-science-society-python-fundamentals-3fb30ec020fa" \t "_blank)**

[towardsdatascience.com](https://towardsdatascience.com/ucl-data-science-society-python-fundamentals-3fb30ec020fa" \t "_blank)

**Python Logic**

The next thing to cover is how logic and process flow works in Python. This is so that you can create more complex programs that have some logic built in such that certain actions are triggered when given conditions are met. In Python, building these complex programs often involves the use of conditional statements, logic statements, loops and functions.

The first thing to cover in this regard is that of conditional statements. While you will have covered comparative operators, this involves how they can be used to check whether a condition is met or not and then triggering some code in response to that. An example of this would be checking whether a variable a is equal to b such that a == b or that a is greater than b such that a > b would respond as True. These comparative operators can then be used to trigger code using conditional statements of if , else , elif . Thse allow you to trigger code if conditions are met, or else what would happen otherwise. These conditions can then be built up into more complex statements through the use of and , or and not which allows you to check more than one condition at a time.

We also need to know how to repeat pieces of code based on conditions or by creating reusable pieces of code. The former can be triggered using loops, which essentially run the same piece of code as long as the condition has been met. This is split into while and for loops which the former performs the given action while a condition is still true, while a for loop will loop over an already defined group. Then we also have functions which are useful when we have code that we need to use over and over again but in different areas of your code. This can be when you want to perform the same action but with different inputs or at a different stage of your workflow and is done by defining a function that can be called later in your code.

**[UCL Data Science Society: Python Logic](https://towardsdatascience.com/ucl-data-science-society-python-logic-3eb847362a97" \t "_blank)**

**[Workshop 3: conditional statements, logic statements, loops and functions](https://towardsdatascience.com/ucl-data-science-society-python-logic-3eb847362a97" \t "_blank)**

[towardsdatascience.com](https://towardsdatascience.com/ucl-data-science-society-python-logic-3eb847362a97" \t "_blank)

**Python Sequences**

Once you have covered the fundamentals and logic of the language, the next step is to then understand how to store different forms of data. This is very important in Data Science as you are unlikely to be storing single pieces of information at a time but rather multiple chunks of data each requiring a specific format. For this, we need to be able to select the correct data format that would allow for the most efficient storage and access possible.

In python, there are four main built in sequences that you would often be taking advantage of. This includes the List, Tuple, Set, and a Dictionary. It is important to learn how to use these and their key characteristics to ensure that you are storing data in the correct way. In this case:

* **Lists**: are mutable, ordered, indexable and can contain duplicate records
* **Tuples**: are immutable, ordered, indexable and can contain duplicate records
* **Sets**: Are mutable, unordered, unindexable and do not allow duplicate records
* **Dictionary**: are mutable, ordered, indexable and cannot contain duplicate values (at least in their keys)

And understanding each of these characteristics will determine which data structure/sequence you will choose to store your data in so that it is easy to access when you want to perform your analysis.

**[UCL Data Science Society: Python sequences](https://towardsdatascience.com/ucl-data-science-society-python-sequences-e3ffa67604a0" \t "_blank)**

**[Workshop 2: Lists, tuples, sets and dictionaries!](https://towardsdatascience.com/ucl-data-science-society-python-sequences-e3ffa67604a0" \t "_blank)**

[towardsdatascience.com](https://towardsdatascience.com/ucl-data-science-society-python-sequences-e3ffa67604a0" \t "_blank)

**Programming Paradigms**

Alongside learning the language, it is also important to understand how different programming paradigms work. In learning most of those above, you will have encountered Procedural and Functional programming paradigms. The former is where the code is laid out in a procedural way whereby the code “proceeds” essentially as it has been written. While the latter often uses Procedural Programming but also takes advantage of abstracting repeatable pieces of code into functions. This reduces the total amount of code that you have to write, and allows for some form of abstraction as well.

The alternative to this, and which you will encounter when delving deeper into libraries in Python, is that of Object Oriented Programming. Contrary to the previous two paradigms, this one structures code so that both characteristics and behaviors of data can be bundled together into a single structure. It does so by creating “blueprints” known as classes that allow you to create objects that can take on certain characteristics and behaviours that are defined earlier in code. Understanding this paradigm is important for being able to interact with many of the libraries that will be a part of any Data Science workflow. The benefit of this paradigm is that it facilitates writing code that can be used repeatedly and bundles both characteristics and behaviours into a single structure, making it easier to use and understand when interacting with libraries.

**[UCL Data Science Society: Introduction to Object Oriented Programming](https://towardsdatascience.com/ucl-data-science-society-object-oriented-programming-d69cb7a7b0be" \t "_blank)**

**[Workshop 4: What is OOP, defining a class, adding attributes, adding methods, and class inheritance in Python](https://towardsdatascience.com/ucl-data-science-society-object-oriented-programming-d69cb7a7b0be" \t "_blank)**

[towardsdatascience.com](https://towardsdatascience.com/ucl-data-science-society-object-oriented-programming-d69cb7a7b0be" \t "_blank)

**Conclusions**

Learning a new coding language can be tough, especially for those learning their first language. Python is beneficial for Data Scientists in this way because of its relative ease in getting started with simple syntax that is easy enough to read and understand. In learning the language for Data Science, it is advised that you cover most of the basics which include: Variables, Data Structures, Sequences, operations, logic, functions, and object-oriented programming. Once you have these fundamentals down, you can then take more confidence in starting your Data Science journey in Python and move onto more complex topics and building your Data Science workflow. Good luck!