SDV602 2021-S2 Journal

Emily Friis

13505646

# Journal Entry One

The first two weeks of SDV602 served as an introduction to Python as a language.

To begin with Python, we first looked over two of the most fundamental principles in programming languages – typing and flow control statements. Compared to C#, where type references are entirely strict, being required in variable and function declaration statements, Python eschews that requirement entirely. That being said, Python is a dynamically typed language, so, while it allows the types of variables to change throughout their lifetime, type is still checked as the code runs with the Python interpreter. Python also features gradual typing – wherein some expressions can be typed strictly without having the same requirement for other expressions. I find that these features make Python more flexible and easier to grok.

Also covered, was the implementation of flow control statements in Python, while not too dissimilar to C#’s implementation, Python’s syntax differs somewhat. Python signifies the inner portion of a flow control statement with a semicolon, rather than enclosing it in curly brackets, to me, this seems more functionally effective and reduces the chance of mistakes leaving unclosed brackets. The signage for conditional statements is also altered, using words (i.e., and, or) rather than symbols (i.e., &&, ||), making code more legible at a quick glance.

These learnings serve as the basis for further understanding of Python, and I look forward to expanding my knowledge of the language going forwards. I hope to in future be able to understand Python’s typing in more specific detail – its limitations and applications.

# Journal Entry Two

Week three of SDV602 focused on the basics of data structures within Python. The types discussed included:

* Lists ( [ , ] ) store a collection of data of many types in separate elements.
* Tuples ( ( , ) ) store a collection of data, with multiple values assigned to each element.
* Sets ( { , } ) store a collection of data that is unordered, unindexed, and cannot have duplicates.
* Dictionaries ( { : , : } ) store a collection of data in pairings with a key and an associated value.
* Ranges ( (n, n) ) store a sequence of numbers based on given arguments.

One of the ways in which these data structures differ in function is mutability. Mutability refers to whether a variable can be altered after being created. This can be important for many reasons – adding, removing, replacing elements without reassignment. Thereby, through mutability, some data structures are more suitable than others for performing certain roles (i.e., tuples are more ideal than lists if you don’t want any elements in the collection to ever be changed).

Additionally, each type of data structure has a number of associated methods (i.e., list.append(element)), which act to further differentiate the applicability of each type to certain functionalities.

# Journal Entry Three

Weeks four and five of SDV602 went into GUI modules in Python, specifically the one that is to be used for the course’s main assessment – PySimpleGUI.

PySimpleGUI is a fairly straightforward GUI framework based on the powerful and popular Tkinter framework, simplifying some elements to make development more intuitive. It focuses on a layout, representing the GUI window, which is filled with elements such as text, buttons, and text fields. Because most of the boilerplate/templating work is done by PySimpleGUI, the code for instantiating a layout is fairly trim.

When a layout is created with PySimpleGUI, it is sustained with an event loop, allowing you to have triggered events on button presses – manipulating the current window, opening or closing other windows, opening or manipulating files, or anything else necessary for the GUI.

These weeks also touched on docstrings in Python. Docstrings are string literals used with almost all functional units in Python – Classes, Modules, Functions, and Methods. Docstrings are useful to give context and a description for a block of code, beyond the scope of the detail a comment might have – instead giving a broad overview. The larger an application gets, the more necessary docstrings become to grasp the overall functionality of code.

# Journal Entry Four

Week six of SDV602 covered the creation and display of figures from csv files using python modules:

* Matplotlib is a plotting utility used to draw figures and embed them into application GUI frameworks like PySimpleGUI.
* NumPy is a module that help to work with arrays of data in Python – being faster than working with Python’s default data structure Lists which, while fulfilling many of the same functions as Arrays, lag behind significantly in speed and resource use.
* Pandas is a module designed to help working with tabular data, taking data from sheets such as CSVs and Excel files, and creating a DataFrame that is more easily parsed through Python.

Using these modules together, it is possible to do a lot with tabular data – creating a number of different kinds of graphs and figures for displaying information in a readable format. Pandas allows for data to be translated into any format necessary in a DataFrame in order to be ideally plotted, and Matplotlib has options for most kinds of graphs – line and dot charts, bar charts, scatter plots, etc. NumPy also acts to smooth over the whole process, making it applicable for use in programs.

Also covered were “key word arguments” or “kwargs” as a means of passing a variable number keyworded arguments to an object in Python. Kwargs makes functions and similar objects in Python incredibly flexible, being able to accept a number of differently formatted inputs without significant risk of exceptions being thrown.

# Journal Entry Five

Week seven of SDV602 introduced the manipulation and handling of exterior files through Python. A lot of the learning was accomplished through a LinkedIn learning course for working with files in Python, which covered a wide range of options for manipulating files in Windows OS.

There are several important modules for working with files in Python:

* Os – Allows interaction with the operating system.
* Glob – Form objects based off file metadata.
* Pathlib – Allowing file paths to be parsed as objects instead of as strings.
* Json – Simplifying the parsing of json files.
* Csv – Simplifying reading data from csv files
* Shutil – Allows direct manipulation of files and directory structures.

Also discussed were the various modes for opening files with Python:

* r mode – Read Only
* a mode – Append Only
* w mode – Write Only
* r+/w+ – Read and Write
* a+ – Read and Append
* r+b – Read in Binary
* w+b – Write in Binary
* x – Create File

Finally, the course also went over reading files and parsing json files to access their contents:

* file.read, file.readline and file.readlines for reading files line by line
* json.load and json\_content[‘attribute’] to read from json files\

Overall, I felt like I gained a lot from this exercise, improving my general knowledge of working with files in Python using all the aforementioned modules and associated methods.

# Journal Entry Six

Weeks eleven and twelve went into using a service called JSNDrop for our project application to store and retrieve data from an online hosted database. The service uses a SQL-like DML to create, store, select, delete, and drop data in tables. Each student was given their own token with which to access a partition of the database for their project. The intention is to use JSNDrop as the medium of remote storage for data as part of the third milestone in the course’s assessment.

Because JSNDrop uses a SQL-like DML, it is ideal for use in the data explorer application’s login center, allowing for easy remote validation of user credentials. It can also be used to store the contents of the CSV files that are being analyzed, as well as any chat messages that are needed to be stored.

Before experimenting with JSNDrop commands, I was entirely unfamiliar to working with Python together with a SQL-like DML, so this was a good experience to have moving forwards with my understanding of Python.