**Backend**

I learnt Python - principles of object-oriented programming and basic functionalities of the language.

* Strings - Python has a powerful built-in class called ‘str’ that allows operations on strings, but the results of these operations need to be stored in new string literals (str variables) since strings are immutable. Operations include concatenation, slicing, getting the length of string, replacing letters or parts of the string, changing case, accessing characters at specific indices, checking whether a character is part of the string etc.
* Lists - This is a built-in list type. List literals are written within square brackets []. Methods (operations) on list objects include appending elements, inserting elements at given indices, removing elements, extending list, slicing, searching for a element, sorting, reversing etc.
* Tuples - This is a grouping of elements of fixed size (such as an (x, y) coordinate). Tuples are like lists, except they are immutable and do not change size (though not strictly so since one of the contained elements could be mutable)
* Dicts - The contents of a dict can be written as a series of key:value pairs within braces {}, e.g. dict = {key1:value1, key2:value2, ... }. This is an efficient key/value hash table structure where looking up values or setting them requires the use of square brackets. The del operator can be used to delete entries.
* Files - Python opens file handles that can be used to read, write or append to files.
* Exceptions - An exception represents a run-time error that halts the normal execution at a line and transfers control to error handling code. A "try/except" structure can be used to handle these.
* FOR loops - Used to iterate commands/statements for a fixed number of iterations, also to access or operate on list, dict and tuple elements.
* WHILE loops - Used when operations or statements need to continue till a condition is fulfilled
* IF statements - Used to check conditions and use statement blocks as needed.
* OOP principles:

1. Encapsulation - mechanism of hiding of data implementation by restricting access to public methods. Each object keeps its state private, inside a class and other objects do not have direct access to this state. Instead, they can only call a list of public functions — called methods. Thus, the binding between the private state and public methods is made.
2. Abstraction - When this principle is applied, each object exposes only a high-level mechanism for using it - internal implementation details are hidden and it reveals only operations relevant for the other objects.
3. Inheritance - the creation of a (child) class by deriving from another (parent) class, forming a hierarchy. The child class reuses all fields and methods of the parent class (common part) and can implement its own (unique part).
4. Polymorphism - means “many shapes” in Greek. Polymorphism gives a way to use a class exactly like its parent so there’s no confusion with mixing types but each child class keeps its own methods as they are. This is valid for classes and functions unrelated to inheritance as well, such as when polymorphism is used through overloading (by changing number, order and types of parameters)

**Frontend**

I did more of frontend than backend, mainly HTML, CSS and a small section of JavaScript.

HTML

I mainly learnt tags for formatting (heading tags and other normal text formatting) and structuring of webpages (through body tags such as <div>, <p>, <span>). More than HTML, I used external and inline CSS through the HTML webpage in the 2 assignments that we completed.

CSS

* Selectors - element, ID and class selectors (as well as attribute selectors) - these are used to select elements that need to be styled
* Box Model - Consisting of Content (where text and images appear), Padding (clears an area around the content and is transparent), Border (border that goes around the padding and content), Margin (clears an area outside the border, is transparent)
* NavBars - various functionalities such as drop-down menus, slide-out navigation bars as well as ones that are fixed to a certain alignment (at ay one side of the page)
* Forms
* Image properties and formatting, including background images and positioning
* Flexbox - flexible responsive layout structure without using float or positioning using flex attributes and a flex container
* Media Queries - used to change attributes, positioning and layout of the webpage to suit different sizes of screens/ viewports.
* Grid - offers a grid-based layout system, with rows and columns, making it easier to design web pages without having to use floats and positioning

JavaScript

I learnt basic element and id functions (mainly used for buttons and other simple usages) along with fundamentals such as operators, variables, strings, arrays, Dates, Events.