Learning Journal

Exercise 1.1:

1) Frontend development is the design and implementation of the parts of an application, program etc. that an end user would interact with. Simply put its all of the things that a user will see and use, be it graphics, text, pictures etc., backend however is the unseen by an end user structure of any running program. Normally this will consist of things like server management endpoint usage and routing among other things. I like to think of backend as all of the code that runs when a user presses a button, and the frontend is the code for the button itself (although it can apply to much more than buttons).

2) For me the difference between python and javascript has always lied primarily in the usability / readability of everything. Python in a lot of ways has always felt like the way I would explain how a coding language works ie: c++ or javascript. I think this mainly comes from the simplicity of the syntax as it feels much more similar to an actual spoken language than most other programming languages ever will. If I were to convince a team to use python, my best argument would be based on that same feeling as I feel it allows you to make code much more easier to understand, and you might think as programmers they should understand regardless, but as someone who has seen blocks of code span multiple monitors it all starts to jumble very fast. Specifically the variable access and management allows you to quickly alter and reassign values of the same name in a way java simply does not.

3) Goal 1 for me would be to become MUCH more comfortable with CLI. I have a large amount of experience coding in a "standard" environment, managed by editors and other applications, but I have always felt CLI practices never came natural to me. Goal 2 would be to change my point of view on python as personally I have always felt python was a bit almost discombobulated for me. I never truly felt at home coding in it, and at multiple points throughout

college I felt it was different for the sake of being different with no actual improvement in mind. Obviously I know there are benefits to python, I just haven't quite yet found what those are for me. Goal 3 would be to familiarize myself with python libraries. When I selected this class as a specialization I did some research into python and found that there are vast amount of libraries used that make python much more applicable to a career

Exercise 1.2)

1). I think the main benefit for me of the Ipython shell is just the ease of use. I have always disliked writing with CLI just solely based on the horrible readability of it. All outputs being essentially the same color and spacing as inputs has always bothered me, and iPython is much more similar to a normal editor

2) String : array of chars : non-scalar

Int : integer : scalar

Bool : true or false value: scalar

Float : number decimal points : scalar

3) A list is a mutable tuple, meaning the elements can be altered

4) I would do the same thing I did in my cookbook, with the individual words being stored as dicts, since there are multiple values each one needs to hold, and then using a list as a catalog of all our dicts, allowing us to change the order and add or remove if necessary.

Exercise 1.3)

1)

Destination = str(input(“Enter your desired Location))

Place1 = “cabo”

Place2 = “prague’

Place3= “geneva”

If destination == place1:

print( “Enjoy your stay in Cabo”)

Elif destination == place2:

print( “Enjoy your stay in prague” )

Elif destination == place3:

print( “Enjoy your stay in geneva”)

Else:

print(“Oops, that destination is not currently available)

2) Logical operators give us the option to perform multiples of checks at the same time. Whether that is checking one variable for multiple things, or checking multiple variables for one, with them we have the option to combine what could be a multitude of lines all into one.

3) Functions allow us to create our own essentially baby script within a file. It gives us the opportunity to run the same processes based on our own parameters and gives us the ability to use them repeatedly without having to reuse the same code over and over again while keeping things very consistent.

4) I feel as if I have become a bit more familiar with CLI mainly just navigating, however with everything im doing being “new” in python it has still been a bit unwieldy. I still feel mostly as if python is more awkward, the learning we have done so far is very basic and I knew most of it previously so my view hasn’t been shifted much. At this point we have not really used any libraries so no progress there.

1.4)

1. If you weren't able to store files, all of the work done would be essentially pointless as none of the data would remain once the terminal is closed
2. A tool that converts data types commonly used in python into readable binary data. You would use them at any point where you are migrating data out of one file and need to perform operations on that same data in a different file.
3. os.getcwd() os.chdir()
4. Using error handling block like try except, then to prevent it from breaking you would designate a resolution for that error, for example in this task we assigned the data an empty list when an error occurred so that the program will not end when there is no data
5. I feel like I am in good spot with what we have covered so far. Most of it has been more of a refresher but some of it has been new.

1.5)

1. A programming concept based around creating your own essential building blocks of code usually classes to provide for functionable readable code that is not repetitive.
2. A class is essentially the tutorial on how to make an object, and an object is the “made” version of that class. For a real world example, (ironically) think that a cake recipe is the “class” and the cake you make with it is the object
3. Inheritance: the ability for different objects/classes to pass over or share the same properties. For example, if I were to make a class called “cars” i could then make further classes called “coupe” “suv” etc, and then I would pass all the cars properties to the subclasses since they would all have the same base characteristics

Polymorphism: when functions or variables from different classes or objects have the same name, using the car example from earlier, I could have a class called cars, and one called planes, and a function for both called getNumberOfWheels() that would be the exact same on both for continuity.

Operator overloading: taking the default python operators which are built to work with base python variables and adapting them to work with custom python objects you’ve created.

1.6)

1. Databases are essentially a hub where information can be stored, they will store and handout information of multiple data types when accessed properly, as well as providing the opportunity to be changed and updated at any point.
2. INT: an integer (same as other language) Float: a number with a possible decimal value (same as other languages) VARCHAR: similar to a string as it is an amount of text however it is predetermined.
3. SQLite is much better when it comes to smaller limited projects that are not created with the intention of large scale implementation
4. For me the difference between python and java (honestly most languages) have been growing on my slightly. I think back to my first COSC course in school and remember being annoyed with having to come up with names for different variables in each function and then having to remember them over and over again. I still struggle with the difference in syntax but I am doing much better and am not as adverse to it as I once was.
5. Python feels limited to me in its simplicity. While I feel it is a strong language for someone to learn in, I feel like its interaction and timing could very much run into snags on larger much more involved uses.

1.7)

1. A tool that will convert database interaction into a more smooth and functional set of actions that will fit together with the languages you are writing in. They are very useful for keeping things much more linear, while adding the advantage being able to remove alot of clunky hardcode database code.
2. I think overall it went fine, I did have some issues with database interaction and indexing, so I guess if i could do it differently I would structure the tables in a way that felt more natural to me for example, the way the ingredients are held got a bit awkward.
3. I would say I have python experience in basic interaction, core structures, data types, OOP, sql integration both natively, and with object mapping.
4. Most of the achievement I would say went well, I felt much more comfortable writing in python than I ever have before which I guess is the goal.
5. Im proud of the fact that for the most part I didn’t ever have to reference other material besides 1.7
6. It did in some ways but not in others, while I feel confident in my python writing now, I feel like the python adaptation is novice at best and I don’t necessarily see the workforce application of my python knowledge yet I guess.
7. I would say its best to keep in mind that end of the day these are “languages” and you wouldn’t be able to learn mandarin overnight so don’t be too hard on yourself if you struggle with python or any other programming language.

2.1)

1. Using python on its own would give you all the normal benefits of python and could possibly help you keep things simple and concise depending on the circumstance, however with django you might run into issues having much more options than actually needed, but with django it gives you the opportunity to manage things on a much more consistent scale as data and total interactions scale.
2. I think it comes down to simplicity, when you can take something that gets more and more complex each day like database interactions, the ability to have that all handled for you is a tremendous bonus
3. Django goals:
   1. The most applicable ways django is used and interacted with in todays society
   2. A feeling that at the end I am “advanced” in django
   3. Hopefully something django or python related

2.2)

1. I have chosen google as my website. Google could use a different project for each of the sites (mail, docs, youtube etc.) while also creating a login project that handles the necessary login information which will not be unique to any of the specific projects. They can also use databases to store a multitude of things within each user. For example, a user will have a database entry for emails, with subcategories for all email types, as well as an entry for uploaded videos, documents in google docs etc.
2. Deploying a django app is a relatively simple process, all you need to do is create a virtual environment of your naming, then install django into that environment. Once all that is done you can activate the environment and django then run the designated commands to have django create the project for you.
3. I think the biggest use case of the django admin site will be a troubleshooting tool. My assumption is that at one point we will be adding, indexing, and editing users to the application, and having the ability to list and edit the users from the admin portal will be a big help.