**1. High-Level Design and Architecture**

When it comes from high-level to large python projects my usual approach is incrementally building up the project towards the final requirement with an agile and flexible mindset. For example, in my UNO card AI sentry turret project the main goal was “Make a turret that points to UNO cards using some pose estimation model”. Before that my first step was “Getting an rpi to control the turret servos”, then “Getting the AI model to work reliably”, then “Using coordinates to work out the aiming angles”, etc… until I get to the final solution.

In terms of architecture I never written my own libraries (aside from local importing) and I usually stick to commenting code in my main script and breaking down as much of the functionality into methods so that the code is as maintainable as possible. So far my larger projects consisted of at most two scripts where one is an “utility” script with some pre-written functions I might’ve needed and the other one is the main script that does all the work.

I never planned the high level architecture at the beginning with python but I have used draw.io for making flowcharts to explain what my code does and for some Unity game projects I did use flowcharts at the beginning to better define the high-level architecture of the project. An excellent example of this was “Crusty’s flight sim” where the airplane physics of the plane was broken into modular scripts such as “Airplane\_Input”, “Airplane\_Controller”, “Airplane\_Characteristics”… where is fulfils a given role. This greatly improved maintainability and allowed me to create a modular system for other planes.

**2. Creating and Structuring APIs**

I don’t have any experience implementing python APIs. I do recall trying to make a “Youtube Sub Display” or Instagram “Follower Display” but I faced a ton of issues with getting the APIs to work at all. Nonetheless I am still optimistic about using APIs and implementing external code since I have plenty of experience troubleshooting code when implementing new libraries and reading through the documentation to get a better idea on how a new library works.

I do not have any experience making an API in python.

**3. UDP Messaging**

I once tried to use python’s “socket” library in the UNO card AI turret library. I managed to get a working connection between my laptop and a raspberry pi, although I kept running into an error which interrupted the connection after 1 single message hence I dropped using that method. The socket library seems to allow for either TCP or UDP. That’s as far as my knowledge with python UDP messaging although I am still willing to further look into it and get a properly running application.

**4. Integrating Existing APIs or Third-Party Libraries**

I do have experience sending data between an rpi and Arduino via python code on the rpi and c++ code on the Arduino. The specific software I used was ROS which is trickier to implement compared to simple serial messaging. I have implemented python serial code for my laptop to send data to an rpi too. Therefore I do not see much issue getting sensor data from an Arduino or sending data to it to control LEDs.

**5. Python GUI Development**

My experience with python GUI development just goes as far as using tkinter for simple menus and tabs in past projects. I am just as comfortable troubleshooting this GUI code as the other code

**6. NumPy for Linear Algebra and Quaternions**

I have a solid understanding of linear algebra and Quaternions from my course and Unity Game development respectively. I have worked with rotations and transformations with Peter Corke’s robotics toolbox library as well as Unity’s game various tools for handling rotation of GameObjects. I am yet to work further with rotations and transformations in python with Numpy.

**7. Real-Time Sensor and Actuator Development**

I haven’t developed any python applications that interact with sensors and actuators in real time. I do have better (if not all) experience in doing so with Arduino C++.

**8. Version Control and GitHub Workflow**

I have plenty of git version control experience from university projects. I can upload and maintain a project repo over time and even keep a few branches of it. I’ve had some merging issues in the past when trying to make a “final branch” from two separate branches,, hence I may need to further explore that area.

**9. Reflecting on Improvement Areas**

Out of all these areas I feel most to least confident:

-Version Control and GitHub Workflow

-High-Level Design and Architecture

-Python GUI Development

-UDP Messaging

-NumPy for Linear Algebra and Quaternions

-Real-Time and Actuator and Development

-Integrating Existing APIs or Third-Party libraries

-Creating and Structuring APIs

**Learning goals based on my honest answers:**

Based on my honest answers and my current goals my main priorities will be:

-Designing the High-Level outline of my software: What will it do? Logic flowcharts? Requirements? I should also aim to break down the python code into as many simpler modules as I can to later make the software process far easier and less prone to getting stuck behind errors. The design should prioritize modularity as much as possible. A modular design may also be much more useful later on to adapt the software for other sims.

-Reading through the XPPython API Documentation and begin tailoring it to the high-level outline I made

-Going back to the UDP project and working towards stablishing working connections between the python client and other clients via UDP

-Get a simple python GUI working before I can move onto the more complex Unity version