For the final project in the cryptocurrency elective, I created a program that mimics the way that smart contracts can be run on the Ethereum blockchain. To give a brief summary of how it works on Ethereum, ethereum can be used to run smart contracts, which are pieces of software that can be triggered by transactions. These are often used for security, or for transparency. During the course, we created our own blockchains, but didn't implement this feature, so for my final project I thought it would be a nice idea to try and make smart contracts work in Python. Throughout the project, there were many challenges. Firstly, Ethereum contracts are paid per operation, such as addition or checking a value. To find these operations, you need to disassemble the code. Python does have a library for this, but it is not very deep or recursive, meaning it is very shallow. This mostly comes from Python not being a language built around basic assembly instructions. This meant that I couldn't do static analysis, analyzing a program by looking at the code, which I thought possible at first: the analysis would have to be during the runtime. For this, I would need either a profiler, which could track the energy consumption per operation, and replace the disassembly, or I could build my own wrapper. Unfortunately, Python does not have any lightweight profilers, since they are very clunky and require low-level analysis. In the end, I just made a wrapper which runs the program line by line, disassembling each one first to calculate the cost. The program has a few functions, the most important of which are the wrappers, with the second most important being the one which calculates the cost of each line.