Chris Hipple's Reflections on Project 1

What I completed

This project submission contains a working game for managing a factory. The goal of the game is improve your production process until you make an income of $3,000 in one day. At the start of each day, customers will put in a random number of orders, and each order is for several of a specific type of product. Both of these quantities can be improved by upgrading your marketing. The actual production process simulates a production line very accurately. Each station can only be processing one component at a time, and products must flow linearly from station to station. If the next station is still busy, the product will sit idle on that station, creating a backlog. Each day, the products that were produced are sold for income, and the ones that you were not able to build get forgotten. Customers are impatient and upset when you are unable to deliver same-day shipping.

Major Challenges Faced

The biggest challenge I faced was writing the interfaces for the objects to run the production simulation. The idea that each product is its own special unit, and the stations work on one at at time, then hand it off to the next one was difficult to implement. The stations on the line need to understand their own state as well as the state of the next station. While not a subclass of the production line class, the stations are contained in a production line. This requires that the stations contain a reference to the production line they belong to, and that the production line contains a reference to all of its stations. The factory class and the line class also share this bidirectional relationship, where each must know information about the other's state. This allows the stations to report their status to the line, and to get information about the next station through asking the line. This is mostly used for checking whether or not a product can be handled off from one station to the next and for data transfer between the line and the factory, for daily reporting and income calculation.

Another challenge was I wrote an algorithm for sequencing the orders in the most efficient manner, but that did not make it into the final version of the code. Because the different types of products have different run-times on each station, the order in which you produce them can have a large impact on your overall efficiency. Unfortunately, I had to cut it to make room for the game aspects. If you pull the repo and reset to this commit, you can test that feature.

How to use this project

To play the game, simply call the python script in your console / preferred interface. No additional arguments are required, and there are no requirements outside of the base python 3.5 library. The flow of the game is simple. The only user interaction required is at the end of each day, you must choose how to spend your money. There are various upgrades you can make to improve your income, such as station efficiency, your marketing, your cost effectiveness, or increase your working hours. These can be brought up by typing “help” during the upgrade phase of the game. Improving station efficiency reduces the processing time for all products on that station. Improving marketing increases the number of customer orders you get, and also increases the size of those orders. Upgrading cost effectiveness reduces the variable costs associated with producing products, meaning each product you build earns you more money. Buying overtime is a one time investment to increase the length of your factory's working day. Each of these upgrades will marginally improve your income, but together will have a large impact.