Individual assignment – Trains

# Background

Two CSV-files are given which are to be construed into a simulation of trains moving through a fictive train network with a given probability.

# Method

## Structuring data

At first, I had to reform the data in the CSV-files into a way that I could easily manipulate to get the result. Most of the time was spend on how I should structure the program, rather than solving the different problems. I try to give a short visual overview in figure 1 of the thought process.

## Walkthrough

1a def main(). First, the program asks the user for the two CSV-files (train network and probability) and the number of trains to simulate.

1b def main(). The program sees if the files exist and if the number of trains given by the user is only numbers.

2a class Initializing\_train\_simulator. To more easily use the data provided by the user, I restructured it. All the restructuring of data is put under one class. *Note: It´s unclear in the instructions if the delay occurs traveling to or from the station, or if the probability is additive. I assume it´s whilst traveling from the station.*

* Probability:
  + Returns a dictionary of probability:
    - {"A":0.01, "B":0.2}
* Train stations:
  + Returns a dictionary of train stations:

***Figure 1***

* + - {"green": ["A", "B"]}
  + The code discriminates between South and North in the CSV-file, e.g: train\_stations = {"green": ["A", "B"]}, then train\_stations[“green”][0] is the most Northern station whilst train\_stations[“green”][-1] is the most Southern station. This is true for no matter how many stations.
* Trains:
  + Returns a dictionary of trains, e.g: {1: ['blue', 'C', 'S']}

Key == train number, Value [0] == Line, Value [1] == Current station, Value [2] == Direction of travel.

2b class Initializing\_train\_simulator. The program checks if the CSV-files are correct lengthwise.

3 class simulation.

Testing

# Discussion

## Angle of attack

## Checking for input

For a small program, it´s incredible how much can go wrong, and the amount of different ways input can be “wrongly” put in by the user. It´s hard for me to know if I´ve been too loose with the checks of inputs

## Optimization versus readability

The focus of the code has been readability and not optimization of run-time of different blocks.