Very Structured Plan for a Problem

Mingye Wang

# Data Structures

Pick a numeric type for currency. Floats are acceptable until you get to 2^52 / 100 pounds, so we will use that for now.

Define a global variable total\_money\_received (currency).

Define a “class Passenger”. This class describes the identity of a ticket-holder, namely their name (str), age (int), address (str), phone number (int), money\_paid (currency).

Define a “class Ticket”. This class describes a particular transaction, and consists the members holder (Passenger), price (any appropriate numeric type), start & end stations (int; see “Train”).

Define a “class Carriage”. This class describes a passenger car, specifically:

* The dimensions: number of rows and seats per row.
* The seating assignment: a 2d list mapping from the seat to a list of tickets.
* Self.\_final\_stop: The numeric ID for the final stop, to make the end of the interval constrained.

Define a “class Train”. This class describes the whole train, in our case:

* The number or anything to identify the… schedule??
* A mapping “sched” from stops on the route to scheduled time. Make it sorted by time.
* A mapping “stops” from stops to a numeric ID for ease of processing. Make it the same as {enumerate(self.sched.keys())}.
* A list of passenger cars defined above.

# Class Methods

For Carriage, define:

* free\_interval\_for\_seat(self, (row, seat)). This method reads the ticking information for the seat given and returns the intervals for which it is available. The intervals can be represented as a set() of numbers.
* Seats\_for\_interval(self, interval). This method returns a list of seat indices given the intervals requested. Assuming the intervals are represented as sets, you would only need to check whether the requested interval is a subset of the available interval for each seat. At this point we have figured out how to find and count free seats on each Carriage.

For Train, define:

* Find\_seats(self, interval). Basically the Carriage method, but applied to each Carriage we have.
* Sell(self, person, price, (carriage, row, seat)). This method creates a new Ticket around the person, assigns it to the seat indexed, and increments the global and personal price counters. Before all that, it should call free\_interval\_for\_seat() to confirm in case of a race condition.

This should be the bare minimal elements of a ticketing system.