Programming and Data Structures with Python 2025

Assignment 4

24 Oct 2025, due ~~31 Oct~~ 3 Nov 2025

Define a Python class `BusBooking` that stores booking information for a single bus.

Context

- The bus has 40 seats of which 20 are window seats, numbered `W1` to `W20`, and 20 are aisle seats, numbered `A1` to `A20`.
- Passengers can book or cancel seats.
  - At the time of booking, each passenger is assigned a unique booking ID.
  - At the time of booking, a passenger request a window or aisle seat. If such a seat is available, it is assigned to them. Otherwise, any available seat is assigned to them.
  - After all seats are filled, fresh requests go into a waiting list.
  - Cancelling a booking with an assigned seat moves the first passenger from the waiting list into the vacated seat.
- Note that the number of bookings and cancellations can be arbitrarily large. For instance, there may be 1050 bookings and 1020 cancellations, interleaved in some arbitrary order, resulting in 30 seats being filled and 10 being empty.

Interface specification You have to implement the class `BusBooking` with the following functions.

- A constructor with no arguments that creates an empty bus.
- `book(name,preference)`
  - `name` is a string, the passenger name
  - `preference` is a string, the seat preference
    - `preference` is an optional parameter, the default is no preference
    - `W`, `Window`, `w`, or `window` indicates a preference for a window seat
    - `A`, `Aisle`, `a`, or `aisle` indicates a preference for an aisle seat
    - Anything else is interpreted as no preference
  - If there is an empty seat matching the preference, allocate one such seat, otherwise, allocate any available empty seat. If there are no empty seats, add this booking to the waiting list.
  - The function should return a pair of strings `(booking_id,outcome)` where
    - `booking_id` is the booking ID, that is unique across all customers
    - If a seat was allocated, `outcome` is the seat number, one of `W1`,..., `W20`, `A1`, ..., `A20`.
    - If no seat is available, `outcome` is `WL-n`, where `n` is the position in the waiting list. Position `1` is the first position in the waiting list.
- `cancel(booking_id)`
  - If `booking_id` is currently assigned a seat, cancel this allocation. If the waiting list is not empty, allocate the newly vacated seat to the first booking ID in the waiting list, remove it from the waiting list and move up all later bookings in the waiting list.
  - If `booking_id` is in the waiting list, remove it and move up all later bookings in the waiting list.
  - Return `True` if either of the two cases above holds. Return `False` otherwise.

- `status(booking_id)`
  - If this is a currently valid booking ID, return a pair `(name,current_status)` where `name` is the customer name and `current_status` is the seat number or waiting list number (use the same format as `outcome` in the return value of `book()` above).
- `__str__()` should create a list of triples `(booking_id,name,current_status)`, sorted in ascending order of `booking_id` and return the string representation of this list.

Instructions

- Submit your solution through Moodle as a single Python notebook
- Add documentation to explain at a high level what your code is doing

---