Computational Problem Solving CSCI-603
AiRIT Lab 6

10/9/2023

1 Introduction

The powers that be at RIT have decided to start a new airline dedicated to getting students to their Spring Break destination free of charge. The airline will of course be named "AiRIT."

From AiRIT's perspective, each student passenger comprises three important pieces of information:

1. **Full Name** - A first name and last name.
2. **Ticket Number** - A potentially variable length string comprising digits and possibly letters. The first character is always a digit and indicates the passenger's *boarding zone*, which will be a number between 1 and 4.
3. **Carry On** - Indicates whether or not the passenger has a carry-on bag that will need to be stowed in the overhead compartment on the plane.

AiRIT operates a single gate at the Greater Rochester International Airport. There is a separate line at the gate for each boarding zone. As passengers arrive at the gate, they take the next position at the back of the line that corresponds with their boarding zone. Local fire codes insist that the gate only allow a set maximum number of passengers to line up at any one time. Once the gate is full, any remaining passengers must wait outside of the gate until *all* of the passengers already in line have boarded a plane.

AiRIT's boarding policy is similar to that of other airlines. When the aircraft is ready for boarding, passengers will board based on their *boarding zone*, with the highest numbered boarding zone boarding *first*. That is to say that passengers assigned to boarding zone 4 will be the first to board the plane, followed by boarding zone 3, and so on, until all passengers have boarded or the plane is full. Passengers in each zone will board the plane in the order that they arrived at the gate and will load the plane from *back to front*, meaning that, as passengers board the plane, they will move to the unoccupied seat that is closest to the back of the plane. It is assumed that the passengers with carry-on luggage will stow their bags in an available space in the overhead compartments as they make their way to their seats. If the gate empties before the aircraft is full, the aircraft will take off with whatever passengers are on board *before* the next set of passengers lines up.

AiRIT's policy for deplaning is somewhat unique. Once the plane arrives at its destination, passengers *without* carry-on luggage deplane from *front to back*. This means that any passengers in boarding zone 1 should depart first, followed by passengers in boarding

zone 2, and so on. Once all of the passengers without carry-on luggage have disembarked, the remaining passengers will begin deplaning again from *front to back*.

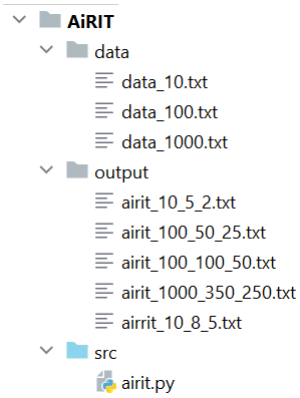
The AiRIT aircraft includes a maximum passenger capacity, but AiRIT runs the plane continuously, 24 hours a day, 7 days a week. As the plane fills up with passengers, it departs, drops its passengers off at their destination, and immediately returns to the gate ready for the next set of passengers. This continues until all student passengers have been safely sent along to their destinations.

1.1 Starter Code

Download the data and example files from here :

`https://www.cs.rit.edu/~csci603/Labs/06-AiRIT/code.zip`

Extract the zip file, and then copy the `data`, `output`, and `src` directories into the root of your project folder in PyCharm. To make PyCharm aware the `src` directory has source code, you should right click on the `src` directory and select **Mark directory as... Sources root**. The `src` directory should be blue color.



Here is a brief description of what have been provided:

1. The `output` folder contains sample runs. The numbers on the file names (e.g., 10 8 5) are the values of the arguments used when running the program. For example, 10 is the suffix of the data file (e.g. data_10.txt), 8 is the gate's capacity and 5 is the aircraft's capacity.
2. The `data` folder contains three files that contain passenger data as comma-separated values: `data_10.txt`, `data_100.txt`, and `data_1000.txt`.

2 Implementation

2.1 Main Program

The main program should be named `airit`. This program will simulate the AirRIT's boarding and disembarking policies. The file of passengers is provided to it on the com-

mand line. The format will be identical to the files provided. If it is not present, you should print the usage statement and exit:

```
Usage: python3 airit.py {filename}
```

If the file name is provided, but does not exist, you should print the following message and exit:

```
File not found: {filename}
```

If the file exists, its contents are guaranteed to all be valid.

Once it runs, the program will prompt for the following:

1. The fire-code mandated maximum number of passengers allowed at an airline gate. This number must be a positive integer.
2. The maximum passenger capacity of an AiRIT aircraft. This number must be a positive integer.

All user input must be error-checked (correct type and value), displaying a descriptive error message when needed. The user is given an unlimited number of chances to enter a valid value.

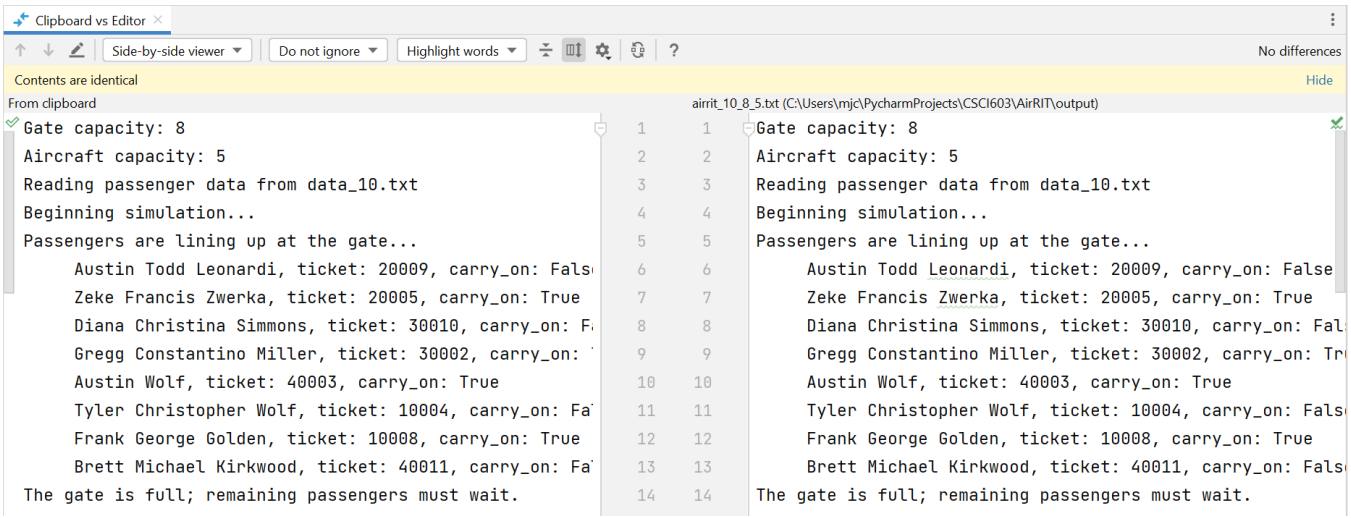
Once all the required information is provided correctly, execute a `run_simulation` method that repeats the following until every passenger has arrived at their destination:

1. Adds passengers to the boarding zone lines at the gate until the gate reaches maximum capacity.
2. Repeats the following until all of the passengers at the gate have boarded a plane:
   (a) Loads passengers on the plane until the plane is full. The passengers should be printed in the order that they are loaded onto the aircraft.
   (b) Unloads passengers from the plane until the plane is empty. The passengers should be printed in the order that they are unloaded from the aircraft. Remember that passengers without carry-on luggage disembark before those passengers that do have luggage.

2.2 Comparing Output

The supplied outputs have a lot of text and can be very confusing to compare your output to. PyCharm provides a utility so you can compare a text file to the clipboard in a visual manner.

1. Open one of the output files in the PyCharm editor window.
2. Run the program with your desired command-line arguments.
3. Select all the text in the console and copy it to the clipboard.
4. In the solution output editor window right click and select **Compare with Clipboard**.
5. A new editor window will open highlighting the differences if there is any.



The ultimate goal here would be to make sure the output matches exactly.

2.3 Constraints

You may use the `Queue` and `Stack` data structures provided to you in lecture. You may not use any other Python data structures or advanced Python features not discussed in lecture.

3 Grading

Your grade will be determined as follows:

- 20%: results of the problem-solving
- 20% Design: The solution is object-oriented and breaks the problem down between various classes and methods.
- 50%: Functionality
  - 5%: Error handling (e.g. file not found, valid maximum passenger capacity)
  - 5%: File handling, reading passengers in from a file
  - 15%: Lining up passengers at the gate
  - 15%: Loading/unloading passengers to/from the aircraft
  - 10%: main simulation loop
- 10%: Code Style and Documentation

Good code design is a significant-graded component. Your program should not be just a "monolitic" main function. The main program should be very small and pass control into classes/methods. You should have things like a method to read the passenger file in, a method for lining up the passengers at the gate, a method to run the main simulation, etc. You should also have separate classes to represent a gate, an aircraft, a passeger, etc.

4 Submission

Create a ZIP file named `lab6.zip` that contains all your source code. Submit the ZIP file to the MyCourses assignment before the due date (if you submit another format, such as 7-Zip, WinRAR, or Tar, you will not receive credit for this lab).