# Passenger Clearance System

Yuhan Xu

**Summary**

In this part of the project, I will use dynamic programming algorithms to allocate them to the corresponding flights according to the needs of travelers in New York. If some passengers want to go to San Francisco, San Francisco is a U.S. city, then it should be assigned to the United States in the United States On flights, domestic flights have separated from various regions, and San Francisco is in western United States, so he should allocate him to flights flying to western United States through the Dynamic Programming algorithm. If a passenger wants to go to Hong Kong, Hong Kong is not a U.S. city, it should be assigned to international flights, and international flights have separated from various regions, and Hong Kong is in Asia, so it should be assigned him to fly to Asian flight (international)

**Input Example**

passenger\_requirement.csv:

1, Asia

2, Oceania

3, Europe

4, Asia

5, Asia

6, United States West

7, United States East

Flight.csv：

1, Los Angeles, United States West, US

2, New York, United States East, US

3, Chicago, United States Mid, US

4, Hong Kong, Asia, China

5, Sydney, Oceania, Australia

6, London, Europe, UK

**Output Example**

Passenger 1 allocated to flight 4,

Passenger 4 allocated to flight 4,

Passenger 5 allocated to flight 4,

Passenger 2 allocated to flight 5,

Passenger 3 allocated to flight 6,

Passenger 6 allocated to flight 1,

Passenger 7 allocated to flight 2,

**Business Requirement**

The first packages need to be included, the areas where each flight goes, if the area is in the United States, marked as domestic such as the eastern United States, domestic and the western United States, domestic etc. or marked as international like Asia,international etc.

The first package needs to be included. The destination cities and destination cities area where each passenger wants to go.

The output result of the final system should include which passenger allocated to which flight .

**Output after Code running**

Insert the path of Flight.csv:

Insert the path of passenger\_requirement.csv:

Passenger 1 allocated to flight 4,

Passenger 4 allocated to flight 4,

Passenger 5 allocated to flight 4,

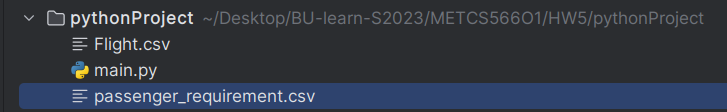
Passenger 2 allocated to flight 5,

Passenger 3 allocated to flight 6,

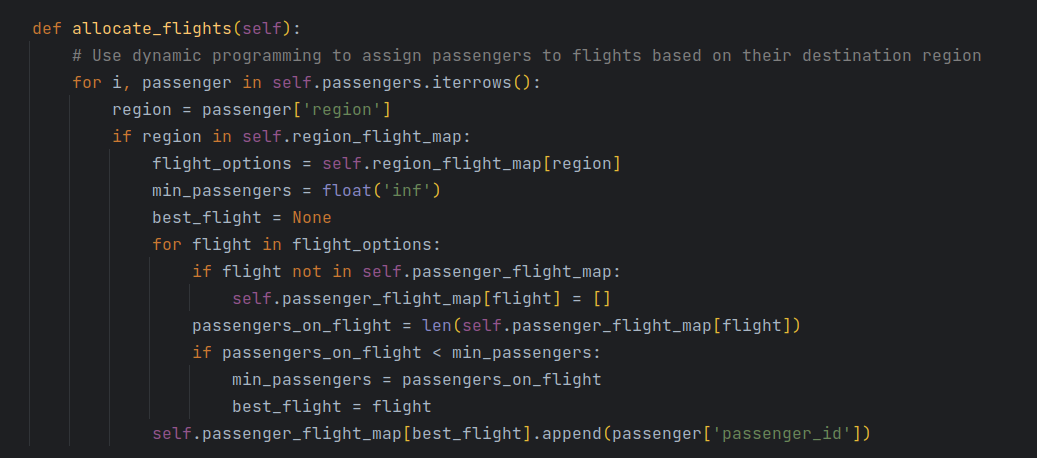
Passenger 6 allocated to flight 1,

Passenger 7 allocated to flight 2,

**Screenshot of the file structure**

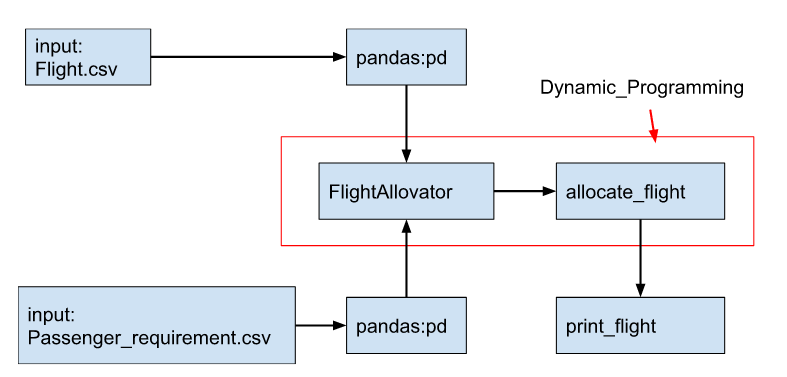


**Code for Dynamic Programming**



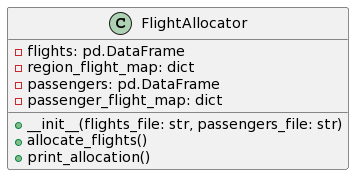
Because the needs of each passenger are different, the system needs to allocate the flights they ride according to the needs of these passengers. For example, if passengers need to go to Australia, Australia is not the United States, so it is an international flight. It can be assigned to the flight of the former Oceania.

**Sequence Model**

****

Sequence model

**Class Model**



class model

**Time Efficiency**

The time complexity of the code depends on the size of the input data. Here is the breakdown of time complexity for each part of the code:

1. Reading CSV files using pandas read\_csv() function: The time complexity of this step is O(N), where N is the number of rows in the CSV files.

2.Creating a dictionary to map regions to flight IDs: The time complexity of this step is O(N log N), where N is the number of flights. This is because the groupby() function in pandas has a time complexity of O(N log N).

3.Creating a dictionary to store the mapping of passengers to flights: The time complexity of this step is O(1), as it involves only creating an empty dictionary.

1. Allocating flights to passengers using dynamic programming: The time complexity of this step is O(N^2), where N is the number of passengers. This is because for each passenger, we need to iterate through all the flight options to find the best flight. In the worst case scenario, where each passenger has a unique destination region, the number of iterations through the flight options will be equal to the sum of the first N integers, which is O(N^2).
2. Priting the allocation result:The time complexity of this step is O(N), where N is the number of passengers, as we need to iterate through each passenger in the passenger\_flight\_map dictionary to output the result.

Therefore, the overall time complexity of the code is O(N^2) for allocating flights to passengers and O(N log N) for reading and processing CSV files. The dominant factor is the dynamic programming step, which has a time complexity of O(N^2).

**Comletely Code**

