**Find Routes Between Airports**

In this final exercise, you'll write a program that can find routes between airports and save them to a JSON file.

You have access to three files - airports.dat, routes.dat, and airlines.dat, in CSV format.

The helper module and the starter code make it so the program can be run at the command line:

$ python3 routes.py SFO BOS

$ python3 routes.py SFO BOS --max-segments 3

Running python3 routes.py SFO BOS should result in the creation of a JSON file that contains something like:

{

"1": [

[

"San Francisco International Airport",

"General Edward Lawrence Logan International Airport"

]

],

"2": [

[

"San Francisco International Airport",

"Philadelphia International Airport",

"General Edward Lawrence Logan International Airport"

],

[

"San Francisco International Airport",

"Cleveland Hopkins International Airport",

"General Edward Lawrence Logan International Airport"

],

...,

[

"San Francisco International Airport",

"London Heathrow Airport",

"General Edward Lawrence Logan International Airport"

],

[

"San Francisco International Airport",

"Amsterdam Airport Schiphol",

"General Edward Lawrence Logan International Airport"

]

]

}

You'll have to implement read\_airports and read\_routes - read\_airlines is done for you.

You'll also have to get a bit algorithmic. Once you have the routes, you'll need to search outwards to find your way from the source airport to the target airport in the find\_paths function. One approach can look something like:

# **Start** **at** the **source** airport. The **only** zero-**length** **path** **is** the **empty** **path**.

# **For** **each** **path** **of** **length** **n**, **from** 0 **to** 1 **less** **than** the total number **of** segments

# Find all neighbors **of** **any** airports reachable **at** the **end** **of** a **path** **of** **length** **n**

# These **are** the paths **of** **length** **n** + 1

# **Return** **any** **and** all paths **of** **length** <= **n** that **end** **in** the target airport.

There are some additional data structures that might be able to speed this particular algorithm, but any functional algorithm will work.

The code is all in finding-routes folder.

**Note:** the .dat files for this exercise will not open in the code editor. If you want to see what is in the files you can use the cat command:

cat <filename> | less