DT265/B CMPU4060: Lab 11

DIT School of Computing | Dr Andrew Hines

Instructions

Creating A Dictionary Lookup Class to find Airports

This lab will apply some of the object oriented programming topics introduced in our lectures on creating classes and objects. It will also use the CSV file reading functionality provided by Python. We will also reuse some of the code developed in an earlier lab. Today we are going to code classes to store `Airport` objects and create an `AirportAtlas` to allow us to search for airports and calculate the distance between them.

Problem

We want to be able to load all of the airport information from a comma separated value (CSV) file into memory and allow us to search it for airports and calculate distances between airports.

airport.csv file

Airport data in CSV format from: http://openflights.org/data.html

Example Airport Data:

```
596,Cork,Cork,Ireland,ORK,EICK,51.841269,-8.491111,502,0,E,Europe/Dublin
597,Galway,Galway,Ireland,GWY,EICM,53.300175,-8.941592,81,0,E,Europe/Dublin
599,Dublin,Dublin,Ireland,DUB,EIDW,53.421333,-6.270075,242,0,E,Europe/Dublin
```

Field	Description
AirportID	Unique identifier for this airport.
AirportName	Name of airport. May or may not contain the City name.
CityName	Main city served by airport. May be spelled differently from Name.
Country	Country or territory where airport is located.
code	3-letter IATA code
ICAOcode	ICAO 4-letter ICAO code.
Latitude	Latitude Decimal degrees, usually to six significant digits. Negative is South, positive is North.
Longitude	Longitude Decimal degrees, usually to six significant digits. Negative is West, positive is East.
Altitude	Altitude In feet.
TimeOffset	Timezone Hours offset from UTC. Fractional hours are expressed as decimals, e.g. India is 5.5.
DST	Daylight savings time. One of E (Europe), A (US/Canada), S (South America), O (Australia), Z (New Zealand), N (None) or U
Tz	Timezone Area

Exercises

1. (*5 points*) Create a class to store `Airport` objects. An `Airport` class defines a container object for airports. It should contain any attributes needed to find and display information about the Airport. It should have a constructor that will populate the attributes.

2. (*5 points*) Create a class called `AirportAtlas`. As the name implies this will hold information on all of the airports and allow us to look it up in a variety of ways.

3. (*5 points*) Add a method to `AirportAtlas` called `loadData(self, csvFile)` that populates a `Dictionary` object stored in the `AirportAtlas` with airports. The class should have an attribute that is a dictionary where the keys are airport codes (3 letter IATA codes) and the values are the corresponding Airport object.

4. (*5 points*) Create a constructor for the `AirportAtlas` class that will take the filename of the CSV file containing the airport data. It should call the `loadData(self, csvFile)` method.

5. (*5 points*) Create `testAtlas.py` with a `main()` function to instantiate a new `AirportAtlas`.

6. (*5 points*) Add a method to the `AirportAtlas` called `getAirport(self,code)`. This method should take a three letter code as input and return the Airport object corresponding to the code. Add code to `testAtlas.py` to test the function.

7. (*5 points*) Add a static method to the `AirportAtlas` class to allow:
`greatcirledist(lat1, long1, lat2, long2)`
that will allow the distance between two points to be calculated. You should have the code for this already written from an earlier lab.

8. (*5 points*) Add a method to the `AirportAtlas` that takes two three-letter airport codes as inputs and return the distance between them in kilometres, e.g.
`getDistanceBetweenAirports(self,code1,code2)`.
Add code to `testAtlas.py` to test the function.

9. (*5 points*) Add a `__str__` method to your Airport class to give a descriptive output of the Airport data.

Further Work, Stretch Goals

This lab has brought you through the basics to get your project started. The design patterns used in this lab introduced the idea of using a call as a container for data (the `Airport` class) and as a utility class that allows you to do things like finding the distance between two points based on their airport code (the `AirportAtlas`)

1. How might you expand on this work with additional attributes, methods or classes to deal with currencies and exchange rates?

2. Think about error/exception handling and how you might deal with invalid filenames. Do you need to add something to your code?

3. How does your getAirport handle invalid an invalid code input?

4. How would you add a method to the `AirportAtlas` to search for an `Airport` by Airport Name. Do you need a different dictionary?

5. There is a nice website with video tutorials on Python 3. The section on object oriented programming is good: http://www.pythonschool.net/category/oop.html