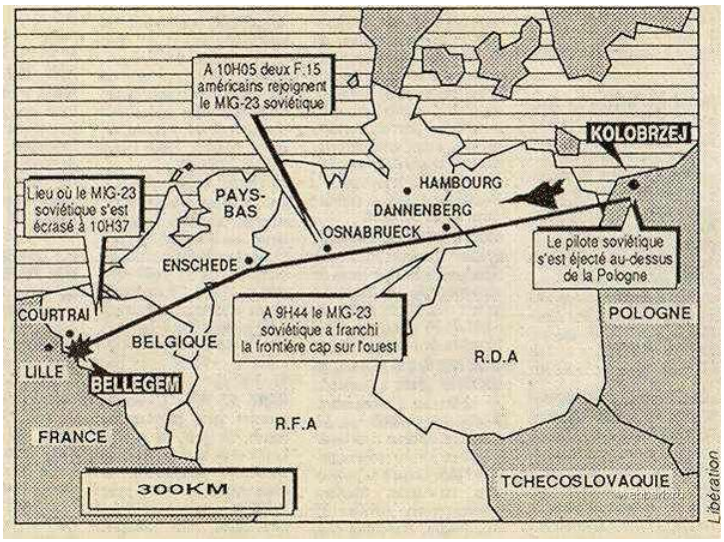
AWOL

On the 4th of July 1989, Soviet MiG-23 pilot Nikolai Skuridin was on a routine training flight near Kolobrzeg, Poland, when his afterburner failed. Skuridin ejected, thinking the engine was completely dead, but the plane recovered and proceeded on autopilot into the west.



Graphics of the flight involving unmanned Soviet MiG-23M "Flogger-B" before crashing into a house in Kortrijk, Belgium, on 4 July 1989, killing an 18-year-old man (taken from French newspaper Libération, description in French).

It must have had a lot of fuel, because it crossed out of Poland into East Germany, then into West Germany, then into the Netherlands, where a startled American air base sent up two F-15s to keep it company. As the MiG passed into Belgium the F-15s were told to shoot it down when it reached the North Sea, but it finally ran out of fuel near the French border, crashing into a house and killing a teenager.

The whole trip had covered 560 miles. Belgian Foreign Minister Mark Eyskens complained that the Soviets had issued no warning and no indication as to whether the pilotless plane was carrying dangerous weapons. It turned out that it was unarmed but carrying ammunition for a 23mm machine gun.

Assignment

Prior to departure, the pilot or flight dispatcher of an airplain must file a flight plan with the local aviation authority. This document generally includes basic information such as departure and arrival points, estimated time and route, alternate airports in case of bad weather, type of flight, number of people on board and information about the aircraft itself.

For this assignment, you are given a text file in **CSV format** (*comma-separated values*) containing information about a series of airports. The file contains twelve information fields for each airport. Of importance for this assignment is only the unique 3-letter code of the airport (first field), together with the latitude and longitude (fields 6 and 7 respectively) where the airport is located. Your task is to use the information in the text file to implement some simple route planning software that can determine the route for given airports of departure and arrival, and a given maximal range of the airplane.

Because it might well be that the distance between the airports of departure and arrival is larger than the range of the airplane, the route planning software might need to select some stopover airports where the airplane can refuel. The flight plan is determined by starting at the airport of departure, and then successively selecting the next **unvisited** airport **within the range** of the airplane that is **closest to the destination** (the airport of arrival). This procedure is repeated until the airport of arrival has been reached, or until no more airport can be reached that meets the required conditions.

In this assignment, locations on Earth are represented as tuples containing two *floating point* numbers, respectively representing the latitude and longitude of the location (expressed in degrees). These are called the coordinates of the location. To determine the distance between two airports having coordinates (b_1, l_1) en (b_2, l_2), we will make use of the Haversine distance d between two points on a sphere that can be calculated by means of the following formula $$\begin{aligned}[c] a &= \left(\sin\left(\frac{b_2-b_1}{2}\right)\right)^2 + \cos(b_1)\cos(b_2)\left(\sin\left(\frac{l_2-l_1}{2}\right)\right)^2 \\ c &= \arctan\left(\sqrt{\frac{a}{1-a}}\right) \\ d &= 2rc \end{aligned}$$ with r equal to the radius of the sphere. Your task is to:

- Write a function coordinates that takes the location of a CSV file as an argument. This file must contain information about a number of airports, in the format as described in the introduction of this assignment. The function must return a dictionary that maps the unique 3-letter code of each airport in the file onto the coordinates of the airport.
- Write a function haversine that takes the coordinates of two locations as its arguments. The function must return the Haversine distance between the two locations, under the assumption that Earth is a sphere with radius $r = 6371\mbox{ km}$. Note that the trigonometric functions (sin, cos, ...) in the math module expect that angles are expressed in radians, not in degrees. However, the math module also has two functions degrees and radians that can be used to convert angles from radians to degrees, and vice versa.
- Use the function haversine to write a function flightplan that can be used to determine the route of an airplane. Three arguments must be passed to the function. The first two arguments represent the unique 3-letter code of the airports of departure and arrival. The third argument is a dictionary mapping the unique 3-letter codes of airports onto their coordinates. In addition, the function has an extra optional parameter range that can be used to pass the range of the airplane (expressed in kilometers; default value: $1000\mbox{ km}$). The function must return the list of 3-letter codes of all airports the airplane will visit on its route from the airport of arrival to the airport of destination, in case the route planning algorithm given above is used. This means that the first and last airports in the list respectively are the airports of departure and arrival. In case the flight planning algorithm ends up in a situation where no more airports can be reached that meet the required conditions, the function must raise an AssertionError with the message no possible route.

CSV format

CSV format (*comma-separated values*) is a specification for storing tabular data (numbers and text) in plain text files. Each line of the file is a data record. Each record consists of one or more fields, separated by commas. The use of the comma as a field separator is the source of the name for this file format. Consider the following tabel as an example

year	make	model	description	price
1997	Ford	E350	airco, abs, moon	3000.00

1999	Chevy	Venture "Extended Edition"	4900.00
1999	Chevy	Venture "Extended Edition, Very Large"	5000.00
1996	Jeep	Grand Cherokee	MUST SELL! air, moon roof, loaded 4799.00

The above table of data may be represented in CSV format as follows

```
Year,Make,Model,Description,Price
1997,Ford,E350,"ac, abs, moon",3000.00
1999,Chevy,"Venture ""Extended Edition""","",4900.00
1999,Chevy,"Venture ""Extended Edition, Very Large""",,5000.00
1996,Jeep,Grand Cherokee,"MUST SELL!
air, moon roof, loaded",4799.00
```

This illustrates some additional rules that must be followed by CSV files:

- the first row might contain the column headers, but this is not mandatory
- any field may be enclosed within double quotes (")
- fields containing commas, double quotes or newlines, or fields starting or ending with spaces should be enclosed within double quotes
- a (double) quotein a field must be represented by two (double) quotes

In order not to take into account all these extra rules when processing CSV files, it is always a good option to use the `csv` module in Python, which is part of the Python Standard Library.

Example

The following interactie sessions assumes the text file airports.csv is located in the current directory.

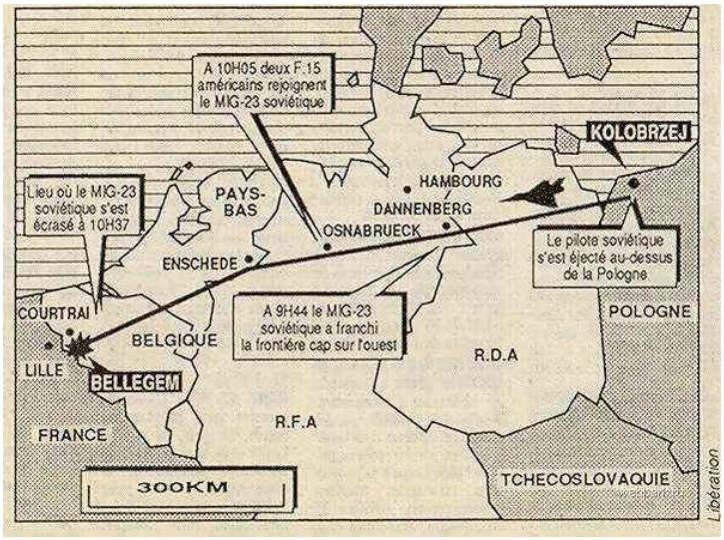
```
>>> coords = coordinates('airports.csv')
>>> len(coords)
9187
>>> type(coords)
<class 'dict'>
>>> coords['BRU']
(50.902222, 4.485833)
>>> coords['CDG']
(49.016667, 2.55)
>>> coords['DCA']
(38.851944, -77.037778)
>>> coords['LAX']
(33.9425, -118.407222)

>>> haversine((50.902222, 4.485833), (49.016667, 2.55)) # BRU <-> CDG
251.2480027355068
>>> haversine((38.851944, -77.037778), (33.9425, -118.407222)) # DCA <-> LAX
3710.8262543589817

>>> flightplan('DCA', 'LAX', coords)
['DCA', 'MTO', 'HLC', 'BFG', 'LAX']
>>> flightplan('DCA', 'LAX', coords, range=2000)
['DCA', 'DDC', 'LAX']
>>> flightplan('DCA', 'LAX', coords, range=4000)
['DCA', 'LAX']
```

```
>>> flightplan('BRU', 'CDG', coords)
['BRU', 'CDG']
>>> flightplan('BRU', 'CDG', coords, range=50)
Traceback (most recent call last):
AssertionError: no possible route
```

De Russische MiG-23 piloot Nikolai Skuridin was op 4 juli 1989 een routinevlucht aan het uitvoeren in de buurt van Kolobrzeg in Polen, toen één van zijn naverbranders de geest gaf. Skuridin gebruikte zijn schietstoel in de veronderstelling dat de motor het volledig had begeven, maar het vliegtuig herstelde zich en vloog op automatische piloot richting het westen.



Grafische voorstelling van de route die de MiG-23 van Nikolai Skuridin maakte op 4 juli 1989, alvorens in de buurt van Kortrijk neer te storten (uit de Franse krant Libération).

Het toestel moet vrij veel brandstof aan boord gehad hebben, want het vloog vanuit Polen Oost-Duitsland binnen, daarna naar West-Duitsland, en dan naar Nederland waar een gealarmeerde Amerikaanse luchtmachtbasis twee F-15s liet opstijgen om het gezelschap te houden. Toen de MiG het Belgische grondgebied betrad, kregen de F-15s opdracht om het toestel neer te halen van zodra het de Noordzee zou bereiken. Vlak voor de Franse grens kwam het echter zonder brandstof te zitten en crashte het in een huis, waarbij een tiener het leven liet.

De volledige route die het vliegtuig had afgelegd was meer dan 900 kilometer lang. Toenmalig Belgisch Minister van Buitenlandse Zaken Mark Eyskens was razend over het feit dat de Sovjets geen waarschuwing uitgestuurd hadden en geen enkele informatie vrijgegeven hadden over mogelijke gevaarlijke wapens die het onbemande vliegtuig aan boord had. Achteraf bleek dat het vliegtuig onbewapend was, op wat munitie voor een 23mm machinegeweer na.

Opgave

Alvorens een vliegtuig mag opstijgen, moet de piloot of de vliegtuigmaatschappij eerst een vliegplan indienen bij de luchtverkeersleiding. Naast de luchthaven van vertrek en aankomst, bevat dit vliegplan welke route men wil volgen, op welke hoogte men deze route wil vliegen, met welk vliegtuig (type, vliegtuigregistratienummer), hoeveel passagiers er aan boord zijn, enzoverder.

Bij deze opgave krijg je een tekstbestand in **CSV formaat** (*comma-separated values*) met gegevens over een aantal luchthavens. Voor elke luchthaven bevat het bestand twaalf

informatievelden, waarbij voor deze opgave enkel de unieke 3-lettercode van de luchthaven (eerste veld) belangrijk is, samen met de breedte- en lengtegraad (respectievelijk velden 6 en 7) van de ligging van de luchthaven. Op basis van deze informatie moet je een eenvoudige routeplanner schrijven waarmee de vliegroute kan bepaald worden voor gegeven luchthavens van vertrek en aankomst, en voor een gegeven maximale actieradius van een vliegtuig.

Omdat de afstand tussen de luchthaven van vertrek en de luchthaven van aankomst groter kan zijn dan de actieradius van het vliegtuig, moet de routeplanner mogelijks een aantal tussenliggende luchthavens uitkiezen waar het vliegtuig kan bijtanken. Het vliegplan wordt vastgelegd door startend vanaf de luchthaven van vertrek, telkens de volgende **nog niet bezochte** luchthaven **binnen de actieradius** van het vliegtuig te bezoeken die het **dichtst bij de bestemming** gelegen is. Deze procedure wordt herhaald totdat de luchthaven van aankomst bereikt wordt, of totdat geen enkele luchthaven kan bereikt worden die aan de gestelde voorwaarden voldoet.

In deze opgave stellen we een punt op Aarde voor als een tuple met twee *floating point* getallen, die respectievelijk de breedte- en de lengtegraad van het punt voorstellen (uitgedrukt in graden). We noemen dit de coördinaten van het punt. Voor het bepalen van de afstand tussen twee luchthavens met coördinaten (b_1, l_1) en (b_2, l_2) maken we gebruik van de **Haversine-afstand** d, die bepaald wordt aan de hand van de formule $$\begin{aligned}[c] a &= \left(\sin\left(\frac{b_2-b_1}{2}\right)\right)^2 + \cos(b_1)\cos(b_2)\left(\sin\left(\frac{l_2-l_1}{2}\right)\right)^2 \\ c &= \arctan\left(\sqrt{\frac{a}{1-a}}\right) \\ d &= 2rc \end{aligned}$$ Hierbij stelt r de aardstraal voor. Gevraagd wordt:

- Schrijf een functie `coordinaten` waaraan de locatie van een CSV bestand moet doorgegeven worden. Dit bestand moet informatie bevatten over een aantal luchthavens, volgens het formaat zoals beschreven in de inleiding van de opgave. De functie moet een dictionary teruggeven die de unieke 3-lettercode van elke luchthaven uit het bestand afbeeldt op de coördinaten van de luchthaven.
- Schrijf een functie `haversine` waaraan twee coördinaten moeten doorgegeven worden. De functie moet de Haversine-afstand tussen de punten met gegeven coördinaten teruggeven, als uitgegaan wordt van een constante benadering van 6371 km als straal van de Aarde. Let erop dat de goniometrische functies (`sin`, `cos`, ...) uit de `math` module hoeken verwachten die uitgedrukt zijn in radialen, niet in graden. De `math` module heeft echter ook twee functies `degrees` en `radians` die hoeken omzetten van radialen naar graden, en omgekeerd.
- Gebruik de functie `haversine` om een functie `vliegplan` te schrijven waarmee de route van een vliegtuig kan bepaald worden. Aan deze functie moeten drie argumenten doorgegeven worden. De eerste twee argumenten stellen de unieke 3-lettercodes voor van de luchthavens van vertrek en aankomst. Het derde argument is een dictionary die de unieke 3-lettercodes van luchthavens afbeeldt op hun coördinaten. Bovendien heeft deze functie nog een optionele parameter `actieradius` waaraan de actieradius van het vliegtuig kan doorgegeven worden (uitgedrukt in kilometer; standaardwaarde: $1000\mbox{ km}$). De functie moet de lijst van 3-lettercodes teruggeven van alle luchthavens die het vliegtuig op zijn route zal aandoen, op basis van het routeplanningsalgoritme dat hierboven werd beschreven. Hierbij zijn de eerste en laatste code respectievelijk deze van de gegeven luchthavens van vertrek en aankomst. Indien bij het bepalen van het vliegplan geen enkele luchthaven kan bereikt worden die aan de gestelde voorwaarden voldoet, dan moet de functie een `AssertionError` opwerpen met de boodschap `geen mogelijke route`.

CSV formaat

CSV formaat (*comma-separated values* in het engels) is een specificatie voor tekstbestanden waarin gegevens in tabelvorm opgeslaan worden. Hierbij worden de waarden in de verschillende kolommen van de tabel van elkaar gescheiden door komma's, en worden de rijen van de tabel van elkaar gescheiden door regeleindes. Beschouw bijvoorbeeld de volgende tabel

jaar	merk	type	omschrijving	prijs
1997	Ford	E350	airco, abs, moon	3000.00
1999	Chevy	Type "Extended Edition"		4900.00
1996	Jeep	Grand Cherokee	IS VERKOCHT! air, moon roof, loaded	4799.00

De gegevens in deze tabel worden op de volgende manier opgeslaan in een CSV bestand

```
jaar,merk,type,omschrijving,prijs
1997,Ford,E350,"airco, abs, moon",3000.00
1999,Chevy,"Type ""Extended Edition""",,4900.00
1996,Jeep," Grand Cherokee ","IS VERKOCHT!
air, moon roof, loaded",4799.00
```

Dit illustreert nog een aantal bijkomende regels die gelden voor CSV bestanden:

- de eerste regel kan kolomtitels bevatten, maar die regel is niet verplicht
- velden die komma's, aanhalingstekens (") of regeleindes bevatten, en velden die beginnen en/of eindigen met een spatie, moeten omsloten worden door aanhalingstekens
- aanhalingstekens binnen een veld worden verdubbeld

Om bij het verwerken van CSV bestanden zelf geen rekening te moeten houden met al deze extra regels, kan je in Python best gebruik maken van de `csv` module, die deel uitmaakt van de Python Standard Library.

Voorbeeld

Bij onderstaande voorbeeldsessie gaan we ervan uit dat het tekstbestand luchthavens.csv zich in de huidige directory bevindt.

```
>>> coords = coordinaten('luchthavens.csv')
>>> len(coords)
9187
>>> type(coords)
<class 'dict'>
>>> coords['BRU']
(50.902222, 4.485833)
>>> coords['CDG']
(49.016667, 2.55)
>>> coords['DCA']
(38.851944, -77.037778)
>>> coords['LAX']
(33.9425, -118.407222)

>>> haversine((50.902222, 4.485833), (49.016667, 2.55)) # BRU <-> CDG
251.2480027355068
```

```
>>> haversine((38.851944, -77.037778), (33.9425, -118.407222)) # DCA <-> LAX
3710.8262543589817

>>> vliegplan('DCA', 'LAX', coords)
['DCA', 'MTO', 'HLC', 'BFG', 'LAX']
>>> vliegplan('DCA', 'LAX', coords, actieradius=2000)
['DCA', 'DDC', 'LAX']
>>> vliegplan('DCA', 'LAX', coords, actieradius=4000)
['DCA', 'LAX']
>>> vliegplan('BRU', 'CDG', coords)
['BRU', 'CDG']
>>> vliegplan('BRU', 'CDG', coords, actieradius=50)
Traceback (most recent call last):
AssertionError: geen mogelijke route
```