CONTENTS OF THIS FILE

INTRODUCTION

This prototype is a travel recommendation engine which uses natural language processing (NLP) - a machine learning concept - to determine similarity between travel destination on various algorithms. The program uses a graphical database neo4j and several Python libraries such as natural language tool kit (NLTK) and flask to run the model and interface.

The is user is asked to provide certain information to in order to determine the best fitting travel destinations.
The engine is providing several information about the recommended destinations including several links to further information.

USE CASE DESCRIPTION

The user is asked to provide following information:

- start date of travel
- approximate duration of travel
- three previous destinations the user liked
- continents the user is interested in
- preferred activity style

All these information get extracted via APIs or are provided as text files in this repository. Currently not all information are used resp. implemented in the model. Version 1.0. includes following technical

features:

 - natural language processing (NLP) based on latent semantic indexing (LSI) or latent dirichlet allocation (LDA) using
 the provided destination descriptions in the respective folder
 - store destination descriptions, respective continents and similarity scores between destinations in neo4j database
 - a running interface with basic functionality based on flask
 - top things to do in recommended city via Google Search API
 - current weather in destinations via OpenWeatherMap API

Goal of this showcase is to demonstrate how natural language processing and recommendation engines are working
and how additional information influence a user travel decision.

REQUIREMENTS, INSTALLATION

 The prototype is implemented in Python. A distribution like Anaconda
 is recommended which you can find here: https://www.anaconda.com/download/

 Java server is presupposed. In case you are not sure if you have it installed, navigate to your respective
 Java folder (probably C:/Program Files(x86)/Java/jreX.X.X/bin) and check whether here is a folder
 named "server". If not, create one named "server" and copy all files from the "client" folder in there.

 1) Install graphical database neo4j (https://neo4j.com/)

 1.1) Extract the provided zip-file neo4j-community-3.3.2-windows
 (for Windows 64-bit, further / newer distributions which also work are
 available at: https://neo4j.com/download/other-releases/) to your folder of choice.

 1.2) Navigate to your selected folder and go to 'conf/neo4j.conf'.
 Open the file in a text editor and delete the "#" in the line with the command 'dbms.security.auth_enabled=false' (26) to prevent authentification.

 1.3) Open cmd prompt. Navigate to the extracted folder '..\neo4j-community-3.3.2' and start the database with the command 'bin\neo4j console'.

 1.4) Go to localhost:7474 in your browser to see the interface.

 1.5) Stop the server with the command ctrl+c.

 1.6) Detailed documentation: https://neo4j.com/docs/operations-manual/current/installation/windows/

 2) Preparation for Web Application (UI)

 2.1) Create virutal environment inside the root directory of the prototype (../travel_recommender). To do so:

 2.2) Open anaconda prompt.

 2.3) Run virtualenv <env_name> (make sure virtualenv is installed, if not: pip install virtualenv)

 2.4) Activate virtual environment. Go to root_dir\<env_name>\Scripts\ and type 'activate.bat'. Go Back to root directory with 'cd..'.

```
 2.5)  Now install all dependencies with: pip install -r requirements.txt

 2.6)  run "python -m nltk.downloader stopwords" from commandline to
download stopword file.

 2.7)  You can run the prototype with the command 'python init.py' from
root directory. Please setup the config.py file before you start.


PROTOTYPE STRUCTURE
-------------------

Folder structure:                        (#COMMENTS)

    |--\destinations
      |--(all .txt files)                    #each respective <IATA
CODE>.txt file is in here wikipedia is in here
    |--\recommender
      |--__init__.py                       #necessary import file
      |--api.py                           #all used api's to web services
      |--config.py                         #setup parameters for the program
      |--database.py                       #reads from the destination file
folder, tokenizes the content and writes it into the database
      |--nlp.py                           #the nlp module calculates the
similarity between the provided texts
      |--weather_api.py                    #deprecated
      |--\gui
         |--__init__.py                    #necessary import file
         |--app.py                        #web application with flask
         |--README.txt                     #tutorial how to install flask
         |--requirements.txt                  #requirements on how to run the
app
         |--\static                       #static data such as pictures
         |--\template                      #html templates for flask
      |--\photoapi                         #crawls destination pictures
    |--\<virtualenv>                        #if installed. is not a must.
      |--\include
      |--\lib
      |--\scripts
         |--activate.bat                     #needs to be activated before
running the prototype
      |--\tcl
      |--pip-selfcheck.json
    |--destinations.csv                    #csv-file of all destinations
    |--requirements.txt                    #requirements for the virutal
environment required by the interface
    |--init.py                             #start file
    |--all_destinations.csv                #all destinations in csv Format
    |--destination_similarity.csv           #similarity scores between each
destination
    |--destinations.csv / .xlsx / .txt          #raw file which can be
adjusted for development
    |--model_selector.ipynb                #python notebook which provides
information about model selection
    |--NLP_knowledge.ipynb                    #python notebook which
provides information about topic selection of natural language processing
    |--Photo Crawler.ipynb                    #script that crawls
destination pictures offline. unsplash api key necessary.
```