**NYC Itinerary Planner README**

**Pre-requisite**

* **Python Installation:** [Download Python | Python.org](https://www.python.org/downloads/)
* **Clingo Installation:** [Releases · potassco/clingo (github.com)](https://github.com/potassco/clingo/releases)

**The project has 3 main parts:-**

* + main.py: main driver code written in python
  + ParsePage.py: crawls TripAdvisor and parses data to build knowledge base
  + itinerary.lp: clingo code called by main.py to generate the stable model

**Running the Program**

* Change the clingo path in main.py to the one on your machine or replace the absolute path with just “clingo” if it is added to the PATH variable.
* Run main.py on the terminal in its directory (“python main.py”) and give the specified inputs to get your itinerary.
* Currently, the code handles only the happy cases as it is the MVP, for example, if you give a tab separated list when asked for a comma separated list, the code would completely break as we don’t handle exceptions.
* The program currently outputs the best path it considers in terms of number of destinations visited within the given time and money. The MVP did not have money as a constraint.

**ParsePage.py**

* Uses 2 html page sources with the top 60 NYC destinations according to TripAdvisor as the initial input.
* Parses the pages using a regex and gets the list of 60 destinations and cleans the data.
* Uses TripAdvisor Location Search API to get the Location IDs of these places.
* Uses TripAdvisor Location Details API to get all information about these places.
* Writes all the information obtained so far to file
* Parses Location Details to get the corresponding co-ordinates and writes it to another file.

**main.py**

* Each function has its own description in the comments and in its documentation.
* Reads the list of destinations generated from ParsePage.py along with co-ordinates.
* Reads user inputs including start point, end point, mandatory destinations, time, cost. (We would not be using cost to generate the itinerary as of now)
* Creates a knowledge base by creating edges between all destinations (it is possible to get from anywhere to anywhere). We use heuristics to generate the time between 2 destinations. We calculate the distance between them using the co-ordinates and assume it takes 100 minutes to travel between the 2 furthest points and time between any 2 points varies proportional to the distance.
* Generates all the other constraints provided by the user to clingo friendly syntax and writes the knowledge base as well as the constraints along with the actual clingo logic (taken from itinerary.lp) to main.lp
* Calls the clingo program main.lp and then parses the output.
* Gives the destinations in order if there is a feasible solution else the program ends with a print statement.

**Miscellaneous Notes**

* We could not use all 60 destinations compiled to create a complete graph as that crashes the program. We create over 3500 edges and the program is not efficient enough to handle that. Hence we pick first 10 destinations to verify the code.