Programmer tutorial-

1. Project set up-
    a. Clone for repository - https://github.com/sagivmap/final_project.git Or download as zip from https://github.com/sagivmap/final_project
    b. Recommended to work with PyCharm – this tutorial will be PyCharm oriented.
    c. Open the cloned project with PyCharm.
    d. Review the requirements.txt file and make sure to install all the python modules via pip install to the requirements.txt file or each module separately.
    e. Set FlaskTry folder as the root directory
    f. Run **FlaskTry/mainServer.py** to launch the web application.
    g. Web app should be at local host with port 5000

2. Project modules summary –
    a. Crawler – modules that contain the two crawlers of the project-
        i. Facebook crawler
        ii. Twitter crawler

    Modules that responsible for information gathering to be input to the AlgorithmSolver module.

    b. AlgorithmSolver – main module that implement the Information Flow Control Algorithme as describe at Ehud Gudes and Nadav Voloch's article – output is the OSN graph as nodes and egfed and score for the second level nodes (TSP)

    c. FlaskTry – Web server which run Flask application, handling requests from client side and execute the Crawler and AlgorithmSolver modules.

3. Crawler-
    a. Facebook Crawler-
    Main script of Facebook Crawler is **FBCrawler.py** – script that connect to facebook account via lite version of Facebook and get in HTTP requests the first circle friends of user and their attribute (Total friends, Number of mutual friends, Age of user account, Friendship duration) and their friend OSN attribute (Second level friends of root user).
    Put result in **FlaskTry/data** folder as csv file that contains all data.
    First circle data stored at - **FlaskTry/first_and_second_data_raw**
    Second circle data stored at - **FlaskTry/first_circle_initial_data**

    FBCrawler's config file-
    At Crawler/config there is .ini file that called config.ini which has all the configuration that should be to FBCrawler.
    Most important thing that is to know about that file is that include the class of HTML tags that Facebook give to friend list of Facebook user – it can change once in while so this is important to know that if the crawler doesn't work to check if the class name changes by Facebook.

    utils.py –
    helper module that helps to hendle files in write and read functions, extract from strings and pretty much to ease the readability of the FBcrawler.

    FBcrawler module use BeautifulSoup for HTML scrapping – quick tutorial can be found here - https://www.crummy.com/software/BeautifulSoup/bs4/doc/

    b. Twitter Crawler –
    Basically the output of twitter crawler is the same of Facebook crawler but Twitter supply friendly API that wrapped as python module that called – twython –
    quick tutorial can be found here - https://twython.readthedocs.io/en/latest/
    Twitter Crawler sends the result CSV to user email that initialize the crawling.

4. AlgorithmSolver –

    Main algorithm of the project – read ADD file for CSV file input file for the wanted format.

    Generate TSP score for each friend in second circle.

    At AlgorithmSolver/config there is config file that has the attributes barriers.

    There is createJson.py that create json file from the output of the AlgorithmSolver for web presentation.

5. FlaskTry -
   Web server which run Flask application, handling requests from client side and execute the Crawler and AlgorithmSolver modules.

   FlaskTry/static – folder that include all static files of the web app, such as css, fonts, images, javascript files and more file that not likely will change.

   FlaskTry/templates – all HTML files.

   FlaskTry/uploads – folder that include all files that user upload to the server for represent the uploaded CSV to the server.

   FlaskTry/csvfile – result folder for twitter crawler

   FlaskTry/data – result folder for Facebook crawler

   FlaskTry/mainServer.py – main script for running the server – handle all requests from user and return good result or error message.

   JavaScript files-

   graphPresentation.js – script that use D3 package to add data manually

   staticGraphPresentation.js - – script that use D3 package to present already calculated CSV.

   Important info:
   D3 - https://github.com/d3/d3-force
   Flask - http://flask.pocoo.org/docs/1.0/