# Public transport network of Budapest

## 1. Project description

This project is an exam task of Eötvös Loránd University. The project description was the following:

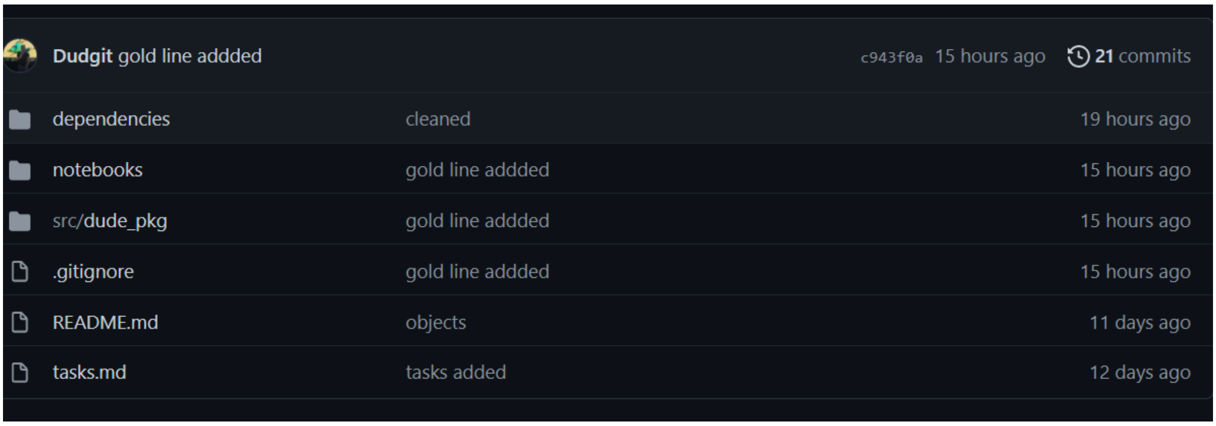
```
1. Download a GTFS status of the BKK from <https://bkk.hu/apps/gtfs/> this
database changes almost on a daily basis, so include in your report the time
you obtained it. (You don't have to keep it up to date.)

2. Load those tables to your database that are relevant in creating a network
between the routes. Which are these? If any tables need some
preprocessing/cleaning you should do it before this task in the environment of
your choice.

3. Create the link list of the transport routes! You can use the environment
of your choice to calculate permutations if you don't want to do it in SQL.

4. Visualize the network you've obtained! Nodes with more links should be
bigger, color the nodes according to what kind of transport they're: black for
nighttime, yellow for trams, blue for busses etc. and label them what route
they're. What seem to be the most well connected nodes in this network? How
can you fix this? (<https://networkx.org/)>

5. Calculate the degree distribution and the average degree of the >fixed<
network. Visualize the results! What does the shape of the distribution tell
you about this particular network?
(<http://networksciencebook.com/chapter/2#degree)>

6. Calculate the clustering coefficient for each node, from that obtain the
average clustering, and also calculate the global clustering coefficient!
Visualize the results! How does the distribution look like? How does the
average clustering compare to the global clustering?
(<http://networksciencebook.com/chapter/2#clustering>;
<http://networksciencebook.com/chapter/2#advanced)>

7. Measure degree correlation function of the network! Visualize the results!
What does this tell you about the assortativity of the network?
(<http://networksciencebook.com/chapter/7#measuring-degree)>
```

## 2. Solutions

Obviously, I solved the tasks in python. I’ve created a data structure, some libraries (though the project not require that, it was much more convenient for me), a virtual environment and a gihtub repository.

In the repo I’ve uploaded every necessary file to identically reproduce my work. However, the data can’t be stored in this repo, due to its size. It is accessible from my google-drive. I uploaded a link to it.

The structure of the project is the following:

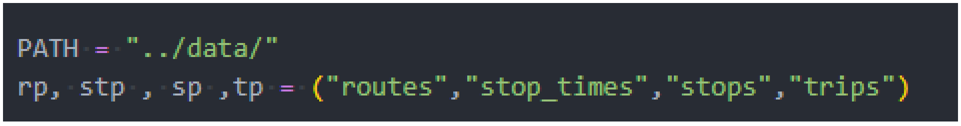


Dudgit gold line addded c943f0a 15 hours ago 21 commits

| | | |
|---|---|---|
| dependencies | cleaned | 19 hours ago |
| notebooks | gold line addded | 15 hours ago |
| src/dude_pkg | gold line addded | 15 hours ago |
| .gitignore | gold line addded | 15 hours ago |
| README.md | objects | 11 days ago |
| tasks.md | tasks added | 12 days ago |

## 3. Results

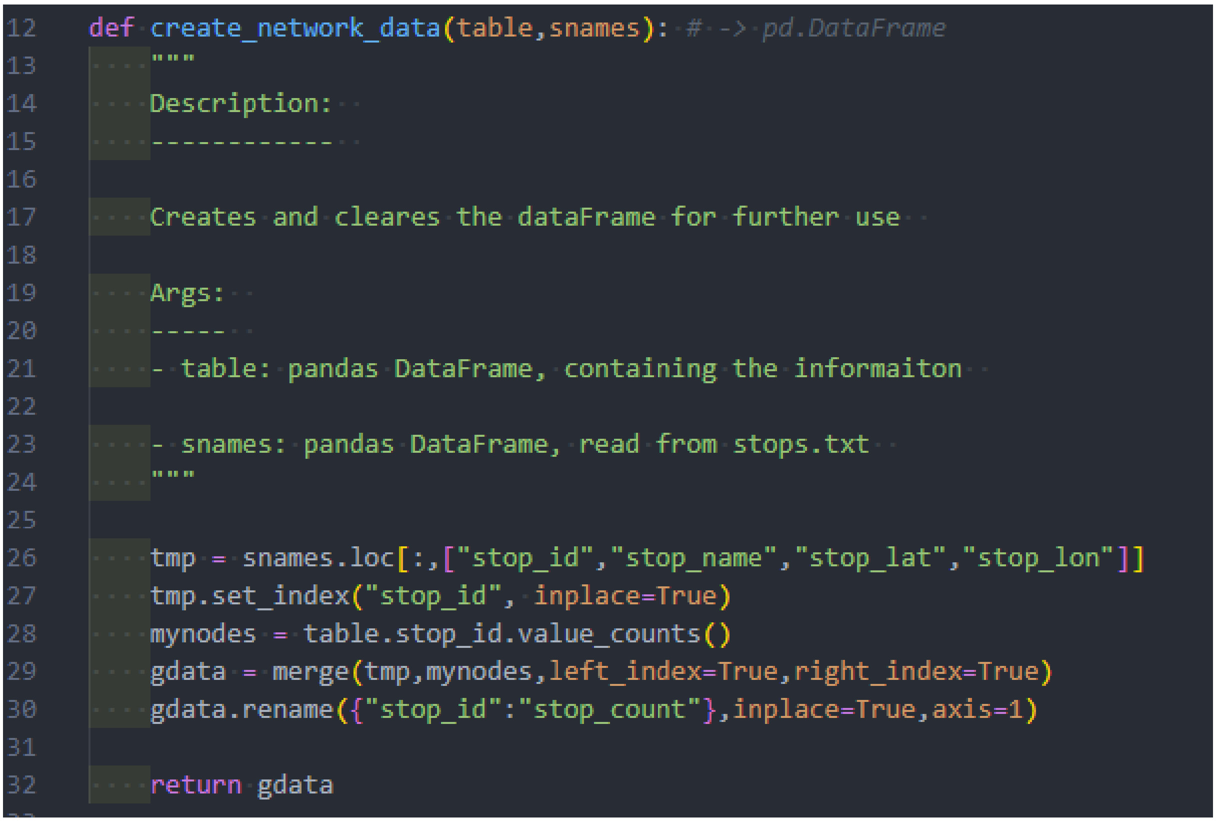
1. Task 1
   Downloading the data was quite simple and easy. Since the link was given too, I don’t think this task needs any more explanation.
2. Task 2
   Inside the source/dude_pkg folder, there is the objects.py file, which contains the following lines:



```
PATH = "../data/"
rp, stp , sp ,tp = ("routes","stop_times","stops","trips")
```

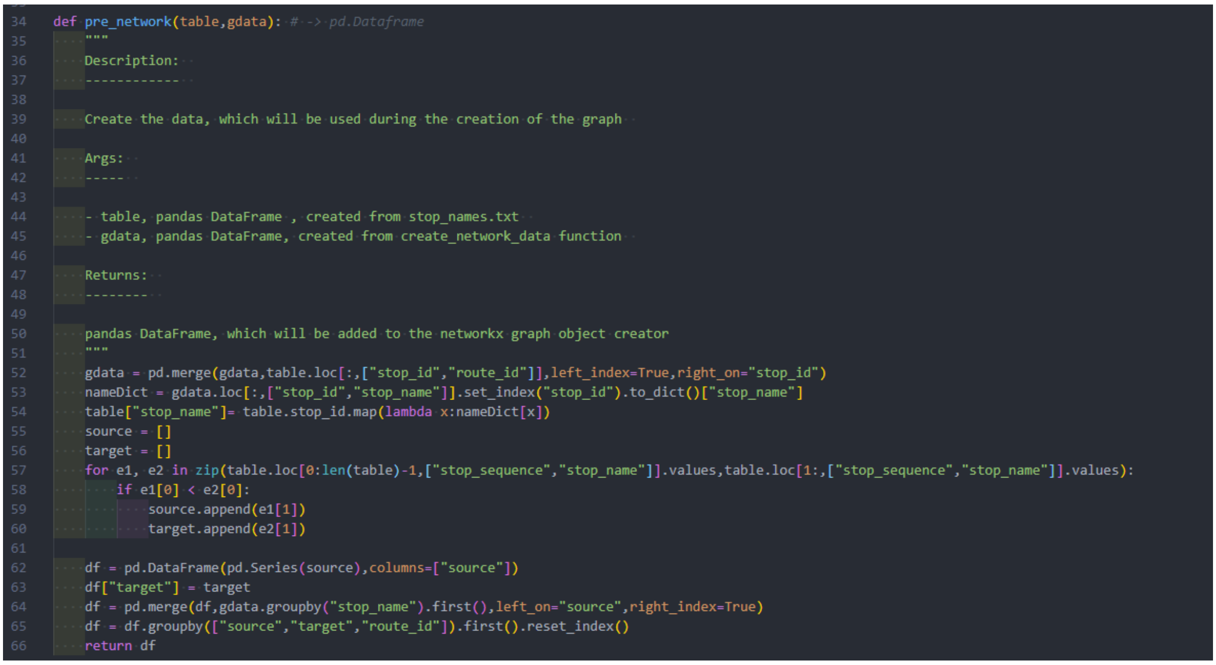
   These are the names of the tables, which I thought essential for solving the task.
3. Task 3
   I have a function for merging the data and an other one to do some preprocess.

   They are in the source/dude_pkg/functions.py. The create_network_data just merge 2 pandas DataFrame object on some specific label.



```python
def create_network_data(table,snames): # -> pd.DataFrame
    """
    Description:
    ------------

    Creates and cleares the dataFrame for further use

    Args:
    -----
    - table: pandas DataFrame, containing the informaiton

    - snames: pandas DataFrame, read from stops.txt
    """

    tmp = snames.loc[:,["stop_id","stop_name","stop_lat","stop_lon"]]
    tmp.set_index("stop_id", inplace=True)
    mynodes = table.stop_id.value_counts()
    gdata = merge(tmp,mynodes,left_index=True,right_index=True)
    gdata.rename({"stop_id":"stop_count"},inplace=True,axis=1)

    return gdata
```

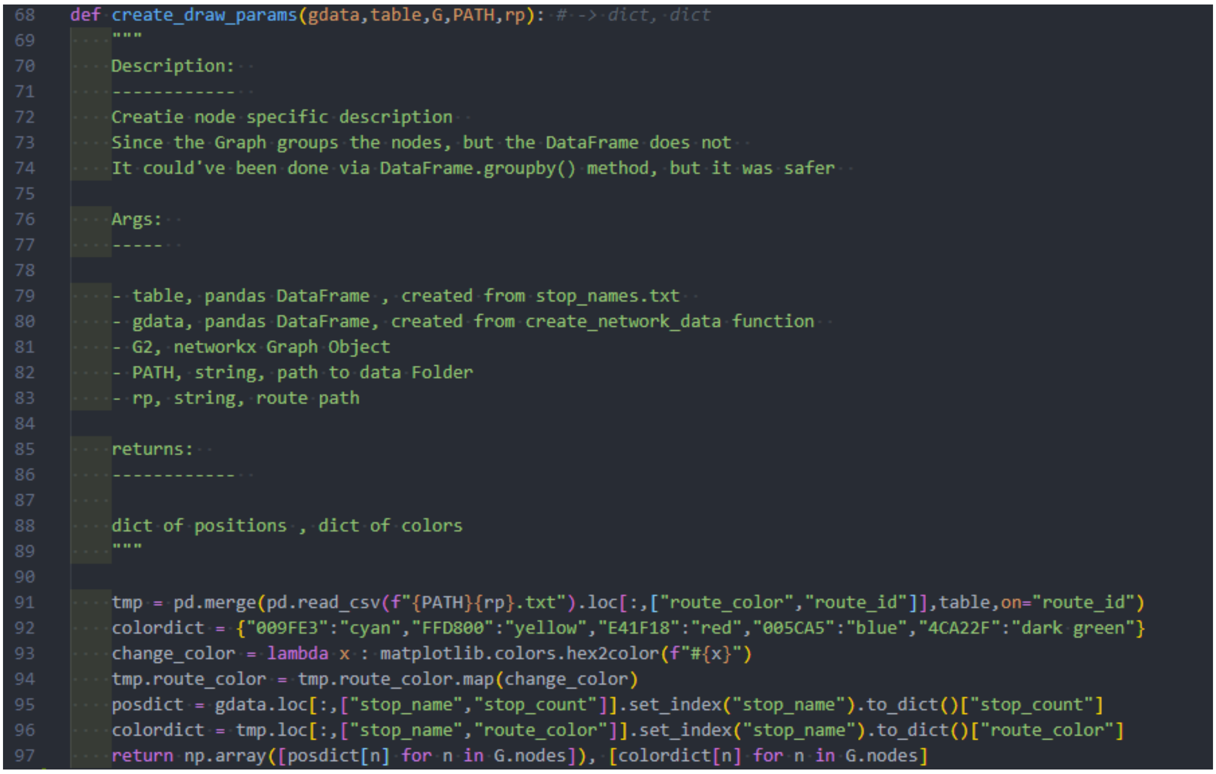
And the pre_network clears the table, so the result can be used for creating the Networkx graph object we want to use.



```python
def pre_network(table,gdata): # -> pd.Dataframe
    """
    Description:
    ------------

    Create the data, which will be used during the creation of the graph

    Args:
    -----

    - table, pandas DataFrame , created from stop_names.txt
    - gdata, pandas DataFrame, created from create_network_data function

    Returns:
    --------

    pandas DataFrame, which will be added to the networkx graph object creator
    """
    gdata = pd.merge(gdata,table.loc[:,["stop_id","route_id"]],left_index=True,right_on="stop_id")
    nameDict = gdata.loc[:,["stop_id","stop_name"]].set_index("stop_id").to_dict()["stop_name"]
    table["stop_name"]= table.stop_id.map(lambda x:nameDict[x])
    source = []
    target = []
    for e1, e2 in zip(table.loc[0:len(table)-1,["stop_sequence","stop_name"]].values,table.loc[1:,["stop_sequence","stop_name"]].values):
        if e1[0] < e2[0]:
            source.append(e1[1])
            target.append(e2[1])

    df = pd.DataFrame(pd.Series(source),columns=["source"])
    df["target"] = target
    df = pd.merge(df,gdata.groupby("stop_name").first(),left_on="source",right_index=True)
    df = df.groupby(["source","target","route_id"]).first().reset_index()
    return df
```

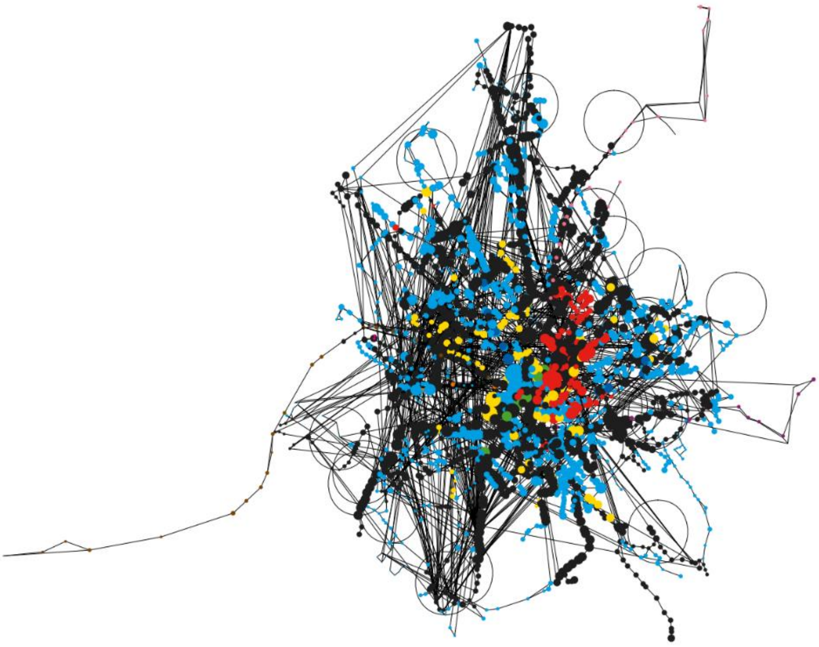
4. Task 4

The following dictionaries were made to solve this problem:



```python
def create_draw_params(gdata,table,G,PATH,rp): # -> dict, dict
    """
    Description:
    ------------
    Creatie node specific description
    Since the Graph groups the nodes, but the DataFrame does not
    It could've been done via DataFrame.groupby() method, but it was safer

    Args:
    -----

    - table, pandas DataFrame , created from stop_names.txt
    - gdata, pandas DataFrame, created from create_network_data function
    - G2, networkx Graph Object
    - PATH, string, path to data Folder
    - rp, string, route path

    returns:
    ------------

    dict of positions , dict of colors
    """

    tmp = pd.merge(pd.read_csv(f"{PATH}{rp}.txt").loc[:,["route_color","route_id"]],table,on="route_id")
    colordict = {"009FE3":"cyan","FFD800":"yellow","E41F18":"red","005CA5":"blue","4CA22F":"dark green"}
    change_color = lambda x : matplotlib.colors.hex2color(f"#{x}")
    tmp.route_color = tmp.route_color.map(change_color)
    posdict = gdata.loc[:,["stop_name","stop_count"]].set_index("stop_name").to_dict()["stop_count"]
    colordict = tmp.loc[:,["stop_name","route_color"]].set_index("stop_name").to_dict()["route_color"]
    return np.array([posdict[n] for n in G.nodes]), [colordict[n] for n in G.nodes]
```

The result of the visualization is the following:



Graph of Budapest public transport system
Colorized by original BKK colors

5. Task 5, 6,7
   These tasks were solved in the jupyter notebook main.ipynb and only there. Every result and figure are contained, so I do not think it is necessary to explain them here.