

Arvid Imgård, Edvin Gustavsson reporting Ali Alkhaled

Section 1:

- Q1: Does the application run? --> **Yes**
- Q2: Does the application display the complete map of tram lines? --> **Yes**
- Q3: Is it possible to query shortest path between any two points? --> **Yes**

Section 2:

- B1: Is the submission successfully accounting for Bonus Part 1? --> **Yes**
- B2: Is the submission successfully accounting for Bonus Part 2? --> **Yes**

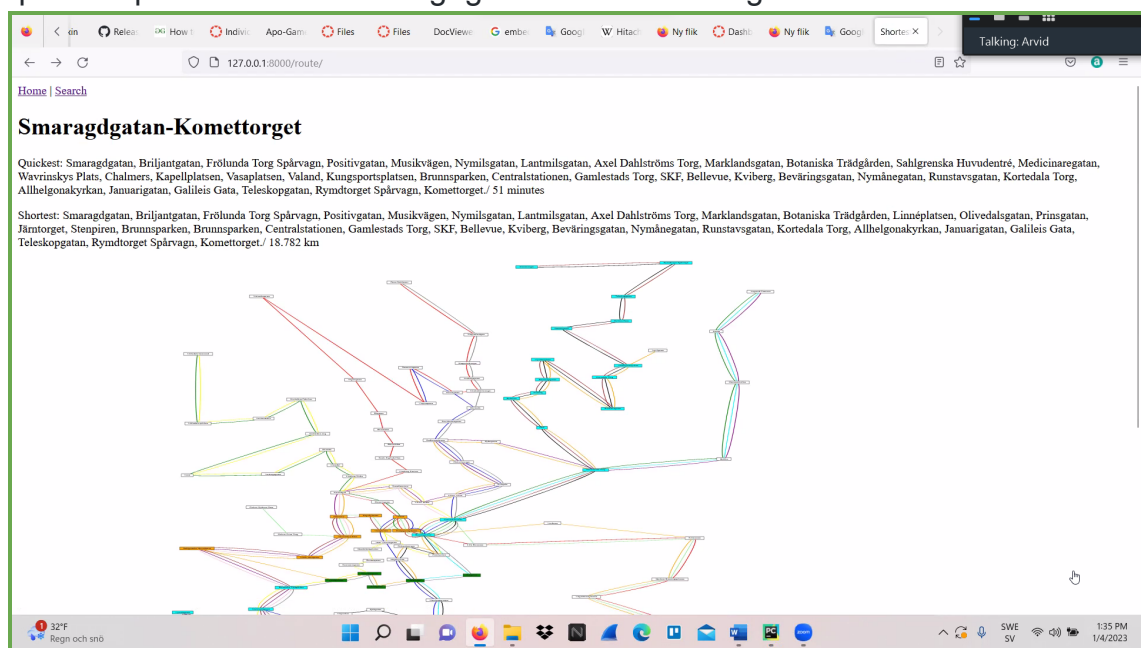
Section 3: Code quality

Very high quality overall. Good code structure with appropriate decomposition and naming of variables which helped readability.

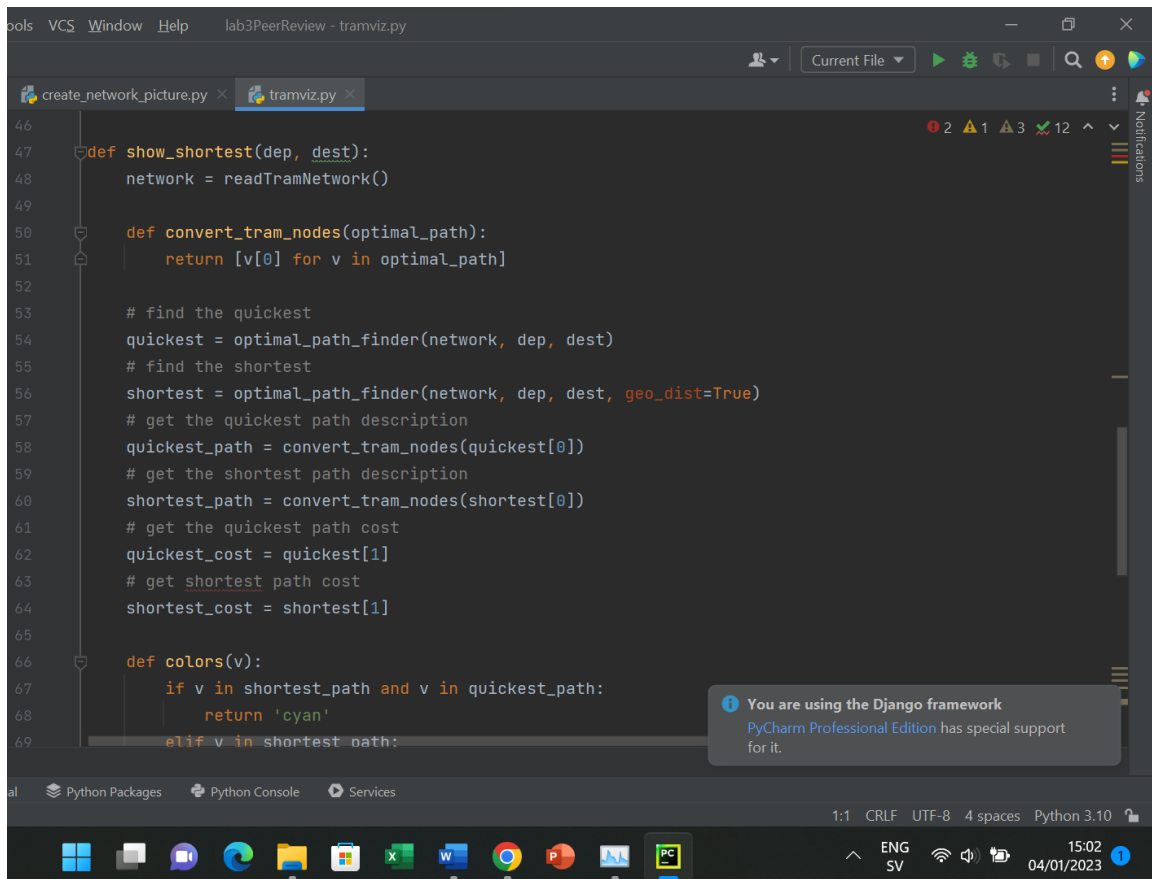
- code from lab 2 has been properly reused (i.e., in an efficient way without boilerplate code)
 - **Yes**
- dijkstra has been implemented and used as intended: there is just one definition of the function itself, and different distances are obtained by just changing the cost function
 - **Yes**

Section 4:

- Screenshot showing the web application displaying both the shortest and the quickest path between Smaragdsgatan and Komettorget.



- Screenshots showing the function `show_shortest()` in the `tramviz.py` file



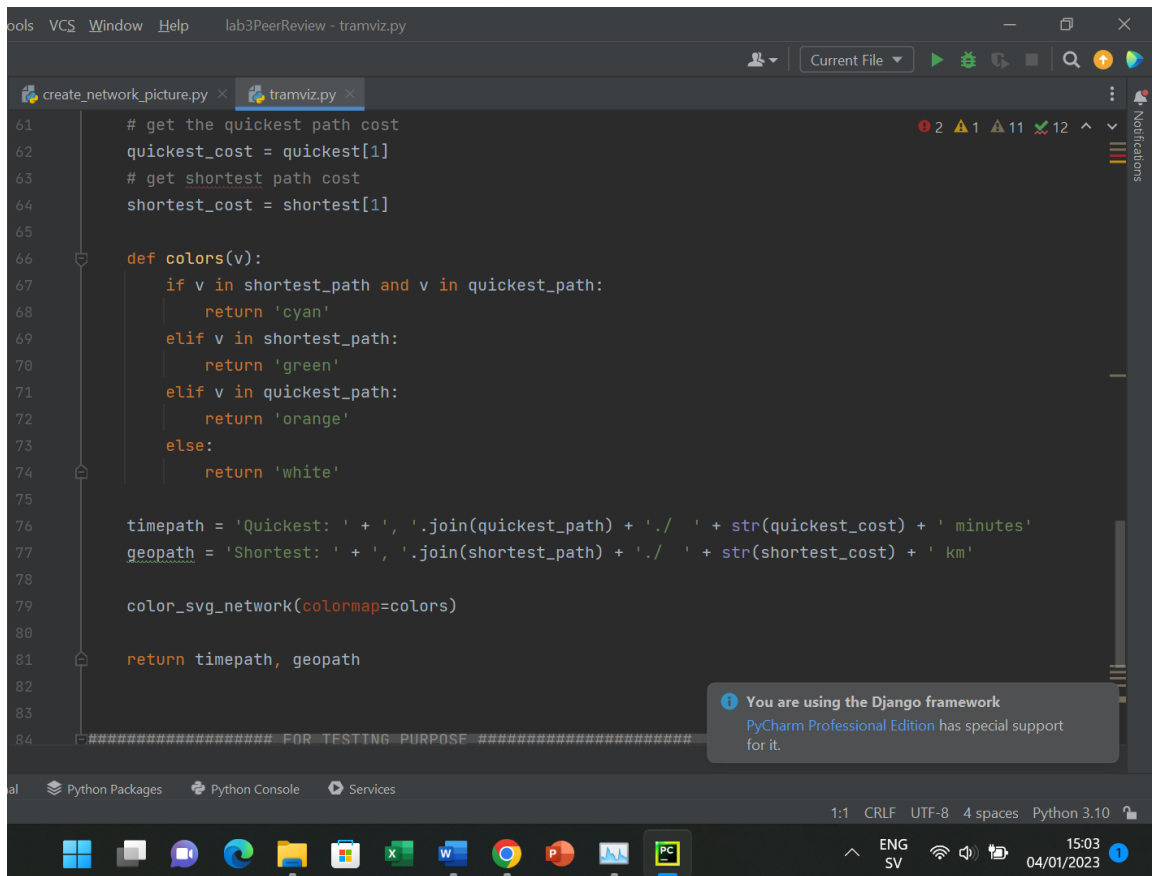
This screenshot shows the PyCharm IDE with the `tramviz.py` file open. The `show_shortest` function is defined, which takes `dep` and `dest` as arguments. It calls `readTramNetwork()` to get the network. Inside the function, there are two nested functions: `convert_tram_nodes` which returns the first element of each node in the path, and `colors` which returns a color based on the node's position in the shortest and quickest paths. The main logic of `show_shortest` involves finding the quickest and shortest paths using `optimal_path_finder`, converting them to tram nodes, and then returning the quickest path cost, shortest path cost, and the color-coded paths.

```

46
47 def show_shortest(dep, dest):
48     network = readTramNetwork()
49
50     def convert_tram_nodes(optimal_path):
51         return [v[0] for v in optimal_path]
52
53     # find the quickest
54     quickest = optimal_path_finder(network, dep, dest)
55     # find the shortest
56     shortest = optimal_path_finder(network, dep, dest, geo_dist=True)
57     # get the quickest path description
58     quickest_path = convert_tram_nodes(quickest[0])
59     # get the shortest path description
60     shortest_path = convert_tram_nodes(shortest[0])
61     # get the quickest path cost
62     quickest_cost = quickest[1]
63     # get shortest path cost
64     shortest_cost = shortest[1]
65
66     def colors(v):
67         if v in shortest_path and v in quickest_path:
68             return 'cyan'
69         elif v in shortest_path:

```

1:1 CRLF UTF-8 4 spaces Python 3.10



This screenshot shows the continuation of the `show_shortest` function in `tramviz.py`. The `colors` function is completed, returning 'white' for nodes not in either path. The main logic continues by formatting the quickest and shortest paths into strings, calling `color_svg_network` with the color map, and finally returning the formatted paths.

```

61     # get the quickest path cost
62     quickest_cost = quickest[1]
63     # get shortest path cost
64     shortest_cost = shortest[1]
65
66     def colors(v):
67         if v in shortest_path and v in quickest_path:
68             return 'cyan'
69         elif v in shortest_path:
70             return 'green'
71         elif v in quickest_path:
72             return 'orange'
73         else:
74             return 'white'
75
76     timepath = 'Quickest: ' + ', '.join(quickest_path) + ' ./ ' + str(quickest_cost) + ' minutes'
77     geopath = 'Shortest: ' + ', '.join(shortest_path) + ' ./ ' + str(shortest_cost) + ' km'
78
79     color_svg_network(colormap=colors)
80
81     return timepath, geopath
82
83
84 ##### FOR TESTING PURPOSE #####

```

1:1 CRLF UTF-8 4 spaces Python 3.10