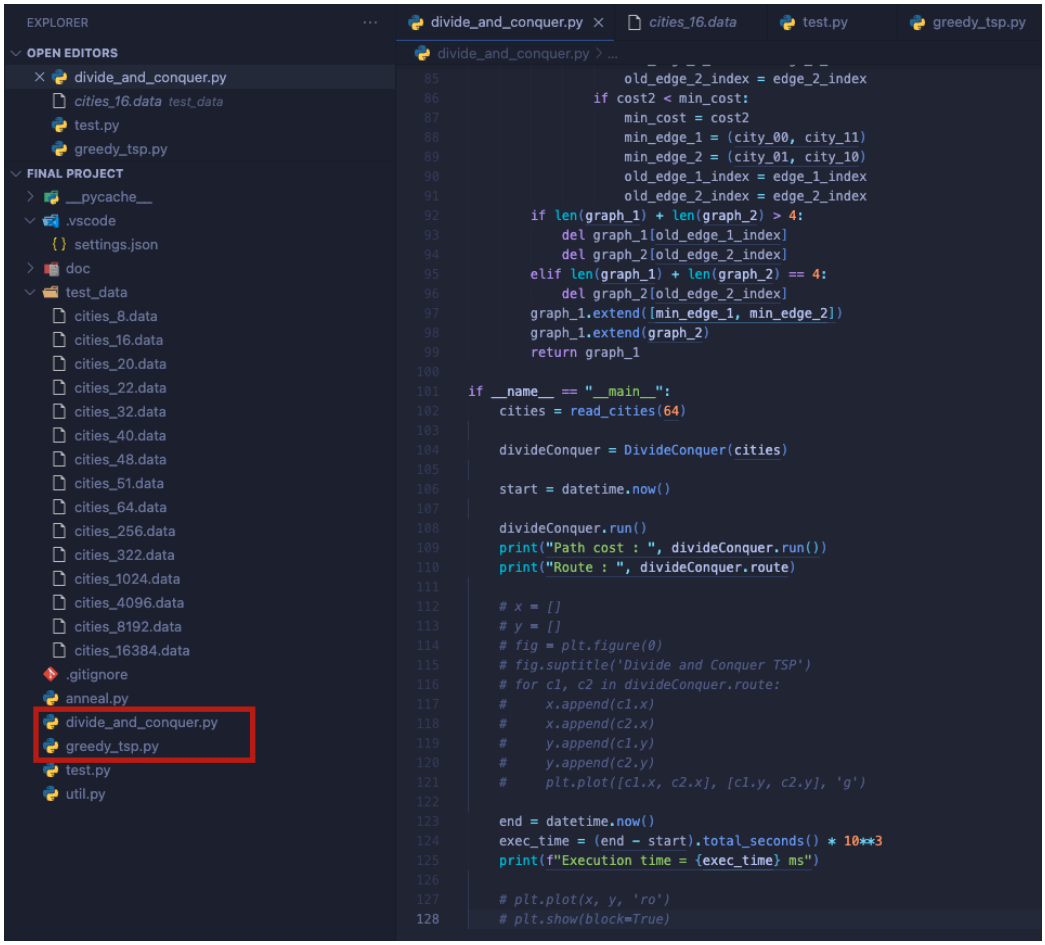
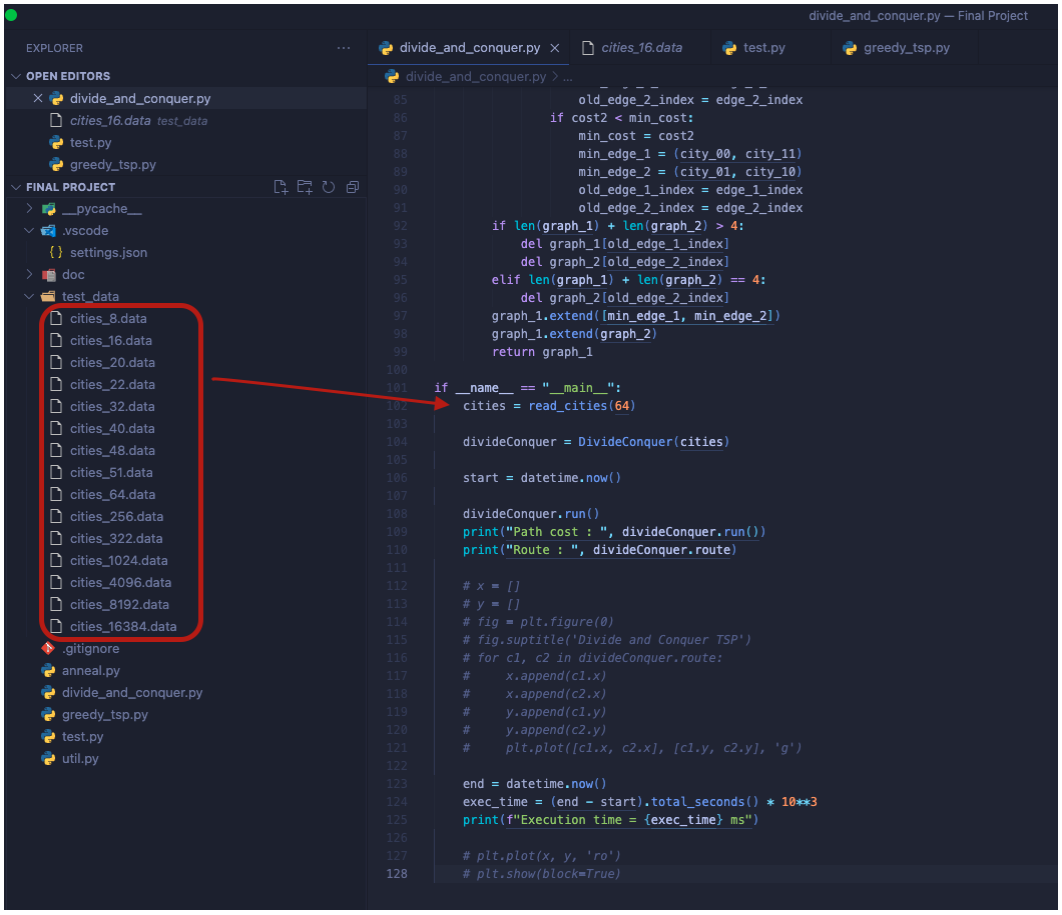
1. Running the Traveling Salesman Problem using 2 algorithms

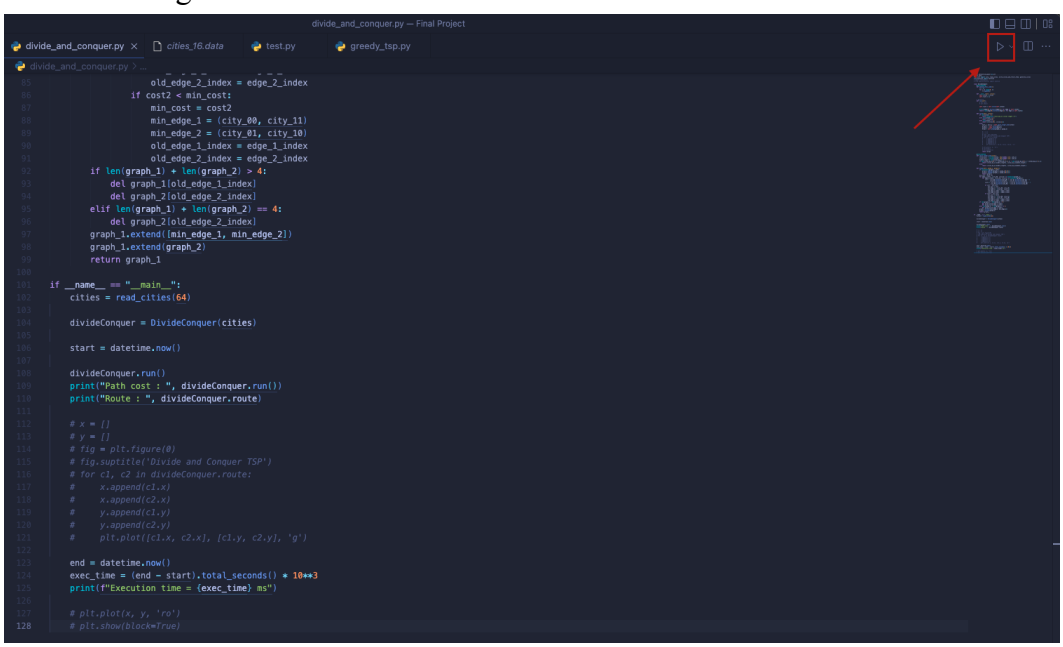
Step 1 : Pick 1 out of 2 algorithms you want to execute the Travelling salesman problem.



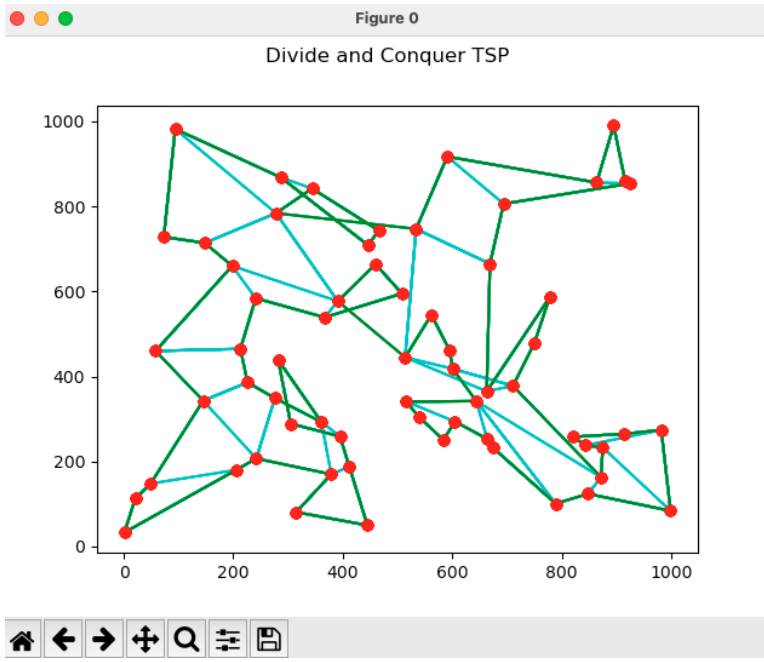
Step 2 : We provided the user with the number of cities they want to solve during the traveling salesman problem. Therefore, they can run it using both algorithms. In addition, to insert the city data using the template (“coordinate (space) coordinate”). The user is required to enter the coordinates of both the arrival and departure cities in each data entry for the algorithm to run smoothly without any interference.



Step 3 : After picking one algorithm and how many cities the user want to solve, we can execute the algorithm



Step 4: This window will pop off when the user executes the algorithm successfully.



In the terminal we also provide the user with the route and also the execution time to run the algorithm.

```
Route :  [((1.0, 33.0), (21.0, 113.0)), ((1.0, 33.0), (205.0, 180.0)), ((21.0, 113.0), (49.0, 148.0)), ((49.0, 148.0), (145.0, 343.0)), ((205.0, 180.0), (242.0, 207.0)), ((276.0, 349.0), (226.0
, 386.0)), ((242.0, 207.0), (379.0, 170.0)), ((276.0, 349.0), (361.0, 293.0)), ((445.0, 50.0), (315.0, 81.0)), ((445.0, 50.0), (412.0, 188.0)), ((315.0, 81.0), (379.0, 170.0)), ((412.0, 188.0),
 (395.0, 259.0)), ((395.0, 259.0), (305.0, 289.0)), ((305.0, 289.0), (282.0, 439.0)), ((361.0, 293.0), (282.0, 439.0)), ((145.0, 343.0), (58.0, 460.0)), ((226.0, 386.0), (213.0, 465.0)), ((58.0
, 460.0), (198.0, 660.0)), ((213.0, 465.0), (241.0, 584.0)), ((241.0, 584.0), (367.0, 539.0)), ((367.0, 539.0), (508.0, 595.0)), ((391.0, 577.0), (460.0, 664.0)), ((460.0, 664.0), (508.0, 595.0
)), ((198.0, 660.0), (148.0, 714.0)), ((148.0, 714.0), (73.0, 728.0)), ((73.0, 728.0), (94.0, 982.0)), ((278.0, 784.0), (344.0, 842.0)), ((94.0, 982.0), (287.0, 868.0)), ((287.0, 868.0), (446.0
, 709.0)), ((344.0, 842.0), (466.0, 743.0)), ((446.0, 709.0), (466.0, 743.0)), ((391.0, 577.0), (514.0, 445.0)), ((278.0, 784.0), (534.0, 747.0)), ((584.0, 251.0), (605.0, 294.0)), ((584.0, 251
.0), (539.0, 305.0)), ((539.0, 305.0), (516.0, 341.0)), ((605.0, 294.0), (664.0, 253.0)), ((516.0, 341.0), (644.0, 342.0)), ((789.0, 101.0), (674.0, 233.0)), ((674.0, 233.0), (664.0, 253.0)), (
(789.0, 101.0), (848.0, 124.0)), ((848.0, 124.0), (999.0, 84.0)), ((872.0, 163.0), (875.0, 233.0)), ((875.0, 233.0), (843.0, 238.0)), ((999.0, 84.0), (982.0, 274.0)), ((821.0, 258.0), (843.0, 2
38.0)), ((821.0, 258.0), (913.0, 265.0)), ((913.0, 265.0), (982.0, 274.0)), ((644.0, 342.0), (601.0, 418.0)), ((872.0, 163.0), (711.0, 378.0)), ((601.0, 418.0), (595.0, 462.0)), ((514.0, 445.0)
, (562.0, 545.0)), ((595.0, 462.0), (562.0, 545.0)), ((663.0, 365.0), (778.0, 586.0)), ((711.0, 378.0), (750.0, 478.0)), ((750.0, 478.0), (778.0, 586.0)), ((663.0, 365.0), (669.0, 665.0)), ((66
9.0, 665.0), (694.0, 806.0)), ((534.0, 747.0), (591.0, 917.0)), ((694.0, 806.0), (926.0, 854.0)), ((591.0, 917.0), (864.0, 856.0)), ((926.0, 854.0), (916.0, 860.0)), ((864.0, 856.0), (895.0, 99
0.0)), ((916.0, 860.0), (895.0, 990.0))]
Execution time = 7930.929 ms
```