Kamran Muhammad Bhatti

3807942

Discrete Mathematics

Contents

[ProblemStatement 2](#_Toc27995379)

[What can I deduce from the problem? 3](#_Toc27995380)

[How will I approach the problem? 4](#_Toc27995381)

[LPP 4](#_Toc27995382)

[Program(lpp.py) 5](#_Toc27995383)

[LC (1,2) 9](#_Toc27995384)

[LC (1,3) 10](#_Toc27995385)

[LC (2,3) 11](#_Toc27995386)

[LC (2,4) 12](#_Toc27995387)

[LC (3,4) 13](#_Toc27995388)

[LC (3,5) 14](#_Toc27995389)

[LC (4,5) 15](#_Toc27995390)

[LC (4,6) 16](#_Toc27995391)

[LC (5,6) 17](#_Toc27995392)

[Conclusion 18](#_Toc27995393)

[Graph Theory 19](#_Toc27995394)

[Dijkstra’s algo.py 19](#_Toc27995395)

[Paths 19](#_Toc27995396)

[Conclusion 22](#_Toc27995397)

[Statistics 23](#_Toc27995398)

[Basic Information.py 23](#_Toc27995399)

[PDF.py 25](#_Toc27995400)

[Path approach 25](#_Toc27995401)

[What I can conclude? 27](#_Toc27995402)

[Cartesian Product Approach 28](#_Toc27995403)

[Basic Information v2.py 32](#_Toc27995404)

[Path PDF.py 34](#_Toc27995405)

[Undirected graph 35](#_Toc27995406)

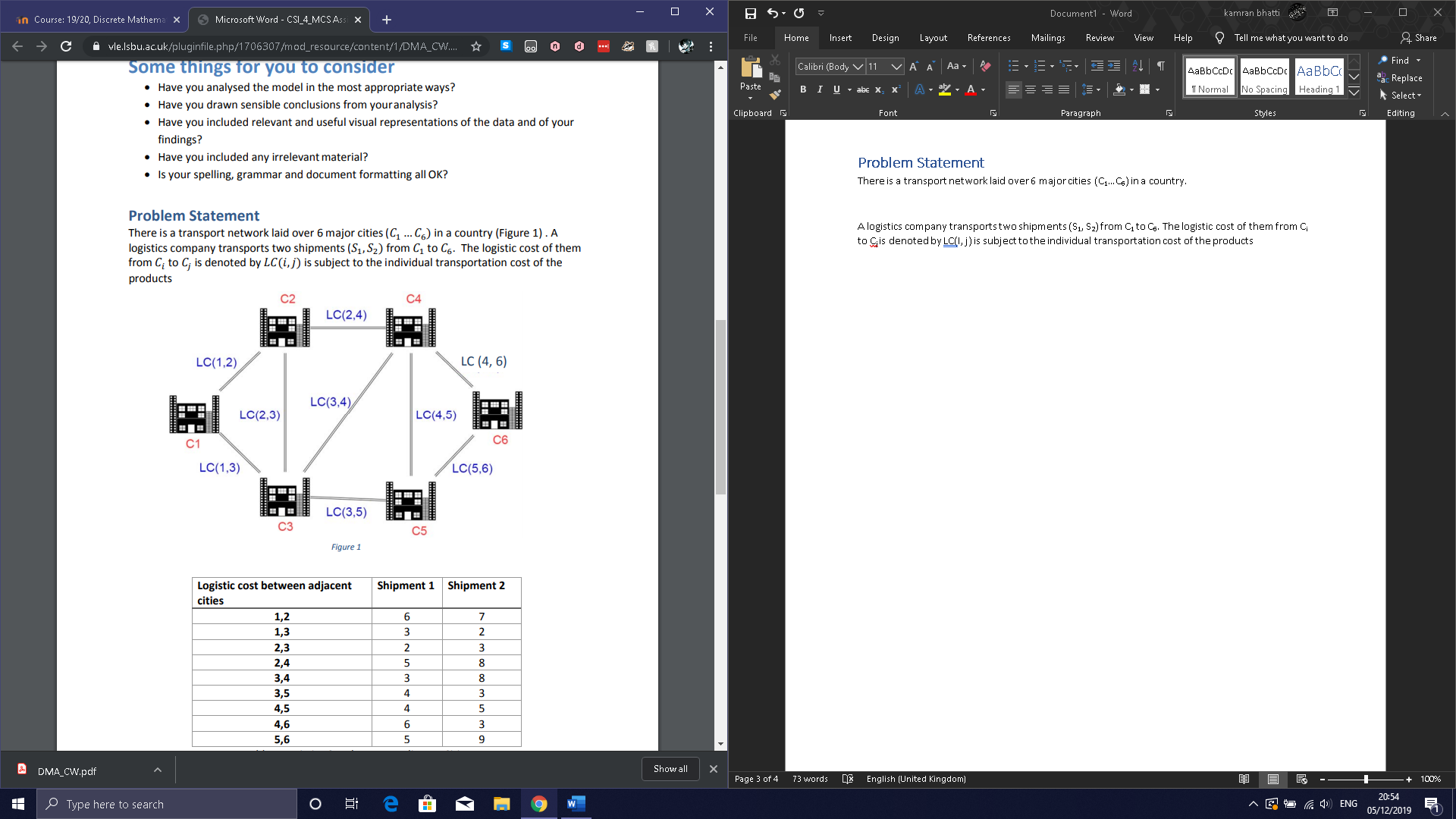
[Directed graph 36](#_Toc27995407)

[Conclusion 37](#_Toc27995408)

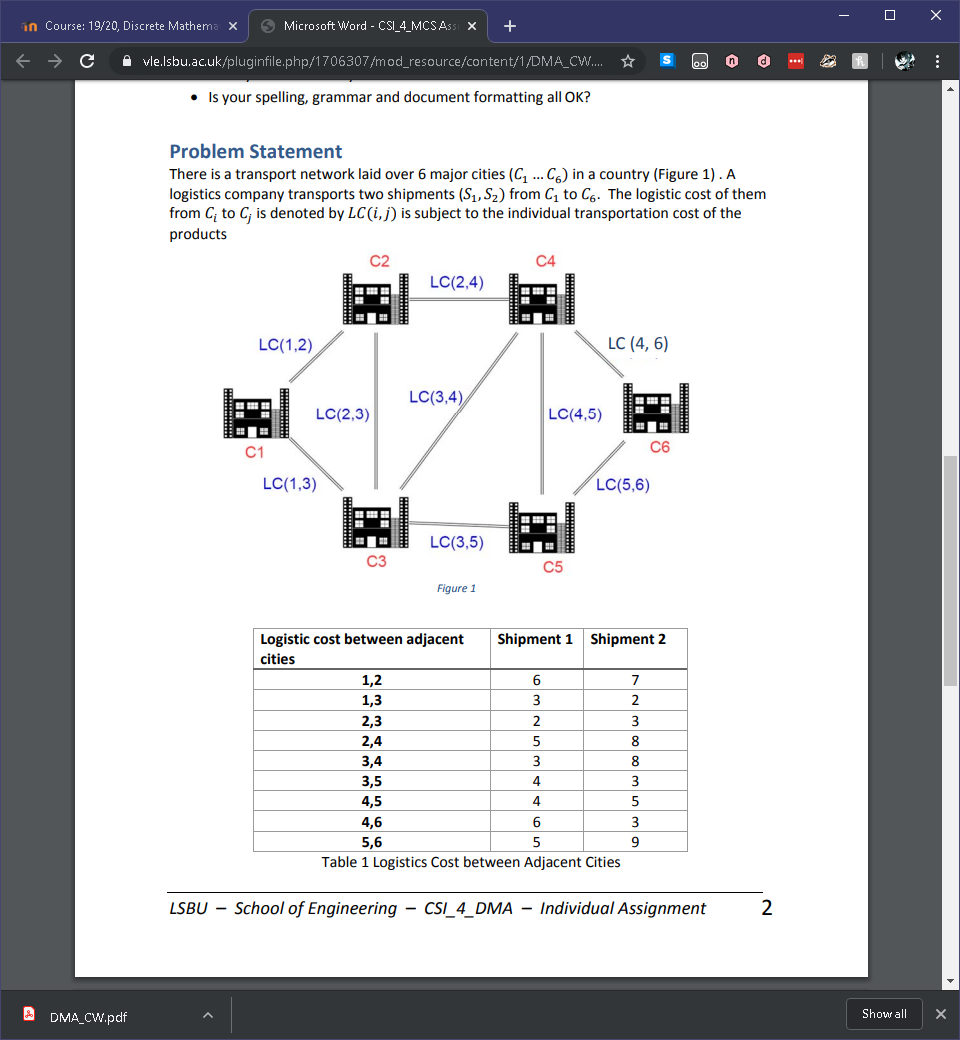
[Bibliography 38](#_Toc27995409)

# ProblemStatement

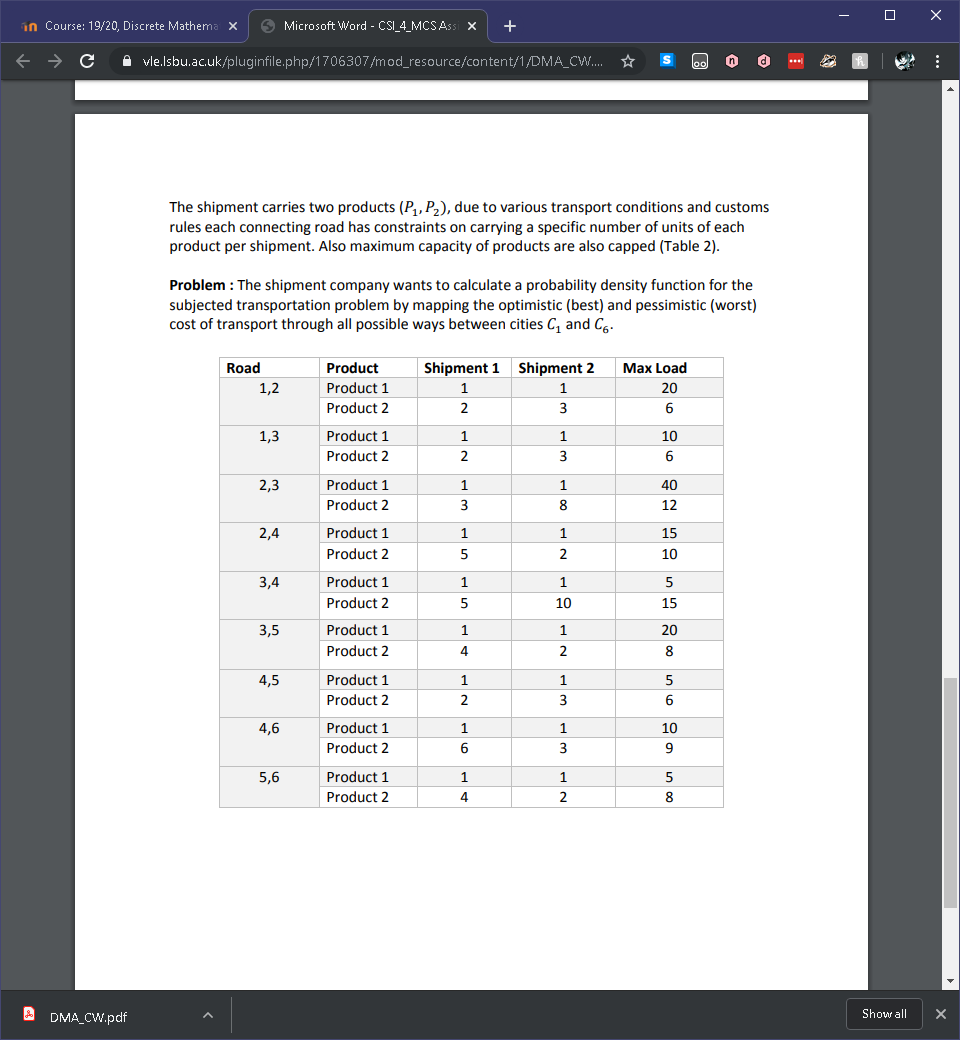
There is a transport network laid over 6 major cities (C1…C6) in a country.



A logistics company transports two shipments (S1, S2) from C1 to C6. The logistic cost of them from Ci to Cj is denoted by LC (I, j) is subject to the individual transportation cost of the products:



The shipment carries two products (P1, P2), due to various transport conditions and customs rules each connecting road has constraints on carrying a specific number of units of each product per shipment. Also, maximum capacity of products is also capped:



**Problem:** The shipment company wants to calculate a probability density function for the subjected transportation problem by mapping the optimistic (best) and pessimistic (worst) cost of transport through all possible ways between C1 and C6.

## What can I deduce from the problem?

Each city and each connection between cities correspond to vertices and edges of a graph respectively.

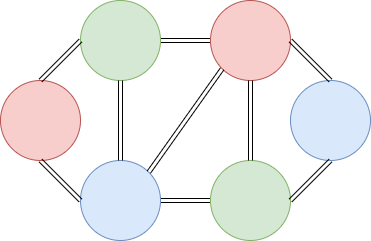
**Characteristics**

The graph G(V,E) has vertex set V ={(C1, C2, C3, C4, C5, C6} and edge set E = {LC(1,2), LC(1,3), LC(2,3), LC(2,4), LC(3,4), LC(3,5), LC(4,5), LC(4,6), LC(5,6)}. The vertices with the largest degree would be C3 and C4 with degree of 4. Additionally, the graph has a Hamiltonian circuit in which you can traverse every vertex exactly once before reaching the starting vertex. Furthermore, I can determine that the graph is planar as none of the edges overlap or cross.

The graph has subgraphs of C3 or K3 between nodes: C1, C2, C3 or C2, C3, C4 or C3, C4, C5 or C4, C5, C6. Additionally, the subgraph C4 can be seen in nodes: C1, C2, C3, C4 or C2, C3, C4, C5 or C3, C4, C5, C6. Finally, the largest subgraph that can be seen is C6 which contains all nodes when following the outer edges.

**Chromatic Number**

The subgraphs of C4 and C6 infer that the graph will have a minimum chromatic number of 2 but as the graph contains a subgraph of C3, the graph will have a minimum chromatic number of 3 which is valid for the given graph. Therefore χ(Cn) when n is even makes the chromatic number 2, when n is odd the chromatic number is 3. χ(G) is 3.



**Rank Nullity Theorem**

The rank of the graph is 6 nodes – 1 component = 5. The nullity of the graph is 5 regions – 6 nodes + 1 components = 0.

|  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  | E1 | E2 | E3 | E4 | E5 | E6 | E7 | E8 | E9 |
| C1 | 1 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| C2 | 1 | 0 | 1 | 1 | 0 | 0 | 0 | 0 | 0 |
| C3 | 0 | 1 | 1 | 0 | 1 | 1 | 0 | 0 | 0 |
| C4 | 0 | 0 | 0 | 1 | 1 | 0 | 1 | 1 | 0 |
| C5 | 0 | 0 | 0 | 0 | 0 | 1 | 1 | 0 | 1 |
| C6 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 1 |

**Adjacency matrix Incidence matrix**

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
|  | C1 | C2 | C3 | C4 | C5 | C6 |
| C1 | 0 | 1 | 1 | 0 | 0 | 0 |
| C2 | 1 | 0 | 1 | 1 | 0 | 0 |
| C3 | 1 | 1 | 0 | 1 | 1 | 0 |
| C4 | 0 | 1 | 1 | 0 | 1 | 1 |
| C5 | 0 | 0 | 1 | 1 | 0 | 1 |
| C6 | 0 | 0 | 0 | 1 | 1 | 0 |

## How will I approach the problem?

The logistic cost between adjacent cities illustrates a problem which can be represented with the use of linear programming in the form of a minimization problem in order to keep the cost of travelling between cities low. However, the condition of the problem is to find the optimistic and pessimistic cost of transport therefore I will be finding the maximization and minimization of the cost of transport between all edges in the graph. The decision variables for both problems will be shipment 1 and shipment 2. The constraints for both problems will be the products on each of the edges between the towns.

Finally, to find the definitive answer to the maximization and minimization problems, I will have to use graph theory in order to find all paths from C1 to C6. The optimistic and pessimistic cost of transport will be determined by evaluating the cost of each path from C1 to C6. In regards to the probability density function, there will be a minimum of 2 distribution figures to depict the minimization of all the edges and the maximization of all the edges when depicting the graph for all of the paths in the graph.

Furthermore, I will use cartesian product for each of the paths to find the cost of all possible scenarios of the transportation problem, then I will make a probability density function for the undirected and directed graph as I have done just for the maximization of the paths. All of the code I will be using will be uploaded to my GitHub page with the appropriate csv files associated with the programs I have written, (Bhatti, 2019)

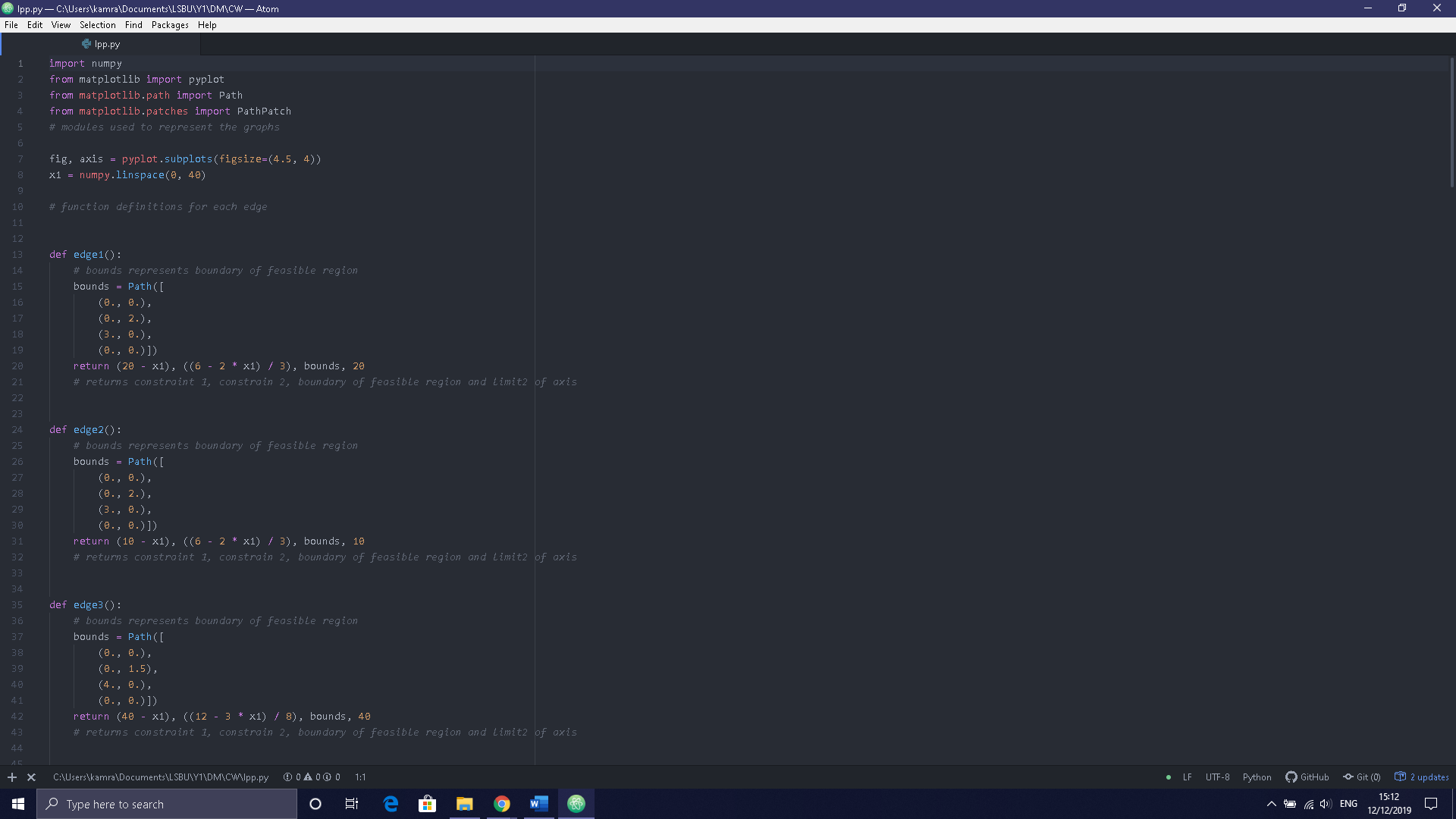
# LPP

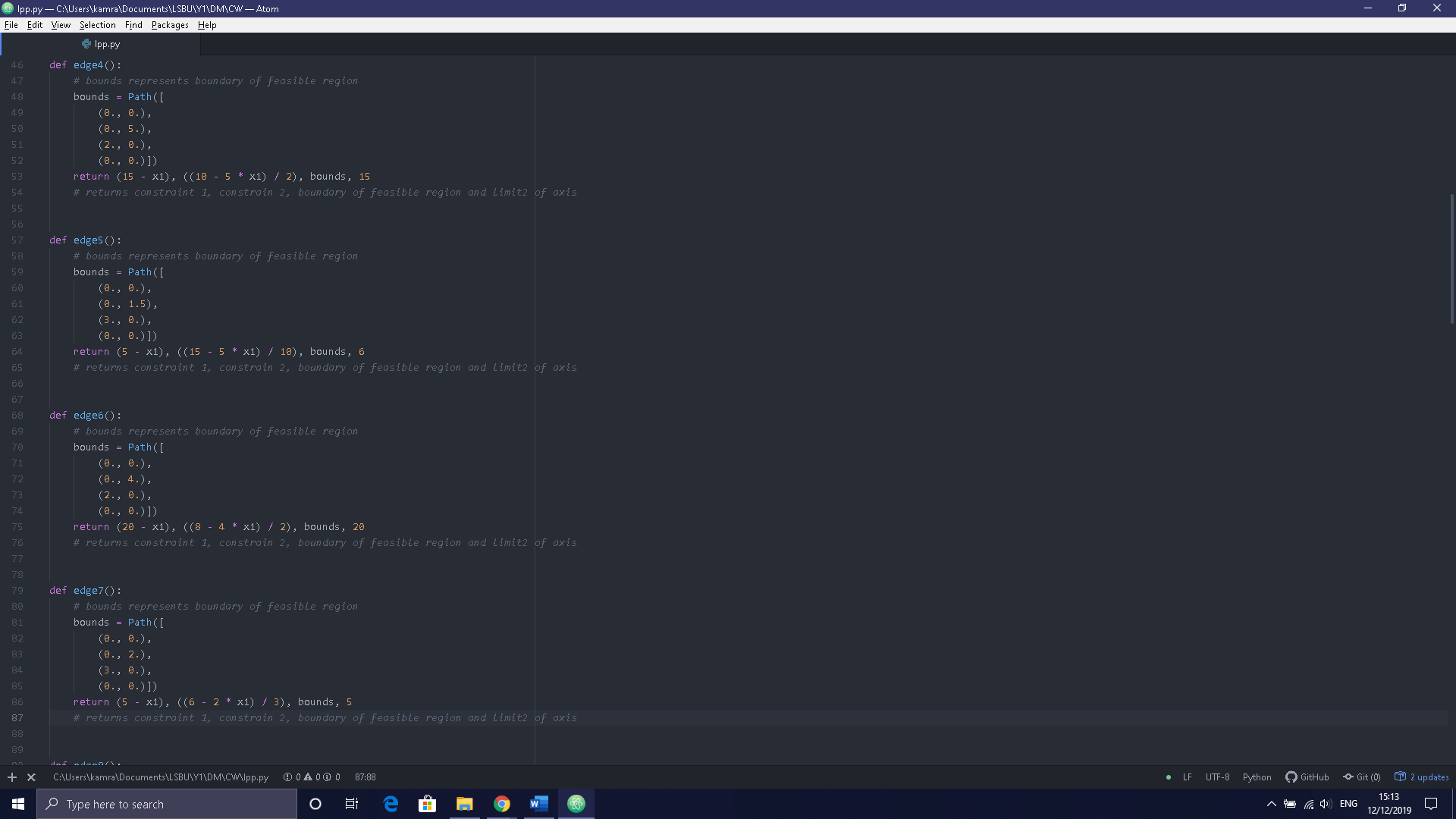
In this section I will carry out the linear programming problem for each of the individual edges. S1 is shipment 1 represented as X and S2 is shipment 2 represented as Y. The first two rows after the objective function for each of the edges will represent product 1 and product 2 being P1 and P2 respectively.

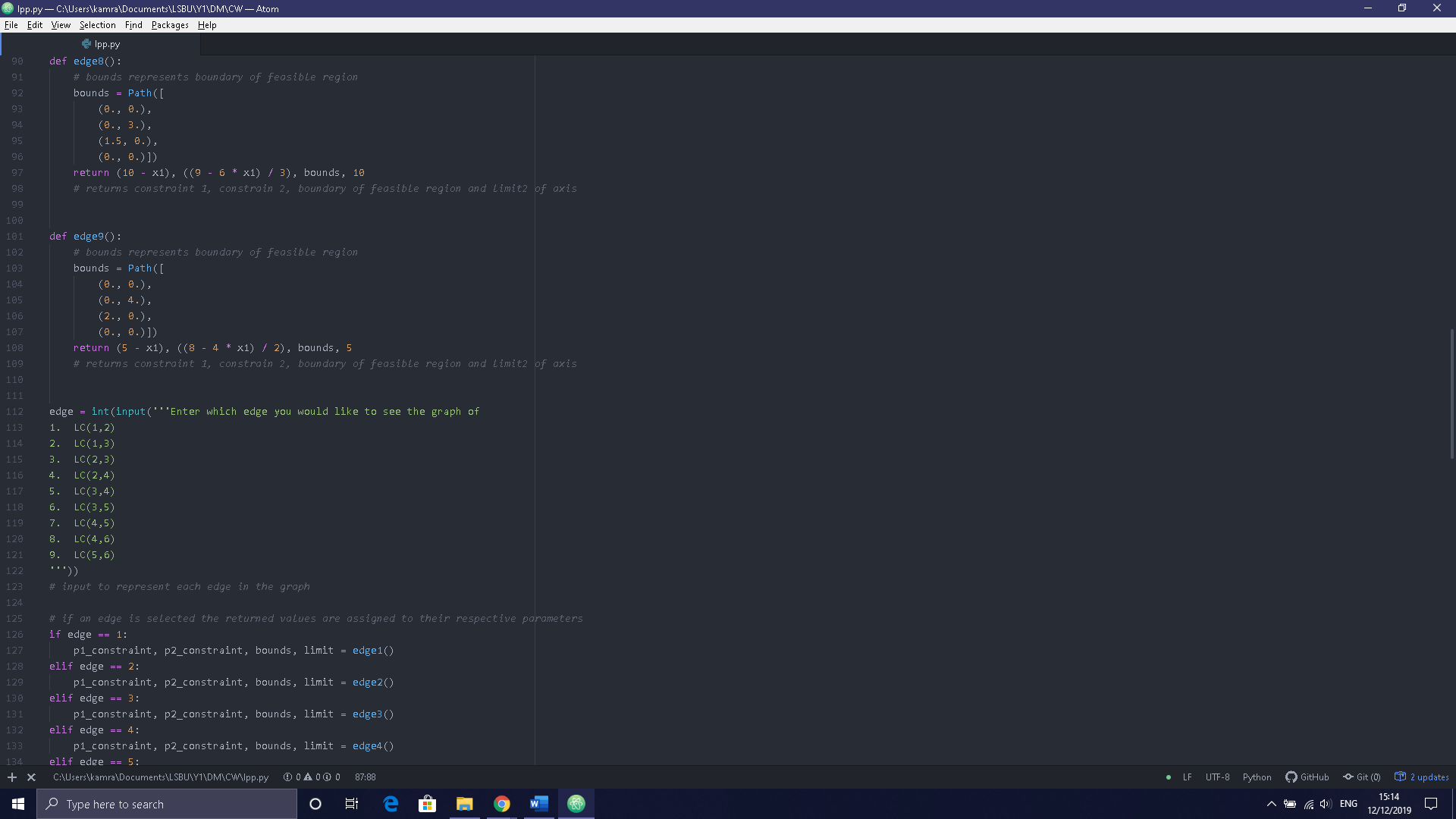
In this situation maximization of the linear programming problems will show the most pessimistic approach by giving the highest cost each edge for the journey, in comparison minimization of the problems will depict an optimistic approach by giving the lowest cost for each edge on the journey.

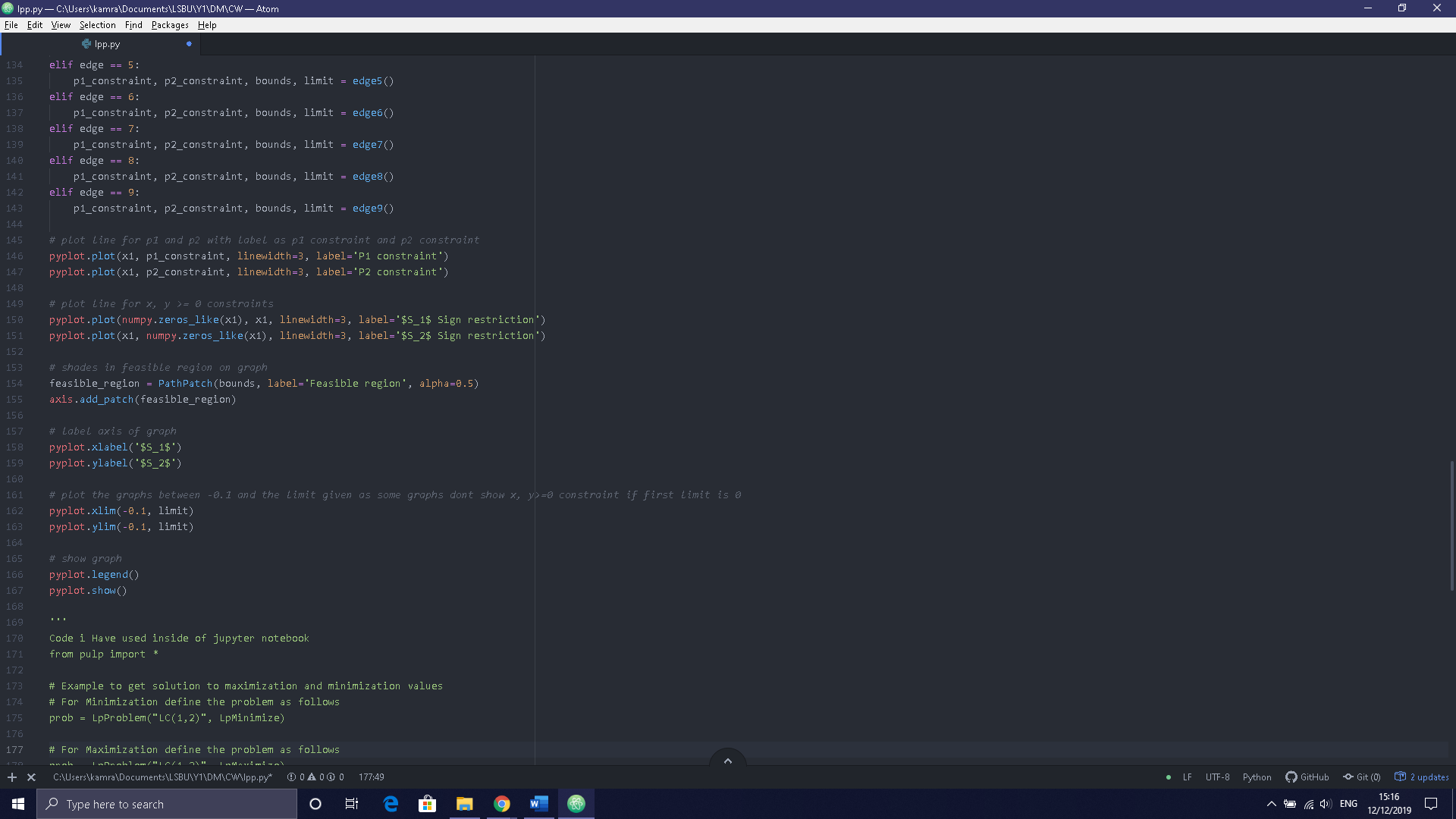
**General form**

## Program(lpp.py)







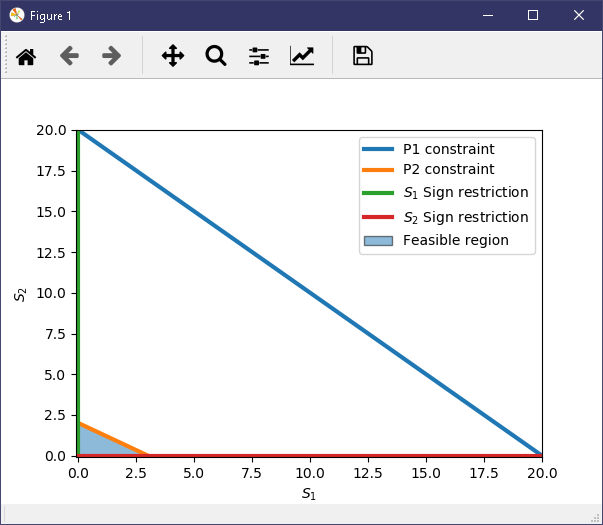


To implement the linear programming problem in python I used code from (Wang, 2019).

## LC (1,2)

|  |  |  |
| --- | --- | --- |
|  | **Argument matrix**  **=** | **Intercepting Form** |

##### Solution



|  |  |
| --- | --- |
| **Maximization** | **Minimization** |
|  |  |

|  |  |  |
| --- | --- | --- |
| S1 | S2 | Z |
| 0 | 0 | 0 |
| 0 | 2 | 14 |
| 3 | 0 | 18 |

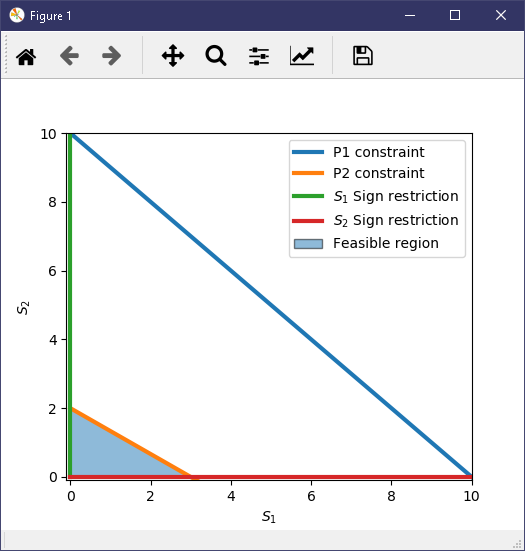
The maximum value is when S1 is 3 and S2 is 0 making Z = 18.

The minimum value is when S1 and S2 are both 0 making Z = 0.

## LC (1,3)

|  |  |  |
| --- | --- | --- |
|  | **Argument matrix**  **=** | **Intercepting Form** |

##### Solution



|  |  |
| --- | --- |
| **Maximization** | **Minimization** |
|  |  |

|  |  |  |
| --- | --- | --- |
| S1 | S2 | Z |
| 0 | 0 | 0 |
| 0 | 2 | 4 |
| 3 | 0 | 9 |

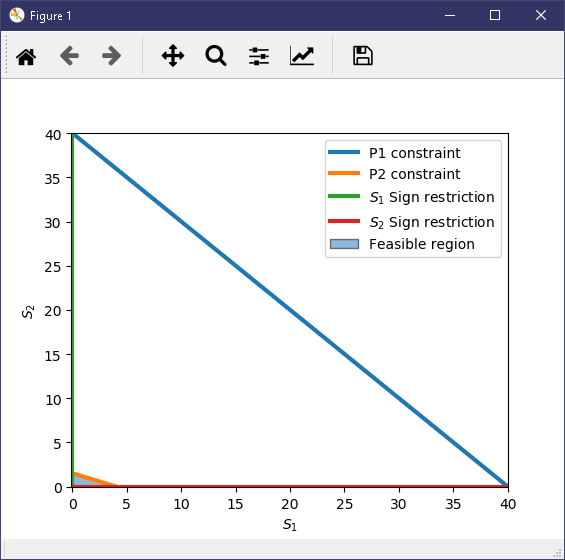
The maximum value is when S1 is 3 and S2 is 0 making Z = 9.

The minimum value is when S1 and S2 are both 0 making Z = 0.

## LC (2,3)

|  |  |  |
| --- | --- | --- |
|  | **Argument matrix**  **=** | **Intercepting Form** |

##### Solution



|  |  |
| --- | --- |
| **Maximization** | **Minimization** |
|  |  |

|  |  |  |
| --- | --- | --- |
| S1 | S2 | Z |
| 0 | 0 | 0 |
| 0 | 1.5 | 4.5 |
| 4 | 0 | 8 |

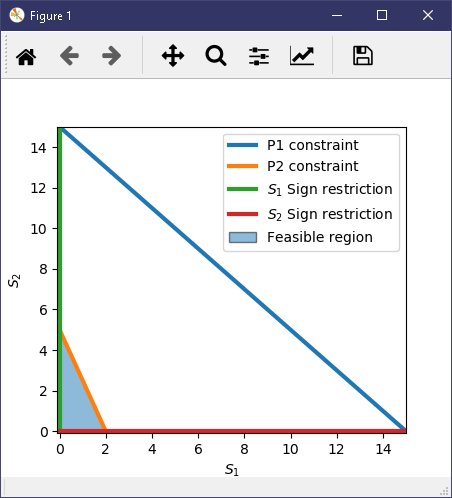
The maximum value is when S1 is 4 and S2 is 0 making Z = 8.

The minimum value is when S1 and S2 are both 0 making Z = 0.

## LC (2,4)

|  |  |  |
| --- | --- | --- |
|  | **Argument matrix**  **=** | **Intercepting Form** |

##### Solution



|  |  |
| --- | --- |
| **Maximization** | **Minimization** |
|  |  |

|  |  |  |
| --- | --- | --- |
| S1 | S2 | Z |
| 0 | 0 | 0 |
| 0 | 5 | 40 |
| 2 | 0 | 10 |

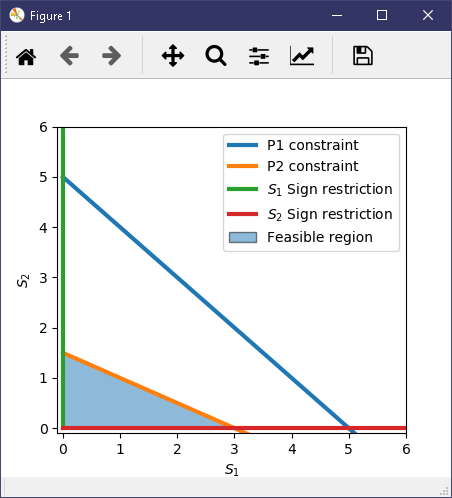
The maximum value is when S1 is 0 and S2 is 5 making Z = 40.

The minimum value is when S1 and S2 are both 0 making Z = 0.

## LC (3,4)

|  |  |  |
| --- | --- | --- |
|  | **Argument matrix**  **=** | **Intercepting Form** |

##### Solution



|  |  |
| --- | --- |
| **Maximization** | **Minimization** |
|  |  |

|  |  |  |
| --- | --- | --- |
| S1 | S2 | Z |
| 0 | 0 | 0 |
| 0 | 1.5 | 12 |
| 3 | 0 | 9 |

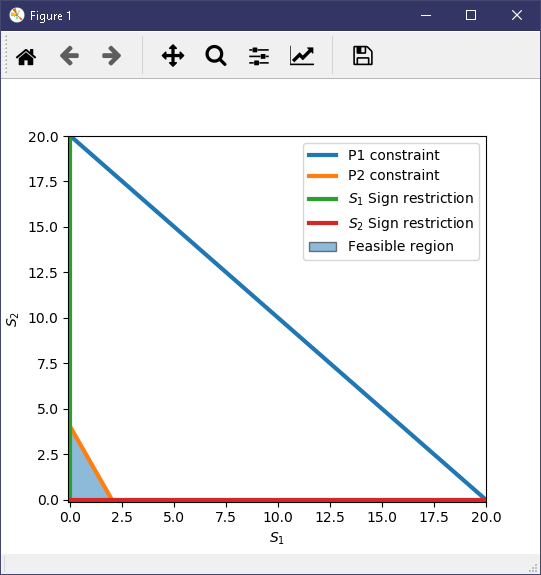
The maximum value is when S1 is 0 and S2 is 1.5 making Z = 12.

The minimum value is when S1 and S2 are both 0 making Z = 0.

## LC (3,5)

|  |  |  |
| --- | --- | --- |
|  | **Argument matrix**  **=** | **Intercepting Form** |

##### Solution



|  |  |
| --- | --- |
| **Maximization** | **Minimization** |
|  |  |

|  |  |  |
| --- | --- | --- |
| S1 | S2 | Z |
| 0 | 0 | 0 |
| 0 | 4 | 12 |
| 2 | 0 | 8 |

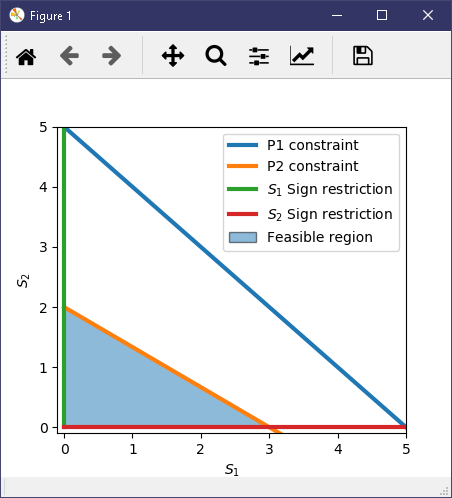
The maximum value is when S1 is 0 and S2 is 4 making Z = 12.

The minimum value is when S1 and S2 are both 0 making Z = 0.

## LC (4,5)

|  |  |  |
| --- | --- | --- |
|  | **Argument matrix**  **=** | **Intercepting Form** |

##### Solution



|  |  |
| --- | --- |
| **Maximization** | **Minimization** |
|  |  |

|  |  |  |
| --- | --- | --- |
| S1 | S2 | Z |
| 0 | 0 | 0 |
| 0 | 2 | 10 |
| 3 | 0 | 12 |

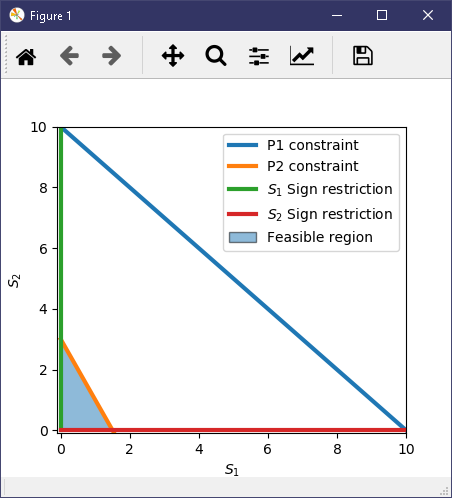
The maximum value is when S1 is 3 and S2 is 0 making Z = 12.

The minimum value is when S1 and S2 are both 0 making Z = 0.

## LC (4,6)

|  |  |  |
| --- | --- | --- |
|  | **Argument matrix**  **=** | **Intercepting Form** |

##### Solution



|  |  |
| --- | --- |
| **Maximization** | **Minimization** |
|  |  |

|  |  |  |
| --- | --- | --- |
| S1 | S2 | Z |
| 0 | 0 | 0 |
| 0 | 3 | 9 |
| 1.5 | 0 | 9 |

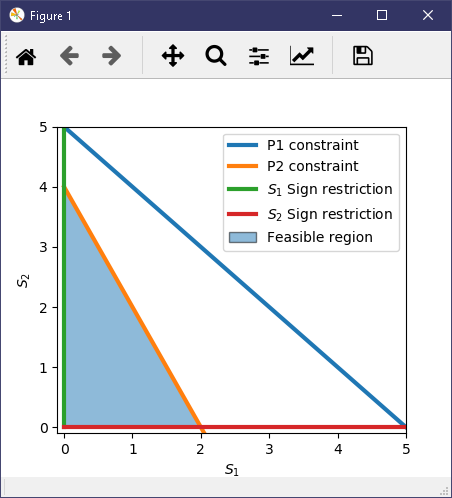
The maximum value is when S1 is 0 and S2 is 3 / S1 is 1.5 and S2 is 0 making Z = 9.

The minimum value is when S1 and S2 are both 0 making Z = 0.

## LC (5,6)

|  |  |  |
| --- | --- | --- |
|  | **Argument matrix**  **=** | **Intercepting Form** |

##### Solution



|  |  |
| --- | --- |
| **Maximization** | **Minimization** |
|  |  |

|  |  |  |
| --- | --- | --- |
| S1 | S2 | Z |
| 0 | 0 | 0 |
| 0 | 4 | 36 |
| 2 | 0 | 10 |

The maximum value is when S1 is 0 and S2 is 4 making Z = 36.

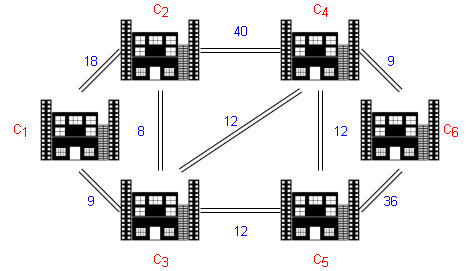
The minimum value is when S1 and S2 are both 0 making Z = 0.

## Conclusion

From drawing the graphs for each edge, I have evaluated that none of the graphs intersect when the constraint is used. Additionally, all of the feasible solutions are members of the feasible regions for each individual edges’ graph thus, a set of each edges’ feasible solutions would be a proper subset of each edges’ feasible region. If Fs represents feasible solutions and Fr represents the feasible region, and .

Furthermore, as all of the graphs for each edge have a minimum value of 0, depicting the transportation problem with 0 values on each edge wouldn’t help in solving the problem and calculating a probability density function. Therefore, I will only be using the maximum values of each edge on the transportation graph.

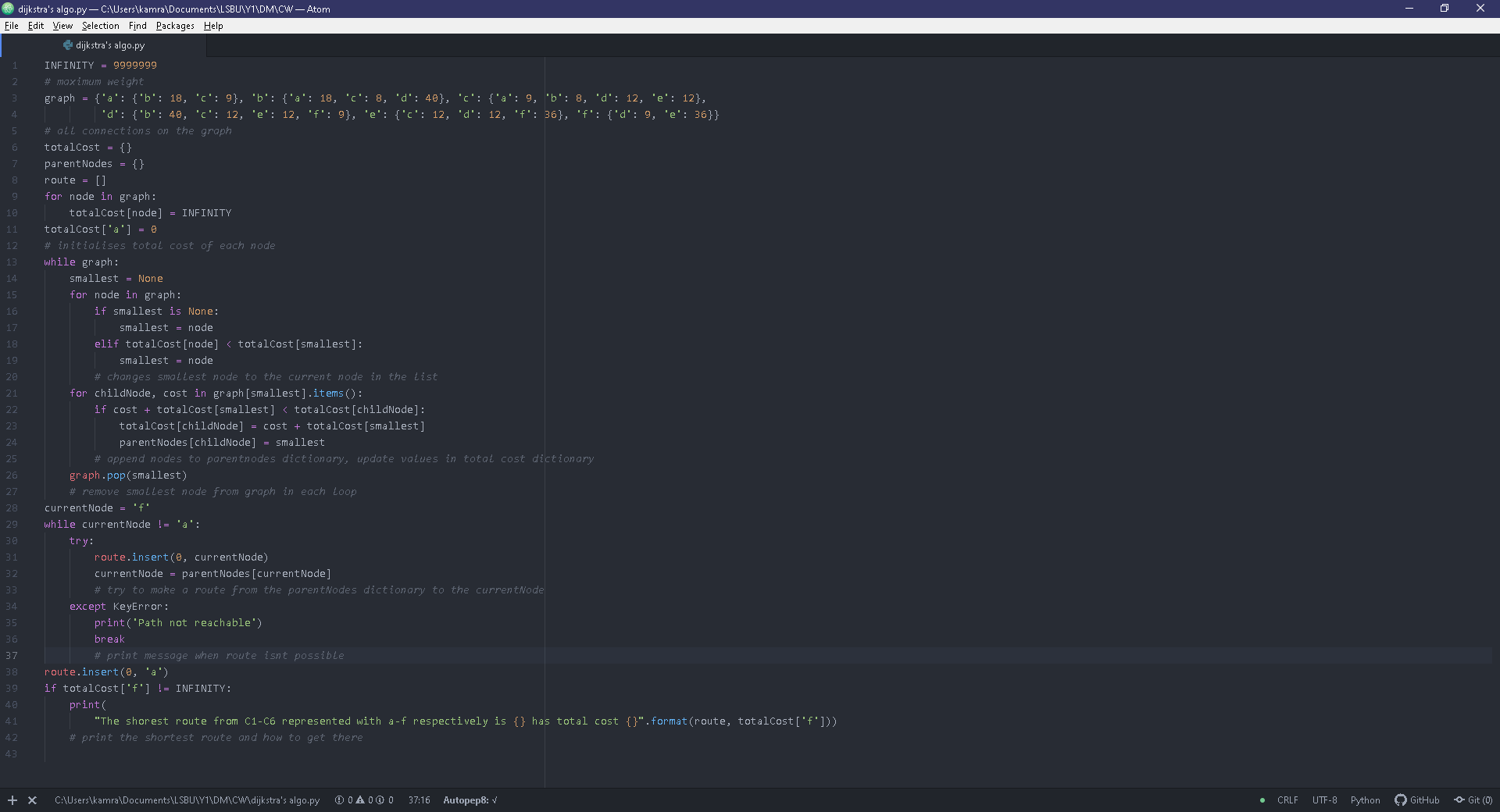
**Graph**



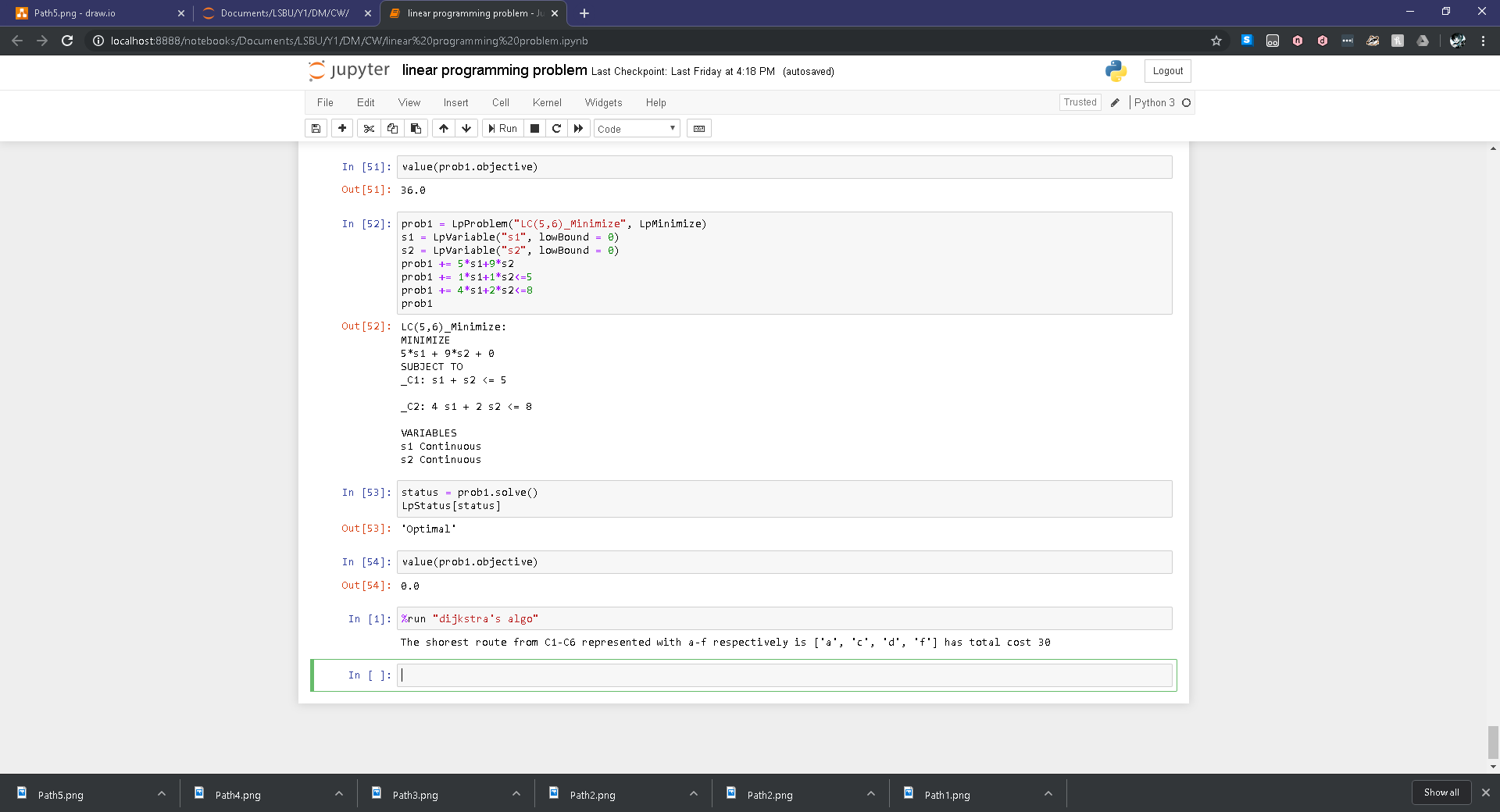
# Graph Theory

In this section I will be determining all the paths that can be found in the transportation graph and the total cost it takes to travel along each path.

## Dijkstra’s algo.py



I used source code from (Sullivan, 2017) to implement Dijkstra’s algorithm in python.



Running this algorithm shows that the shortest path would be from A, C, D, F giving a cost of 30 which would be from C1, C3, C4 and finally C6.

## Paths

|  |  |  |
| --- | --- | --- |
| **Number** | | **Cost** |
| 1 |  | 67 |
| 2 |  | 106 |
| 3 |  | 47 |
| 4 |  | 86 |
| 5 |  | 59 |
| 6 |  | 74 |
| 7 |  | 66 |
| 8 |  | 105 |
| 9 |  | 30 |
| 10 |  | 69 |
| 11 |  | 42 |
| 12 |  | 57 |

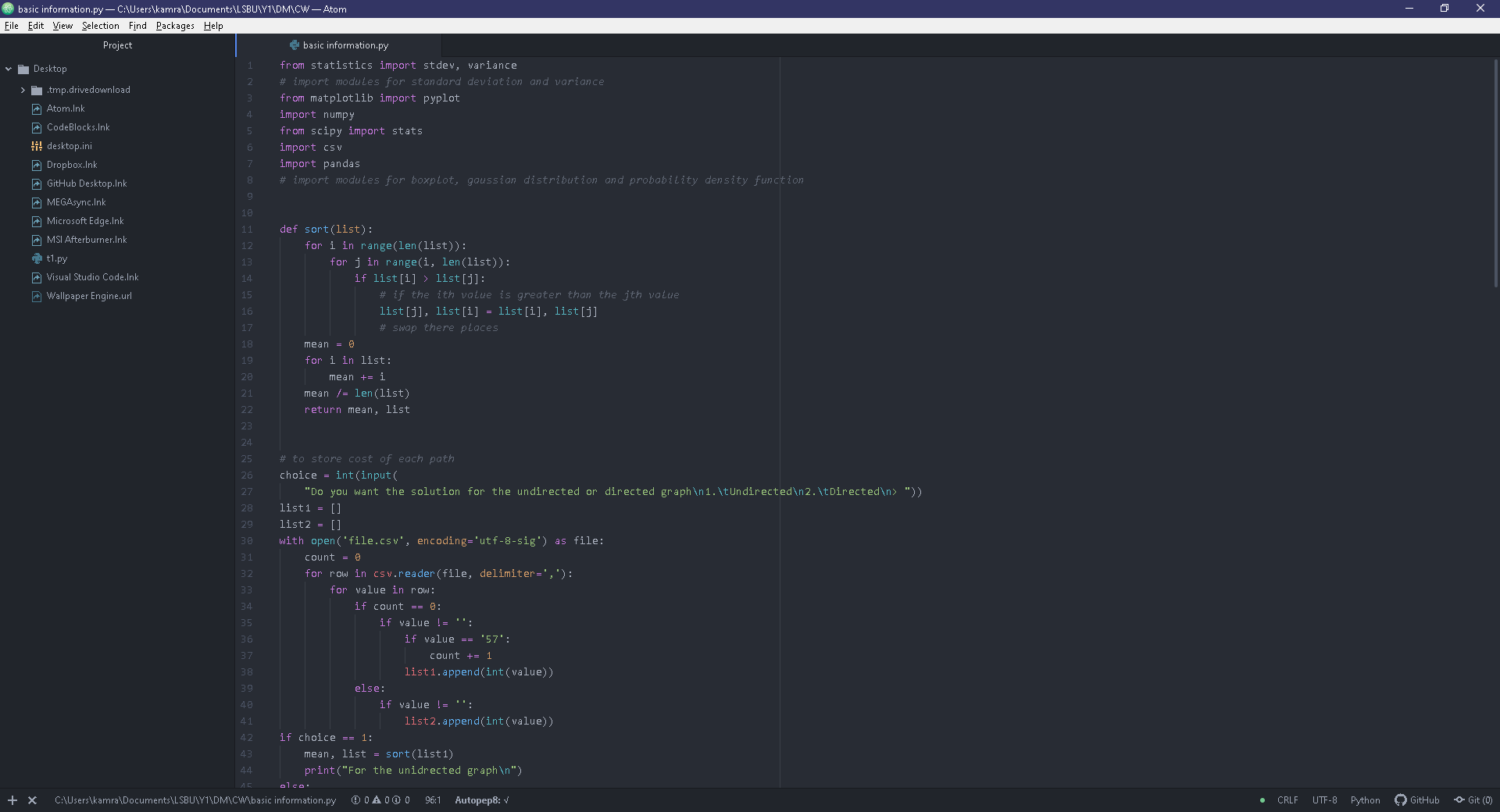
## Conclusion

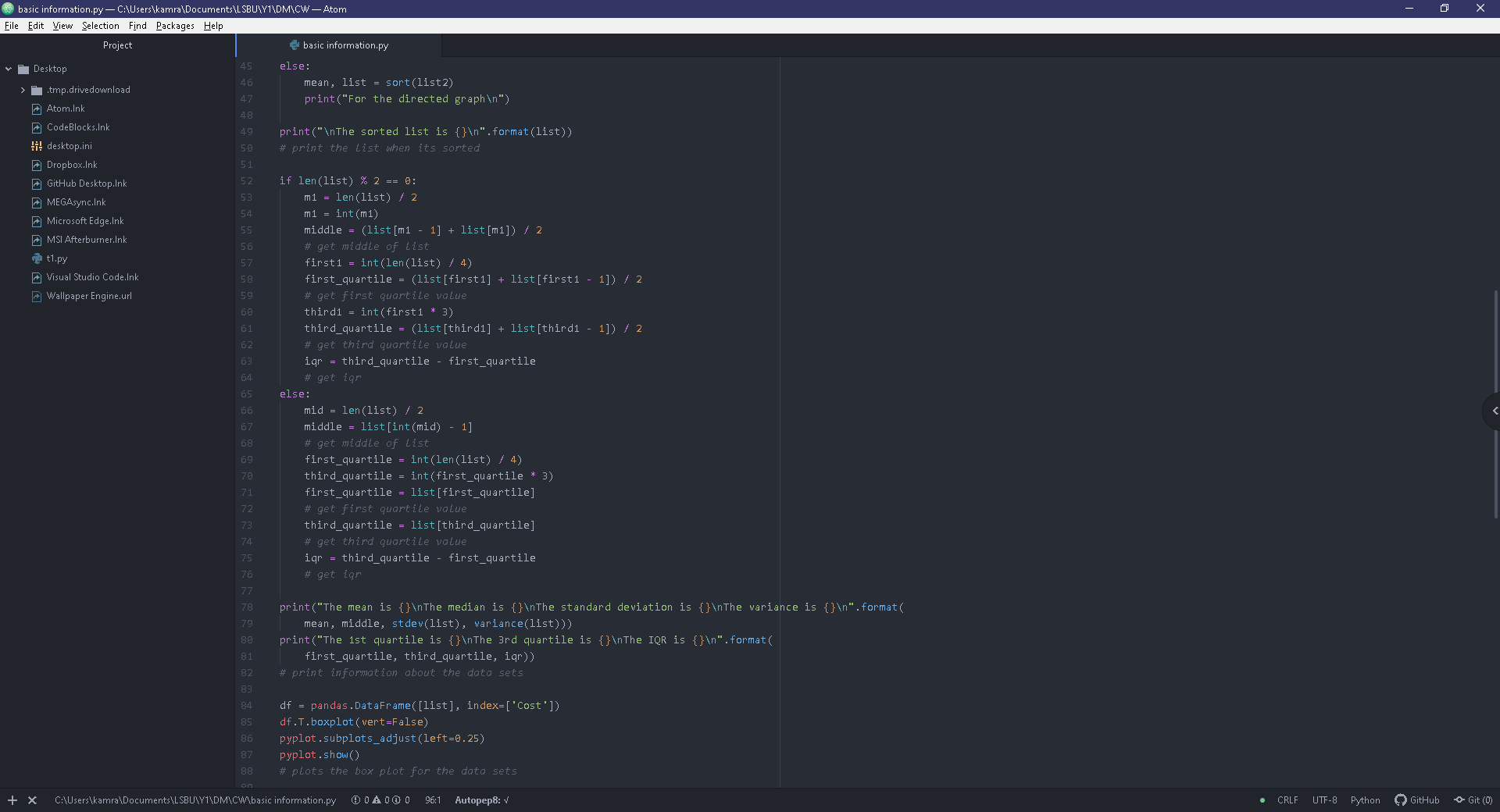
All of the paths shown have a vertex set which is a proper subset of the vertex set of the transportation graph and this can be similarly linked to the edge set of the paths with the edge set of the transportation graph thus making all of the vertices and edges members of the total vertex and edge sets of the transportation graph. This can be depicted as and and and where V represents vertices and E represents edges. Paths 4, 5 and 8 are minimal spanning trees as they traverse the tree whilst visiting every node thus these paths are Hamiltonian paths. Each of the paths are subgraphs of the transportation graph such that they can be represented as components of the transportation graph. From the graphs I can deduce that the optimum path is C1, C3, C4, C6 with a cost of 30 which is validated by the Dijkstra’s algorithm I implemented in python.

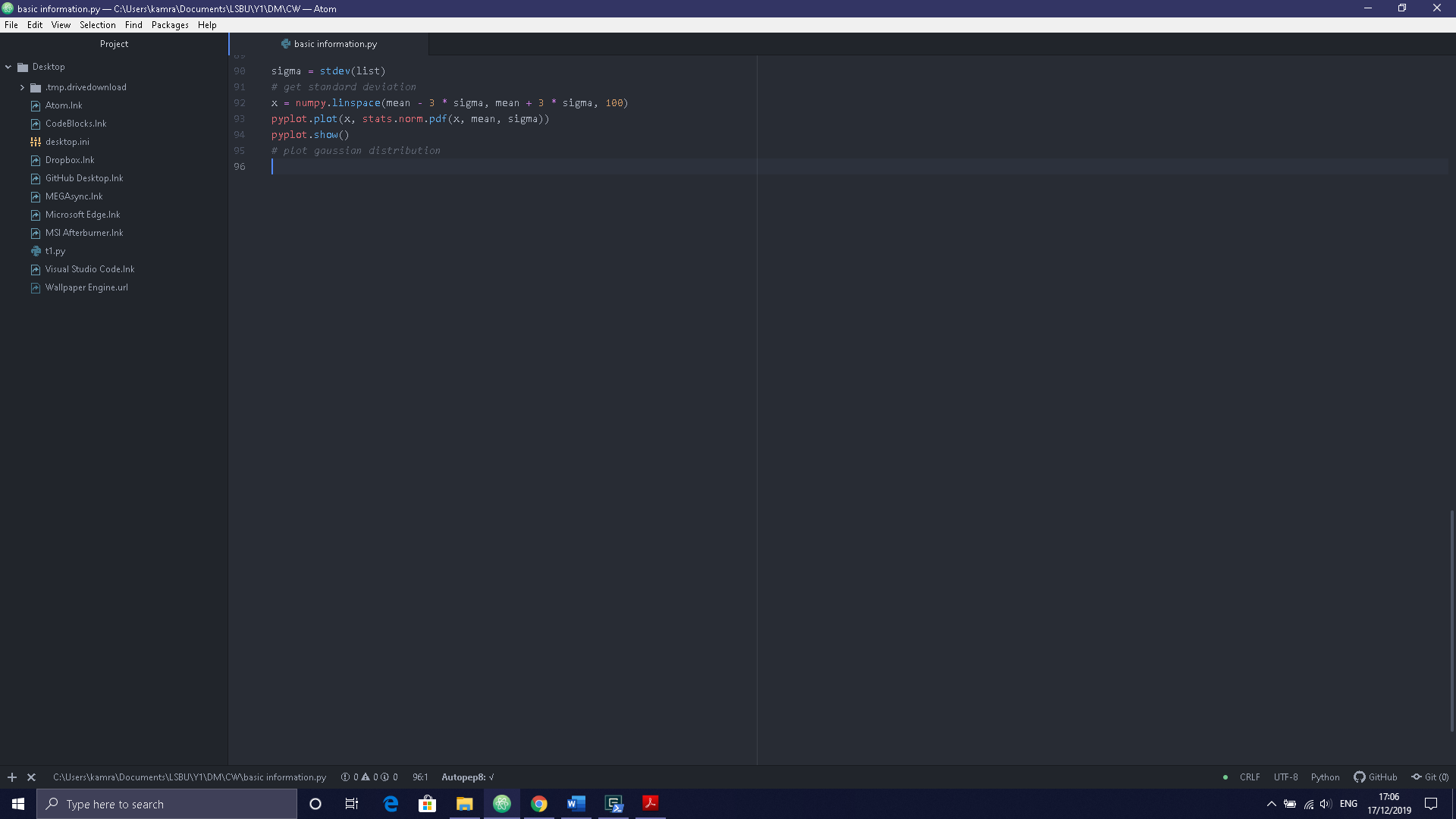
If the transportation problem is depicted as a digraph then paths 5,7,8,11 wouldn’t be included as they oppose the direction of the edges (assuming that the edges are directed such that LC (1,2) is a direction from node 1 to node 2 and that the other edges follow this rule). Due to this discrepancy, when calculating the probability distribution function and analysing the data for the paths, I will calculate the data for an undirected graph then have an additional section for if the graph is a digraph.

# Statistics

## Basic Information.py

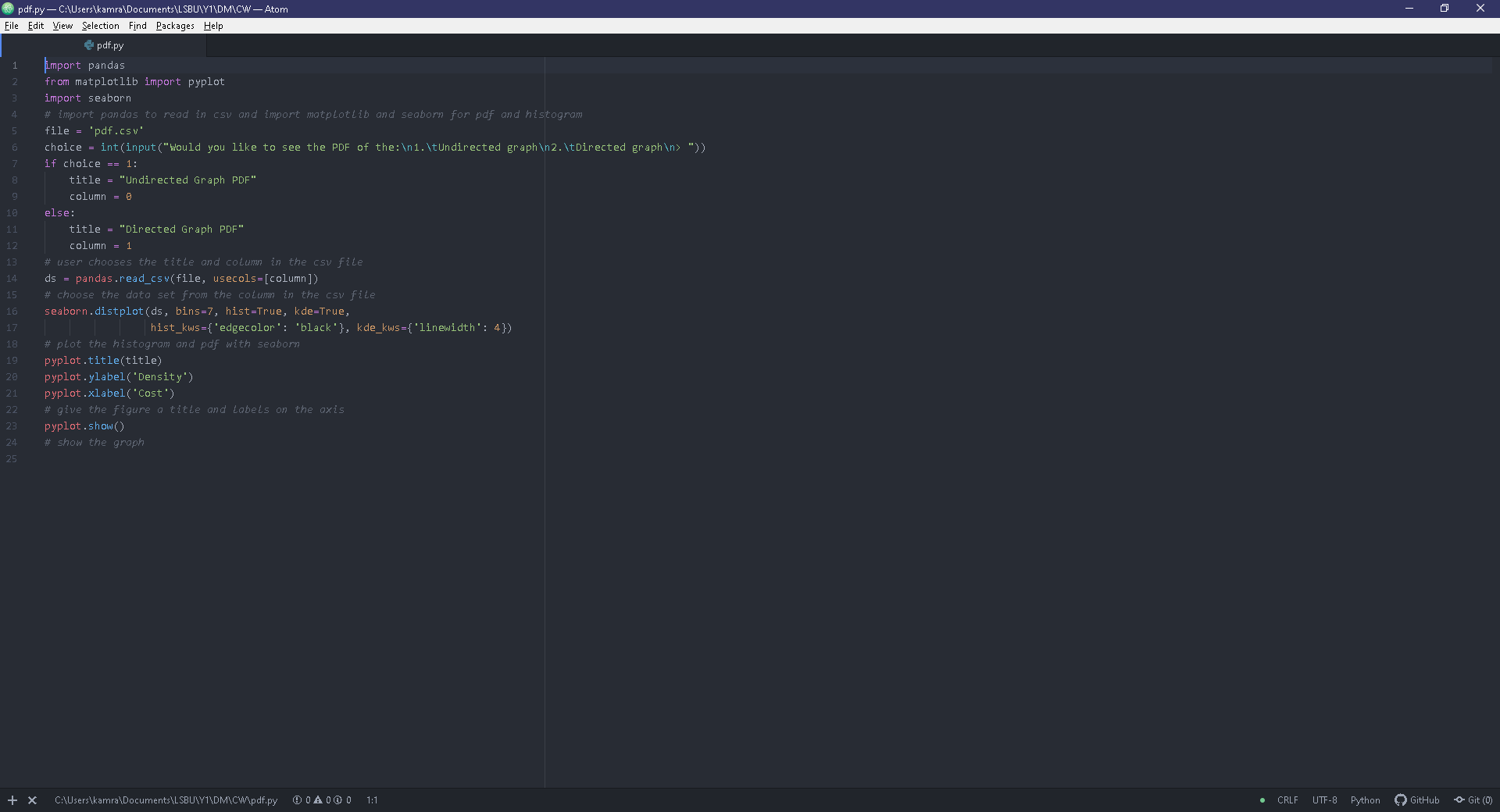






To fix the encoding on the csv file to be used in a python list I followed (senshin, 2015) which led me to implementing the boxplot with the aid of (unutbu, Horizontal box plots in matplotlib/Pandas, 2013) and the Gaussian Distribution with inspiration from (unutbu, python pylab plot normal distribution, 2012). This code reads in the file called “file.csv”.

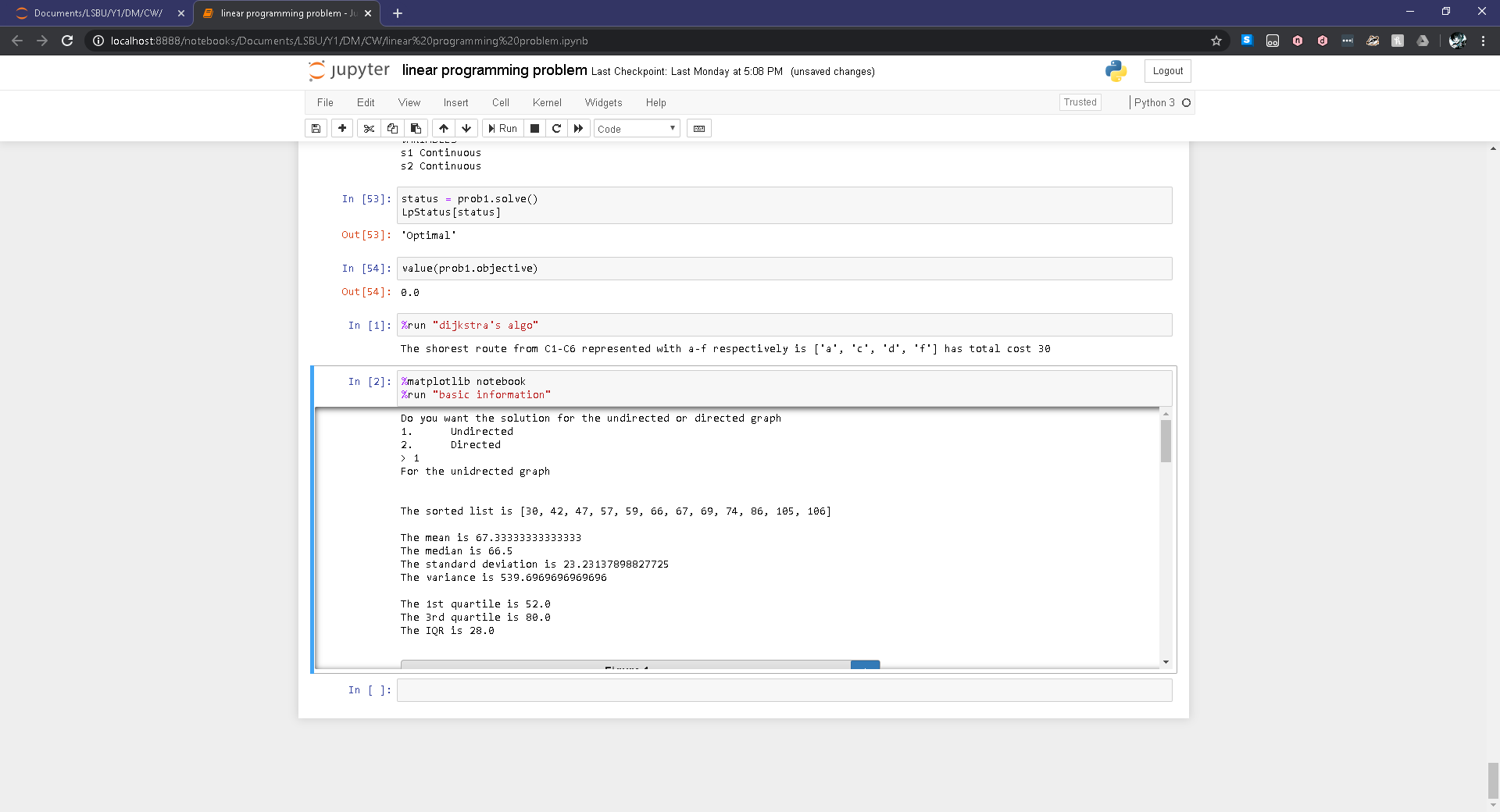
## PDF.py



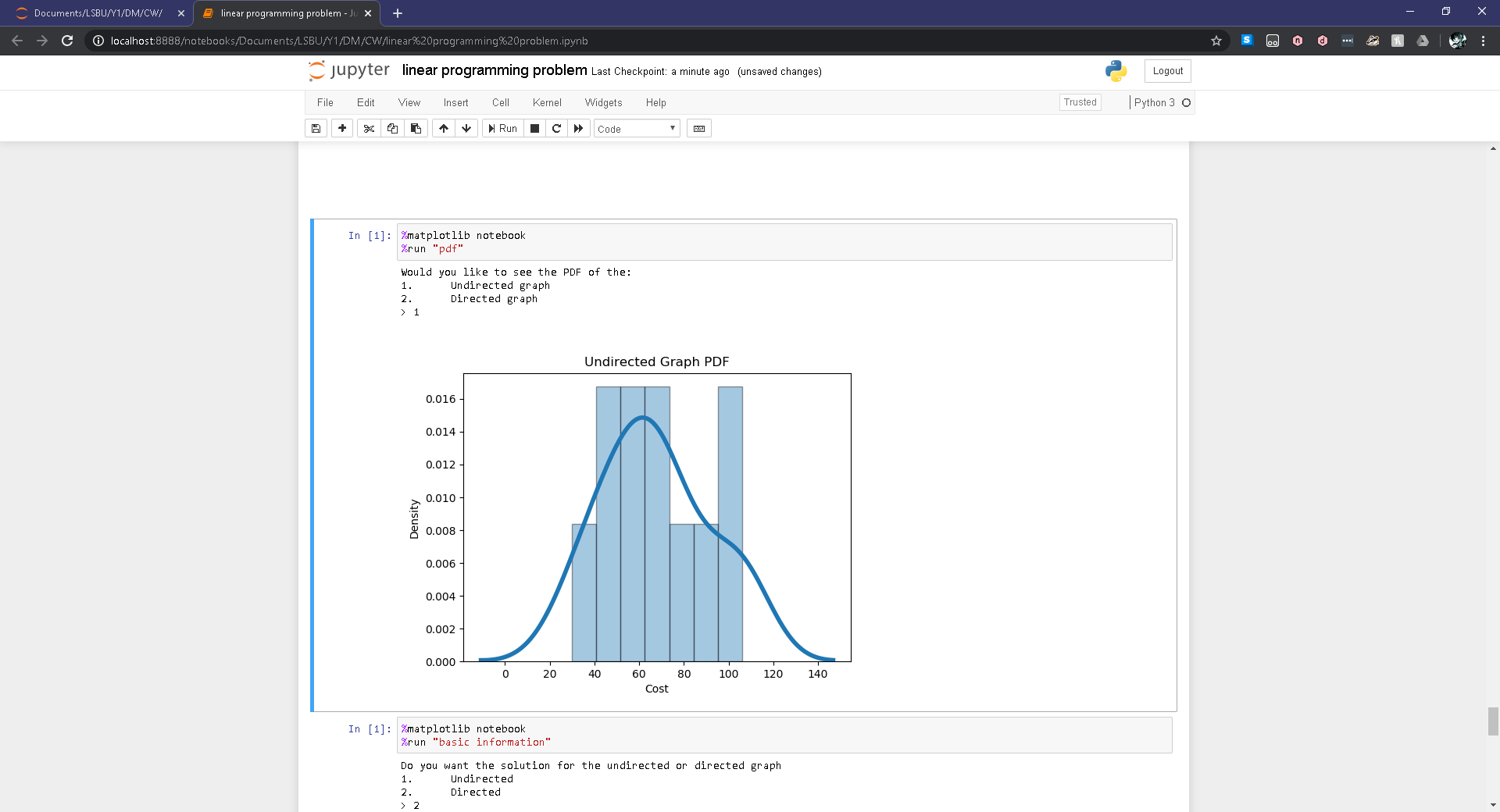
To plot the probability density function, I used (Koehrsen, 2018). This code reads in the file called “pdf.csv”.

## Path approach

##### Undirected Graph

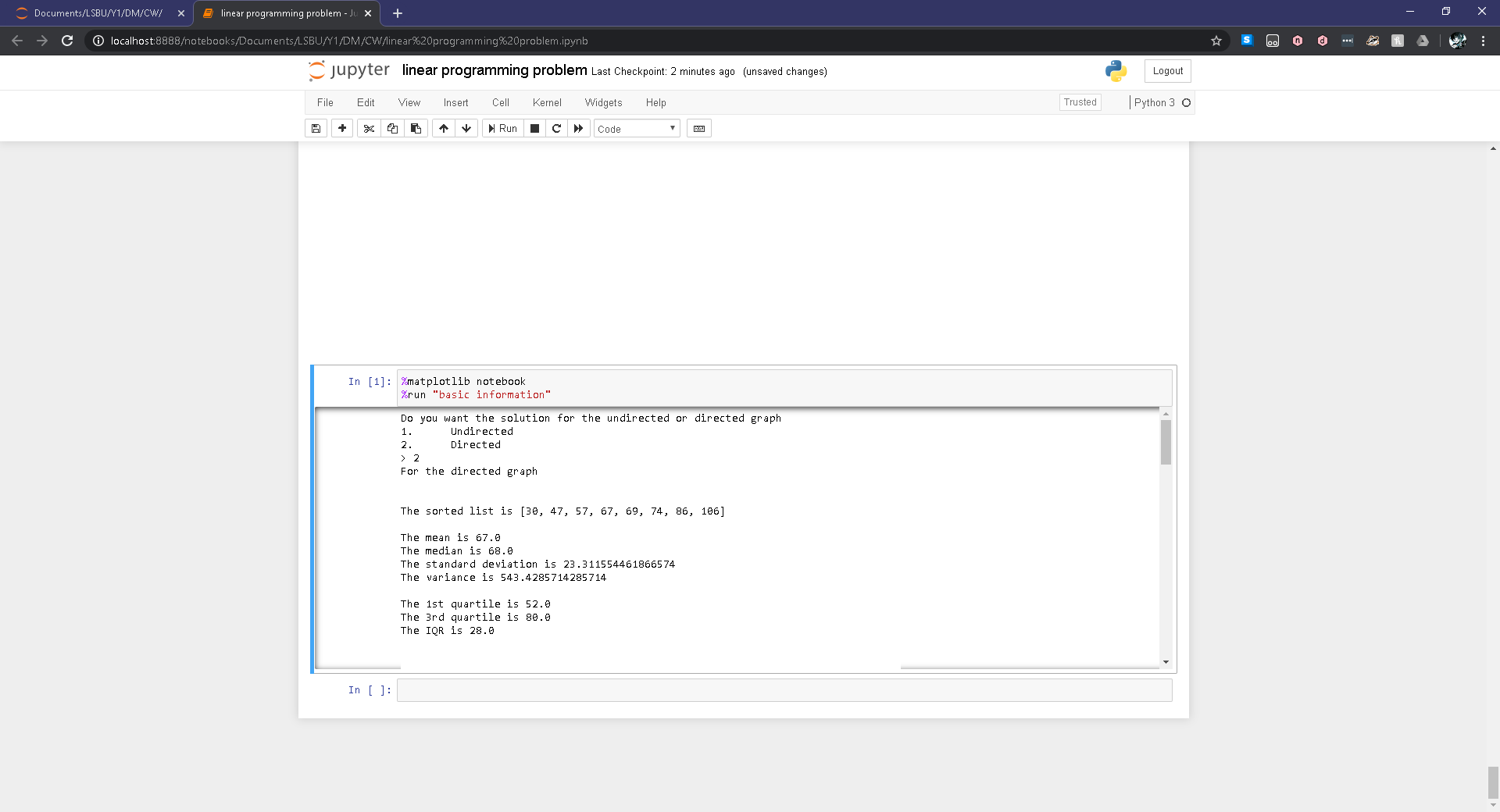


|  |  |
| --- | --- |
| **Box plot** | **Gaussian Distribution** |
|  |  |



The mean is greater than the median thus the distribution is positively skewed which is depicted clearly in the probability density function. When calculating for outliers (which is approximately 1.5 times the standard deviation from the mean) I found that the costs of 106, 105 and 30 would be considered outliers as the acceptable range of values would lie between 32.1529315176 and 101.847068482. This means that incorporating these values in my calculations and figures could be the cause of this distribution being positively skewed.

##### Directed Graph



|  |  |
| --- | --- |
| **Boxplot** | **Gaussian Distribution** |
|  |  |



The mean is less than the median thus the distribution is negatively skewed. When calculating for outliers (which is approximately 1.5 times the standard deviation from the mean) I found that the costs of 106 and 30 would be considered outliers as the acceptable values would approximately be in the range of 32.0326680729 and 101.967331927. This means that incorporating these values within my calculations could be the cause of the distribution being negatively skewed. Even though there appears to be a negative skewness in the data, the probability density function plotted appears to depict that the skewness isn’t perceptible as the graph appears similar to a Gaussian distribution due to its symmetrical nature which could be depicted by .

## What I can conclude?

On both graphs the minimum and maximum points on the boxplots are the same as both paths are applicable to both scenarios. The mean in the undirected graph is fractionally larger than the mean in the directed graph but the means are very similar. Although the directed graph has less data points in the distribution than the undirected graph; the first quartile, last quartile and inter quartile range are the same in both situations thus insinuating that most of the data is heavily weighted within the region enclosed by the first quartile and last quartile.

The standard deviation is the same for both scenarios thus showing that the data of both graphs are similarly distributed however, the variance in the directed graph is higher than the variance in the undirected graph thus the data in the directed graph is further distributed from the mean than the data in the undirected graph. This is further emphasised by the probability density function of both graphs as for the directed graph, the data is distributed evenly but in the undirected graph, the data is more concentrated between an approximate cost of 40 and 70.

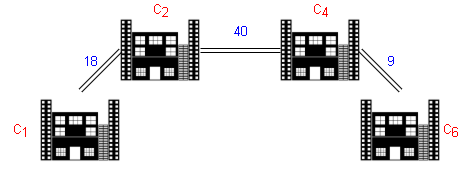
If the outliers are included the optimistic cost is 30 and pessimistic cost is 106 however, if not included the pessimistic cost in both scenarios is 86 but for the undirected graph, the optimistic cost would be 42 whilst for the directed graph the optimistic cost would be 47 while operating under the assumption that optimistic means least costly and pessimistic means most costly means of transport.

|  |  |  |
| --- | --- | --- |
| **Situation** | **Optimistic** | **Pessimistic** |
| **With outliers** | **Cost = 30** | **Cost = 106** |
| **Undirected without outliers** | **Cost = 42** | **Cost = 86** |
| **Directed without outliers** | **Cost = 47** |

## Cartesian Product Approach

I will be using set theory to determine the cartesian product of all of the given paths. Each member of the set made by the cartesian product will be a tuple.

##### Path 1



{0,18} x {0,40} x {0,9} = {(0,0,0), (0,0,9), (0,40,0), (0,40,9), (18,0,0), (18,0,9), (18,40,0), (18,40,9)}

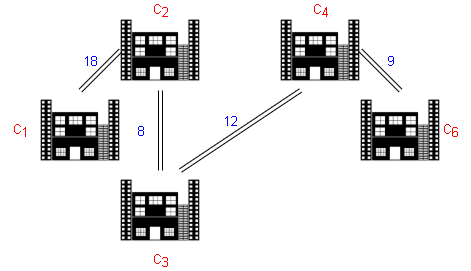
Costs = 0, 9, 40, 49, 18, 27, 58, 67

##### Path 2

{0,18} x {0,40} x {0,12} x {0,36} = {(0,0,0,0), (0,0,0,36), (0,0,12,0), (0,0,12,36), (0,40,0,0), (0,40,0,36), (0,40,12,0), (0,40,12,36), (18,0,0,0), (18,0,0,36), (18,0,12,0), (18,0,12,36), (18,40,0,0), (18,40,0,36), (18,40,12,0), (18,40,12,36)}

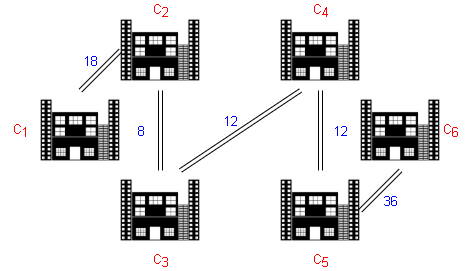
Costs = 0, 36, 12, 48, 40, 76, 52, 88, 18, 54, 30, 66, 58, 94, 70, 106

##### Path 3

{0,18} x {0,8} x {0,12} x {0,9} = {(0,0,0,0), (0,0,0,9), (0,0,12,0), (0,0,12,9), (0,8,0,0), (0,8,0,9), (0,8,12,0), (0,8,12,9), (18,0,0,0), (18,0,0,9), (18,0,12,0), (18,0,12,9), (18,8,0,0), (18,8,0,9), (18,8,12,0), (18,8,12,9)}

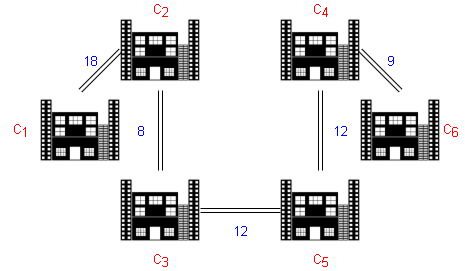
Costs = 0, 9, 12, 21, 8, 17, 20, 29, 18, 27, 30, 39, 26, 35, 38, 47

##### Path 4

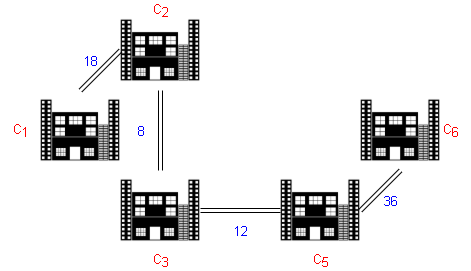
{0,18} x {0,8} x {0,12} x {0,12} x {0,36} = {(0,0,0,0,0), (0,0,0,0,36), (0,0,0,12,0), (0,0,0,12,36), (0,0,12,0,0), (0,0,12,0,36), (0,0,12,12,0), (0,0,12,12,36), (0,8,0,0,0), (0,8,0,0,36), (0,8,0,12,0), (0,8,0,12,36), (0,8,12,0,0), (0,8,12,0,36), (0,8,12,12,0), (0,8,12,12,36), (18,0,0,0,0), (18,0,0,0,36), (18,0,0,12,0), (18,0,0,12,36), (18,0,12,0,0), (18,0,12,0,36), (18,0,12,12,0), (18,0,12,12,36), (18,8,0,0,0), (18,8,0,0,36), (18,8,0,12,0), (18,8,0,12,36), (18,8,12,0,0), (18,8,12,0,36), (18,8,12,12,0), (18,8,12,12,36)}

Costs = 0, 36, 12, 48, 12, 48, 24, 60, 8, 44, 20, 56, 20, 56, 32, 68, 18, 54, 30, 66, 30, 66, 42, 78, 26, 62, 38, 74, 38, 74, 50, 86

##### Path 5

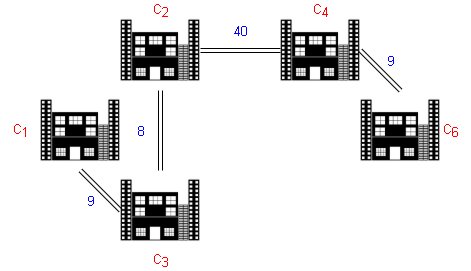
{0,18} x {0,8}, {0,12} x {0,12} x {0,9} = {(0,0,0,0,0), (0,0,0,0,9), (0,0,0,12,0), (0,0,0,12,9), (0,0,12,0,0), (0,0,12,0,9), (0,0,12,12,0), (0,0,12,12,9), (0,8,0,0,0), (0,8,0,0,9), (0,8,0,12,0), (0,8,0,12,9), (0,8,12,0,0), (0,8,12,0,9), (0,8,12,12,0), (0,8,12,12,9), (18,0,0,0,0), (18,0,0,0,9), (18,0,0,12,0), (18,0,0,12,9), (18,0,12,0,0), (18,0,12,0,9), (18,0,12,12,0), (18,0,12,12,9), (18,8,0,0,0), (18,8,0,0,9), (18,8,0,12,0), (18,8,0,12,9), (18,8,12,0,0), (18,8,12,0,9), (18,8,12,12,0), (18,8,12,12,9)}

Costs = 0, 9, 12, 21, 12, 21, 24, 33, 8, 17, 20, 29, 20, 29, 32, 41, 18, 27, 30, 39, 30, 39, 42, 51, 26, 35, 38, 47, 38, 47, 50, 59

Path 6{0,18} x {0,8} x {0,12} x {0,36} = {(0,0,0,0), (0,0,0,36), (0,0,12,0), (0,0,12,36), (0,8,0,0), (0,8,0,36), (0,8,12,0), (0,8,12,36), (18,0,0,0), (18,0,0,36), (18,0,12,0), (18,0,12,36), (18,8,0,0), (18,8,0,36), (18,8,12,0), (18,8,12,36)}

Costs = 0, 36, 12, 48, 8, 44, 20, 56, 18, 54, 30, 66, 26, 62, 38, 74

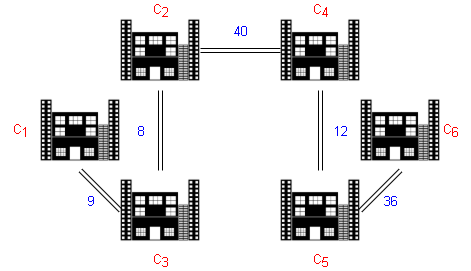
##### Path 7



{0,9} x {0,8} x {0,9} x {0,40} = {(0,0,0,0), (0,0,0,40), (0,0,9,0), (0,0,9,40), (0,8,0,0), (0,8,0,40), (0,8,9,0), (0,8,9,40), (9,0,0,0), (9,0,0,40), (9,0,9,0), (9,0,9,40), (9,8,0,0), (9,8,0,40), (9,8,9,0), (9,8,9,40)}

Costs = 0, 40, 9, 49, 8, 48, 17, 57, 9, 49, 18, 58, 17, 57, 26, 66

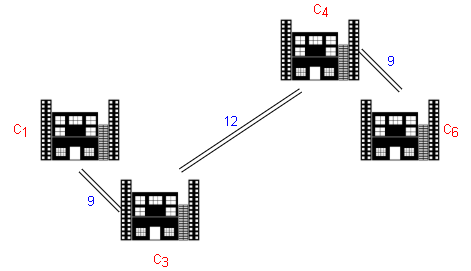
##### Path 8



{0,40} x {0,8}, {0,12} x {0,36} x {0,9} = {(0,0,0,0,0), (0,0,0,0,9), (0,0,0,36,0), (0,0,0,36,9), (0,0,12,0,0), (0,0,12,0,9), (0,0,12,36,0), (0,0,12,36,9), (0,8,0,0,0), (0,8,0,0,9), (0,8,0,36,0), (0,8,0,36,9), (0,8,12,0,0), (0,8,12,0,9), (0,8,12,36,0), (0,8,12,36,9), (40,0,0,0,0), (40,0,0,0,9), (40,0,0,12,0), (40,0,0,36,9), (40,0,12,0,0), (40,0,12,0,9), (40,0,12,36,0), (40,0,12,36,9), (40,8,0,0,0), (40,8,0,0,9), (40,8,0,36,0), (40,8,0,36,9), (40,8,12,0,0), (40,8,12,0,9), (40,8,12,36,0), (40,8,12,36,9)}

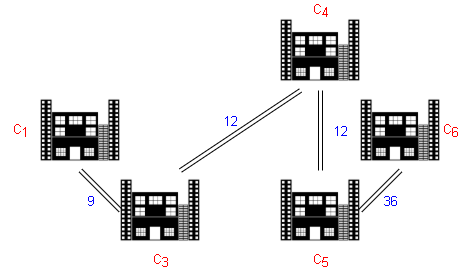
Costs = 0,9, 36, 45, 12, 21, 48, 57, 8, 17, 44, 53, 20, 29, 56, 65, 40, 49, 52, 85, 52, 61, 88, 97, 48, 57, 84, 93, 60, 69, 96, 105

##### Path 9

{0,9} x {0,12} x {0,9} = {(0,0,0), (0,0,9), (0,12,0), (0,12,9), (9,0,0), (9,0,9), (9,12,0), (9,12,9)}

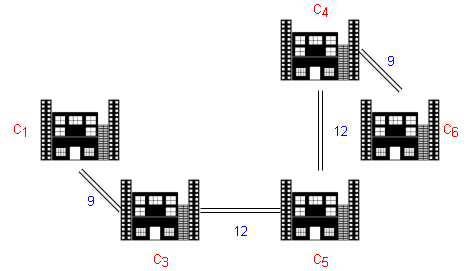
Costs = 0, 9, 12, 21, 9, 18, 21, 30

##### Path 10

{0,9} x {0,12} x {0,12} x {0,36} = {(0,0,0,0), (0,0,0,36), (0,0,12,0), (0,0,12,36), (0,12,0,0), (0,12,0,36), (0,12,12,0), (0,12,12,36), (9,0,0,0), (9,0,0,36), (9,0,12,0), (9,0,12,36), (9,12,0,0), (9,12,0,36), (9,12,12,0), (9,12,12,36)}

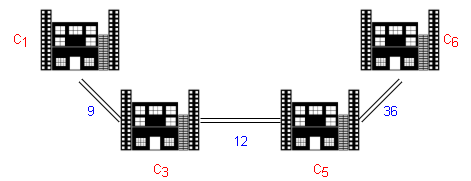
Costs = 0, 36, 12, 48, 12, 48, 24, 60, 9, 45, 21, 57, 21, 57, 33, 69

##### Path 11

{0,9} x {0,12} x {0,12} x {0,9} = {(0,0,0,0), (0,0,0,9), (0,0,12,0), (0,0,12,9), (0,12,0,0), (0,12,0,9), (0,12,12,0), (0,12,12,9), (9,0,0,0), (9,0,0,9), (9,0,12,0), (9,0,12,9), (9,12,0,0), (9,12,0,9), (9,12,12,0), (9,12,12,9)}

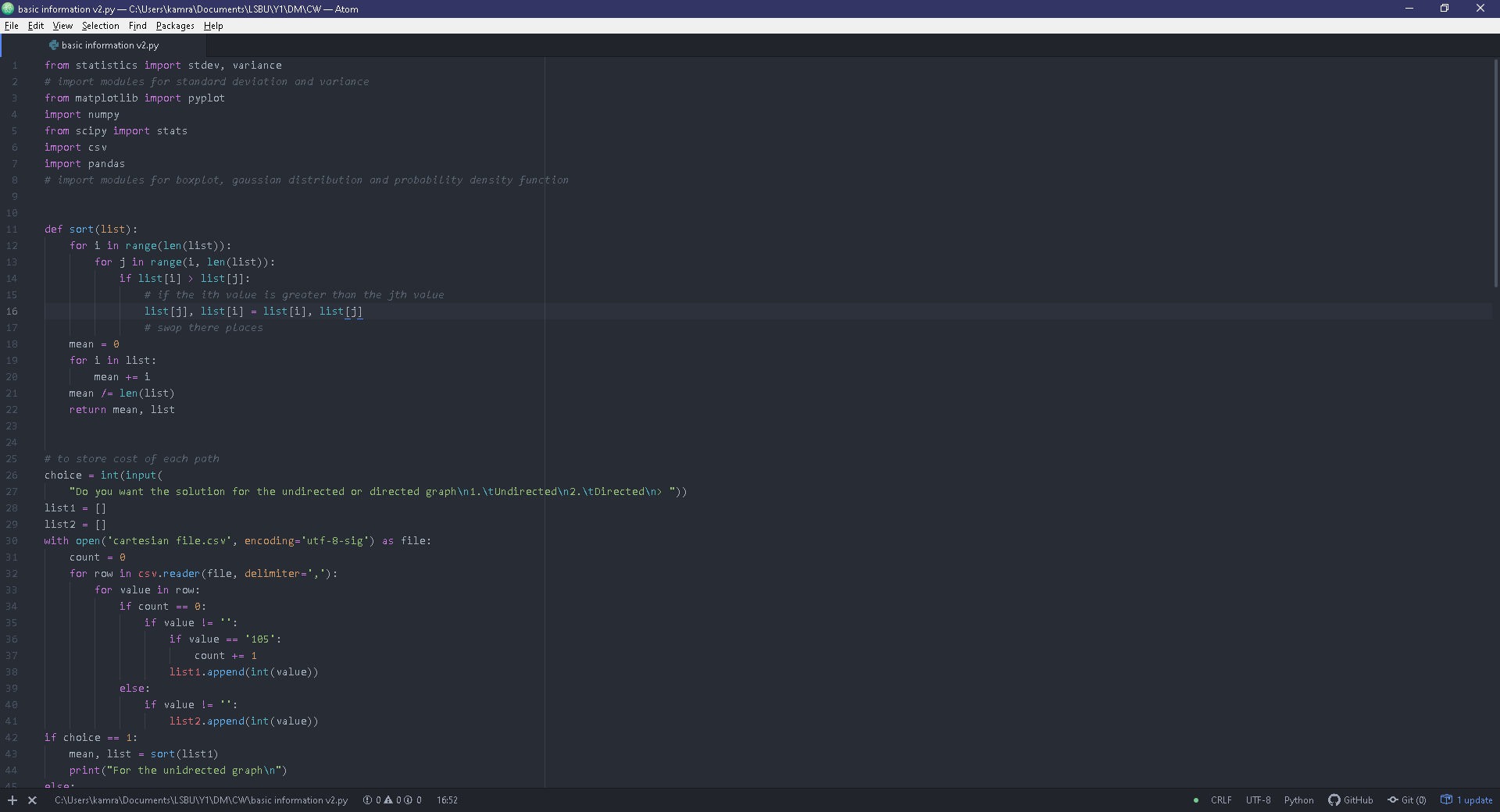
Costs = 0, 9, 12, 21, 12, 21, 24, 33, 9, 18, 21, 30, 21, 30, 33, 42

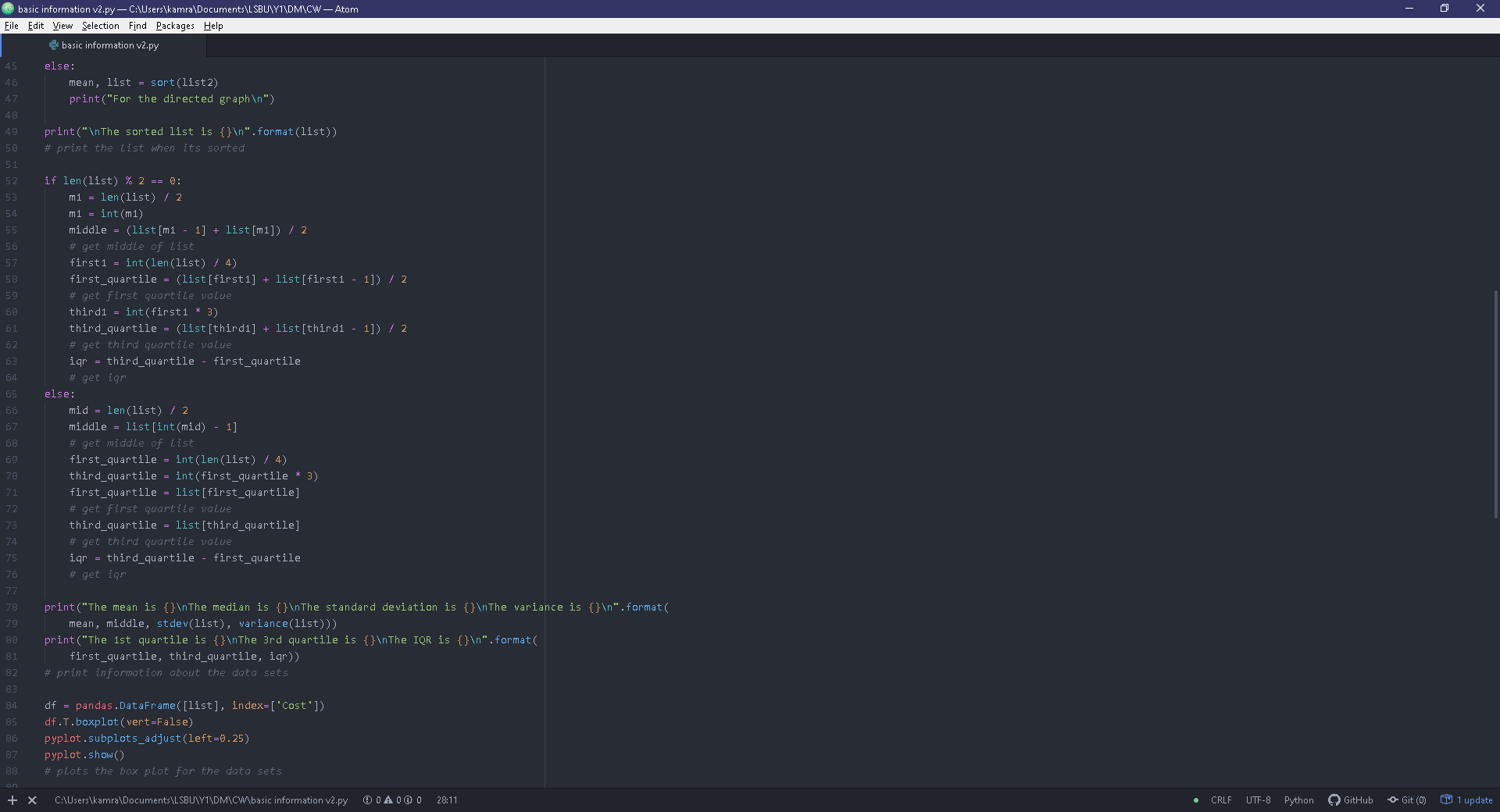
##### Path 12

{0,9} x {0,12} x {0,36} = {(0,0,0}, (0,0,36), (0,12,0), (0,12,36), (9,0,0), (9,0,36), (9,12,0), (9,12,36)}

Costs = 0, 36, 12, 48, 9, 45, 21, 57

## Basic Information v2.py

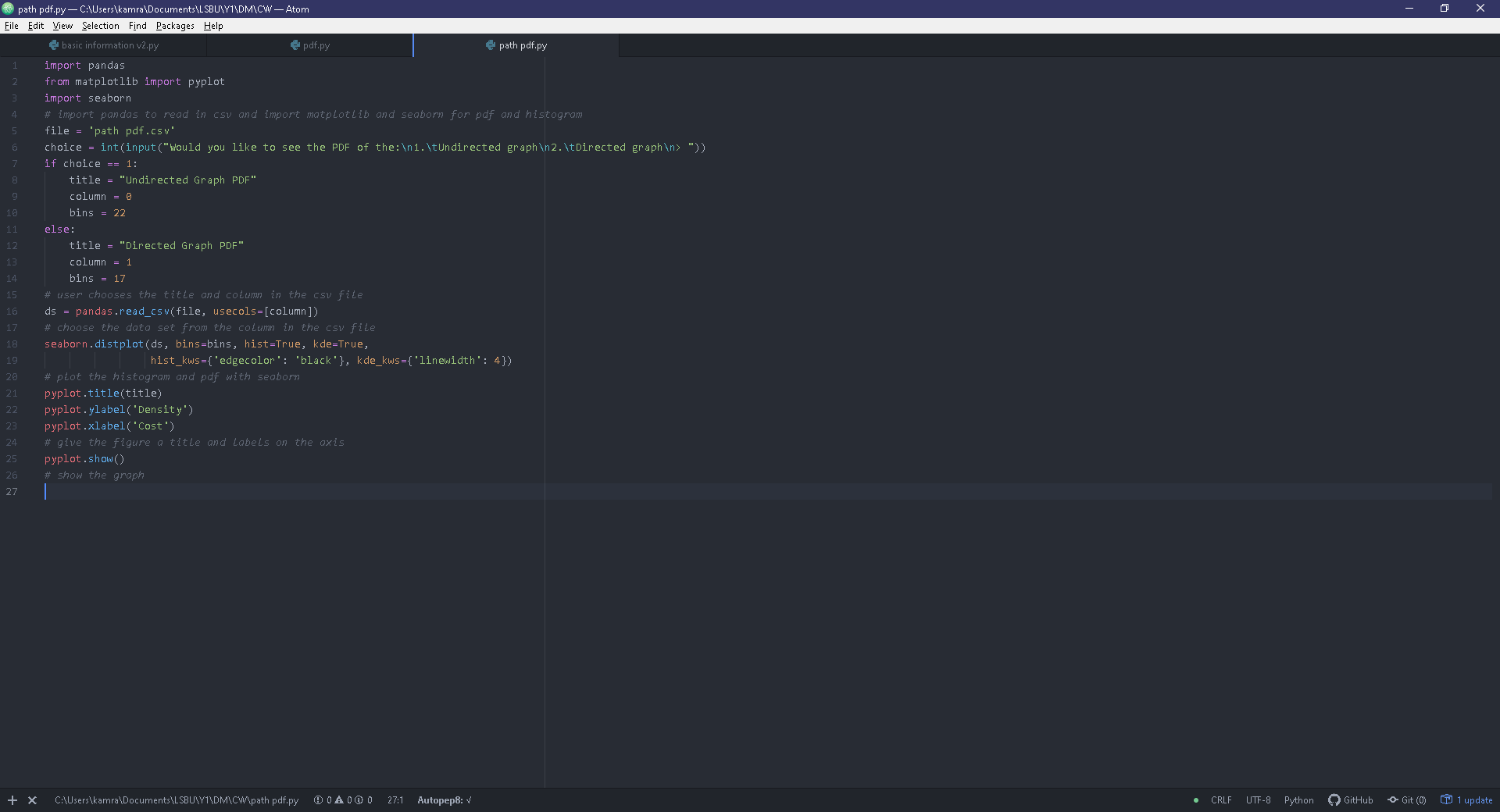






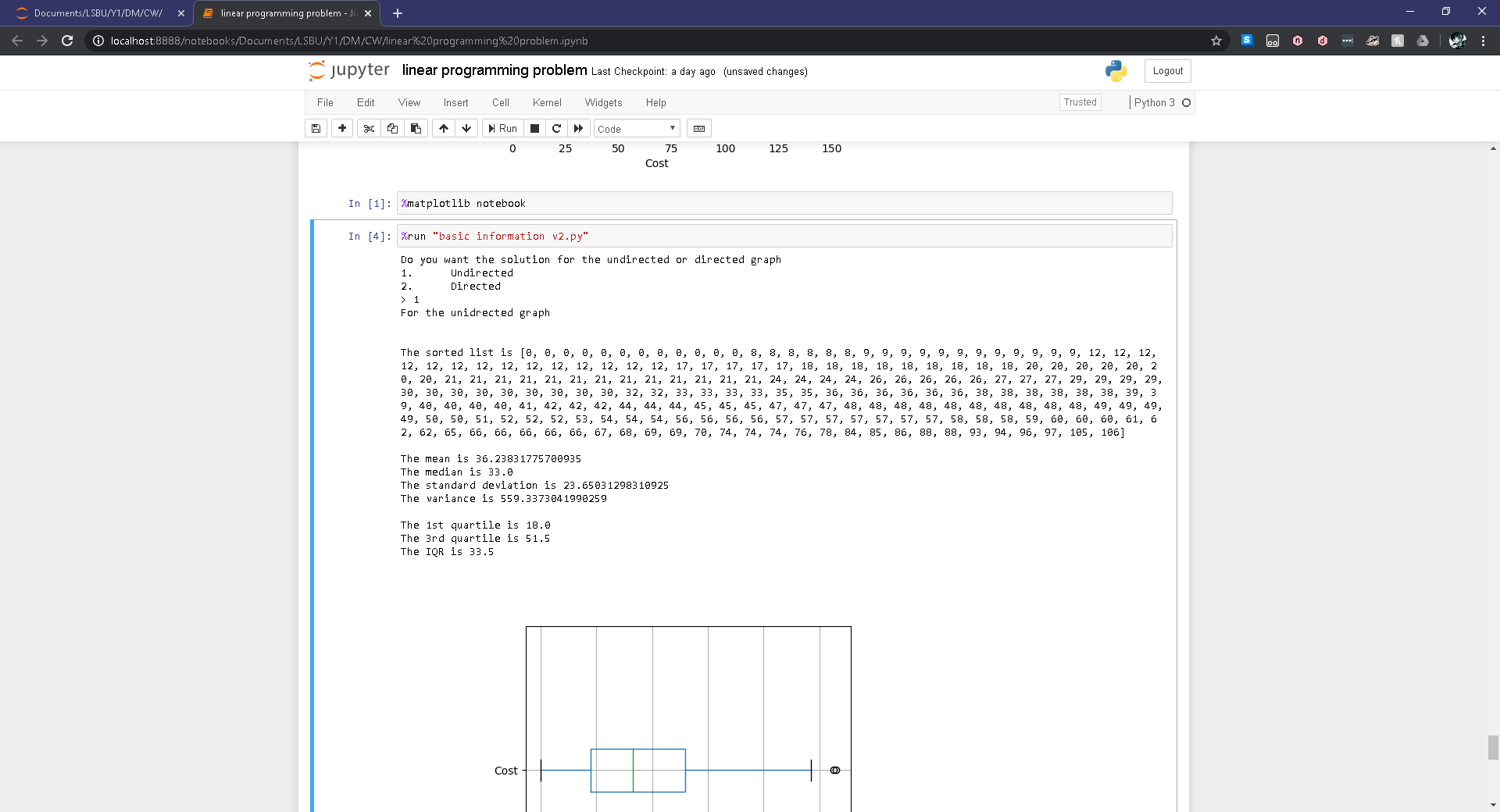
This is essentially the same as the original ‘Basic Information.py’ file inspired from however, instead of reading from ‘file.csv’, I am reading from ‘cartesian.csv’.

## Path PDF.py

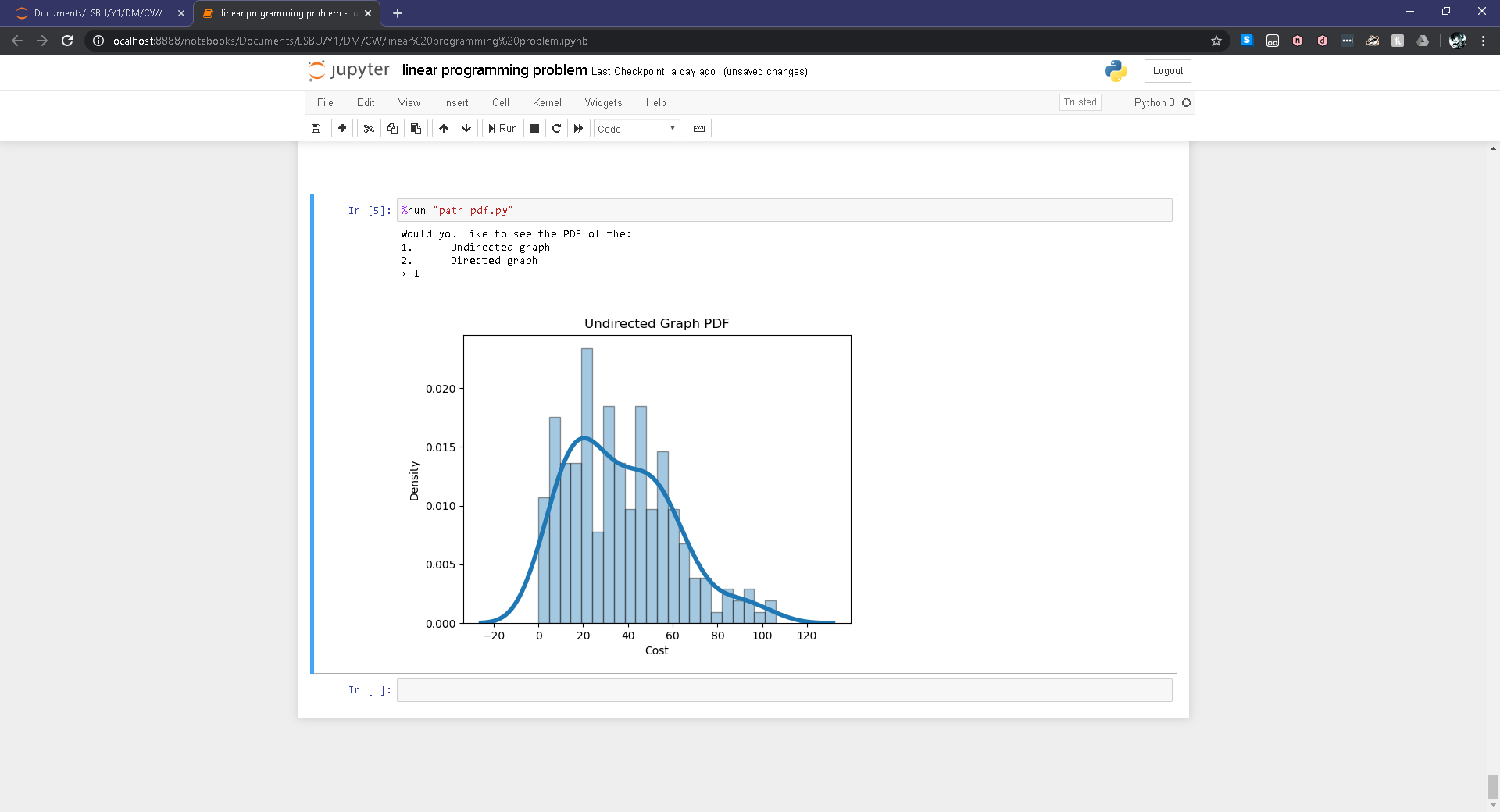


This is essentially the same as the original PDF.py but instead of reading ‘pdf.csv’, the program is reading in ‘path pdf.csv’.

## Undirected graph

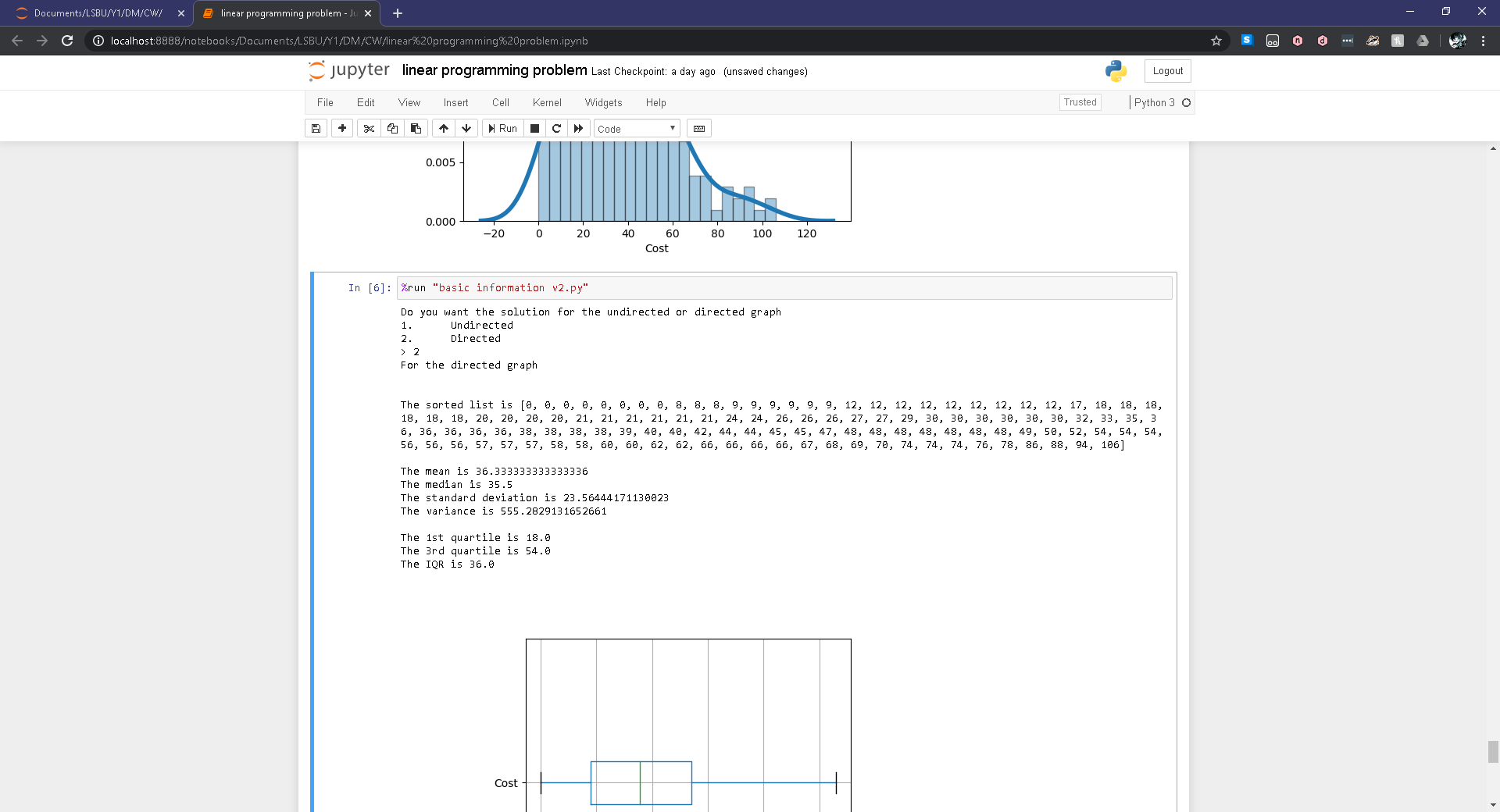


|  |  |
| --- | --- |
| **Box plot** | **Gaussian Distribution** |
|  |  |

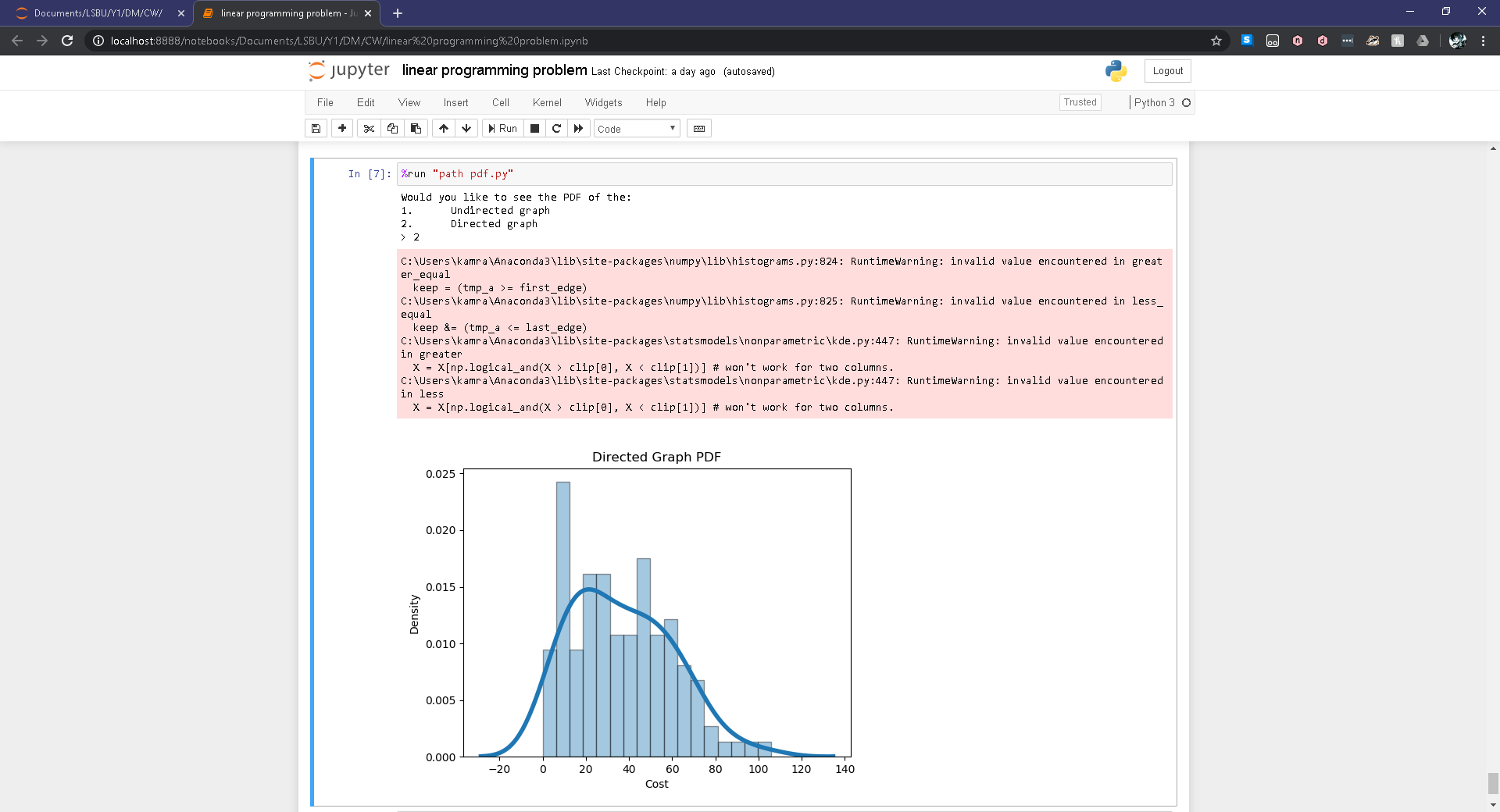


The mean is greater than the median therefore the distribution should appear to be positively skewed which is clearly displayed in the probability density function. The circles in the boxplot represent outliers which don’t coincide with the distribution of data. When calculating for outliers (which is approximately 1.5 times the standard deviation from the mean) I found that any costs below approximately 0.76 and above 71.71 could cause the distribution to be skewed in a given direction. As most of the concentration of data is within the region closer to 0, I can assume that outliers below 0.76 contributed to giving the distribution a positive skewness.

## Directed graph



|  |  |
| --- | --- |
| **Box plot** | **Gaussian Distribution** |
|  |  |



The mean is greater than the median therefore the distribution should appear to be positively skewed which is clearly displayed in the probability density function. When calculating for outliers (which is approximately 1.5 times the standard deviation from the mean) I found that any costs below approximately 0.99 and above 71.68 could cause the distribution to be skewed in a given direction. As most of the concentration of data is within the region closer to 0, I can assume that outliers below 0.99 contributed to giving the distribution a positive skewness.

## Conclusion

The mean in the undirected graph is fractionally smaller than the mean in the directed graph but the means are very similar however, the median for the undirected graph is greater than the median for the directed graph.

Although the directed graph has less data points in the distribution than the undirected graph, the first quartile is the same in both graphs but the third quartile and interquartile range are much larger in the directed graph. From this I can infer that the data in the directed graph is more distributed than the undirected graph as more data points lie within the region enclosed by the directed graphs inter-quartile range than the undirected graphs inter-quartile range. This is further emphasised by the probability density functions as the directed graph has less kutosis than the undirected graph to which, in the directed graph the concentration appears to gradually build nearer the mean but in the undirected graph the concentration appears to build nearer the mean faster thus making the distribution appear sharper.

Given that outliers are not included I have concluded that the optimistic cost of transport would be a path giving a cost of 8 and the pessimistic cost would be a path giving a cost of 70. However, if the outliers are included then the optimistic cost of transport would be a path giving a cost of 0 and the pessimistic cost would be a path giving a cost of 106. This is applicable for both scenarios as the values I have chosen for optimistic and pessimistic costs are in both graphs.

# Bibliography

Bhatti, K. M. (2019, December 8). *Discrete Maths*. Retrieved from GitHub: https://github.com/k5924/DiscreteMaths

Koehrsen, W. (2018, March 23). *Histograms and Density Plots in Python*. Retrieved from Towards Data Science: https://towardsdatascience.com/histograms-and-density-plots-in-python-f6bda88f5ac0

senshin. (2015, December 21). *Why does my Python code print the extra characters “ï»¿” when reading from a text file?* Retrieved from stackoverflow: https://stackoverflow.com/questions/34399172/why-does-my-python-code-print-the-extra-characters-%C3%AF-when-reading-from-a-tex/34399309

Sullivan, I. (2017, June 14). *Implementation of dijkstra in python*. Retrieved from Youtube: https://www.youtube.com/watch?v=IG1QioWSXRI&t=578s

unutbu. (2012, April 13). *python pylab plot normal distribution*. Retrieved from stackoverflow: https://stackoverflow.com/questions/10138085/python-pylab-plot-normal-distribution

unutbu. (2013, August 28). *Horizontal box plots in matplotlib/Pandas*. Retrieved from stackoverflow: https://stackoverflow.com/questions/18500011/horizontal-box-plots-in-matplotlib-pandas

Wang, Y. (2019, August 20). *18B\_Linear\_Programming\_using\_PuLP.* Retrieved from GitHub: https://github.com/yongtwang/engineering-python/tree/master/18B\_Linear\_Programming\_using\_PuLP