Exercise 2 Part 1
Implementation:
In this assignment, we assigned a PageRank algorithm that with the goal to deal with dead ends, spider traps and minimizing computational resources. This involves the modification in calculation by using each random surfer a small probability of teleporting to a random page based on the equation v = Mv by Gaussian elimination.

Taxation is utilised to address both dead ends and spider traps in the text file. For context, dead ends are nodes that does not have any outward links to other nodes, while spider traps are a set of nodes with edges that only links within the page. To do so, we iteratively compute PageRank using the formula $v' = \beta Mv + (1 - \beta)e/n$, where v' is the new vector estimate of PageRank from the current PageRank estimate v, the transition matrix M and β as a chosen constant, which in this case will be 0.85 in this implementation for comparison with the default pagerank function in networkx. e is a vector of all ones and n is the number of nodes. βMv represents the case where, with probability β, the random surfer decides to follow an out-link from their present page, which can be further accessed using the formula $(1 - \beta)e/n$ if the transition matrix is not used, while obtaining the probability $1 - \beta$, of a new random surfer at a random page.

Computational time and memory usage:
The computational time for Part 1 implementation is 273.7 seconds, which is lower compared to 471.3 seconds from networkx dictionary approach while running on the web-Google.txt dataset from Diagram 1.

Exercise 2 Part 2:
The solution for the PageRank of all nodes is saved in the `ex2.txt` file, while the ordered PageRank list for largest 10 nodes is depicted in Diagram 1 taken from the console when run:

```
--------------------------------------------------------------------------

PageRank, with a list of 10 nodes that have the highest page ranks:

[('163075', 0.0013094172929953 29), ('537039', 0.0011599085345 8439), ('597621', 0.0011183948584514883), ('605856', 0.0010040486306168623), ('819223
', 0.0009281287640686552), ('885605', 0.0009134698157415682), ('551829', 0.0008958384326752293), ('751384', 0.0008955038067346354), ('908351', 0.00
08846861989414661), ('504140', 0.0008529450642486126)]

The algorithm for implemented PageRank used 273.7 seconds 1772.1 megabyte


Networkx results for benchmarking, with a list of 10 nodes that have the highest page ranks:

[('163075', 0.0009521123333766876), ('597621', 0.0009013686628239585), ('537039', 0.0008953815726589841), ('837478', 0.0008761661604313413), ('8856
05', 0.0008216087428295121), ('551829', 0.0007901082073484922), ('41909', 0.0007794946931031651), ('605856', 0.0007791356753662204), ('504140', 0.0
007457503352830866), ('819223', 0.0007101828701990114)]

The algorithm for networkx built-in PageRank used 471.3 seconds 1852.0 megabyte

--------------------------------------------------------------------------
```

Diagram 1: The ordered list of 10 largest PageRank and its computational resources

Exercise 3 Part 1
Implementation:
Given that it is a 1-dimensional set of points, we first use the points as separate clusters. By assigning the distance between any 2 clusters as the Euclidean distance between their centroids, we merge the 2 clusters at the shortest distance in each iteration and plot the dendrogram to show the clusters based on their correlation as shown in Diagram 2:

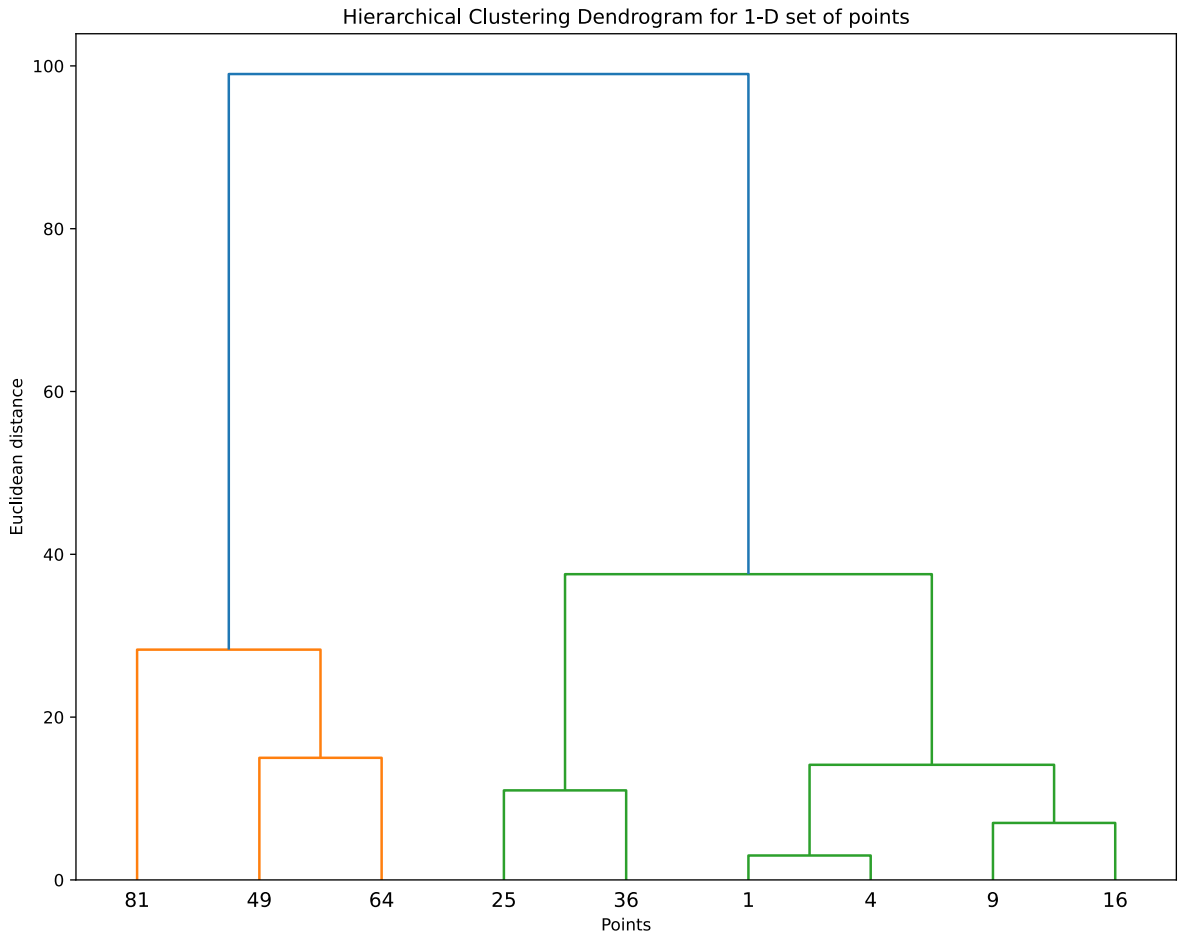

Diagram 2: A dendrogram for the given set of points

Exercise 3 Part 2a
Implementation:
We first decide on the number of clusters, which is k = 3 based on the elbow and silhouette coefficient score. Next, we select k random points from the data as centroids. We then iteratively assign the points to the nearest cluster centroids and calculate the centroid of the newly formed clusters until it reached a maximum iteration of 100 or there is no change in the centroids. Attached below is the outcome when iris dataset is computed:

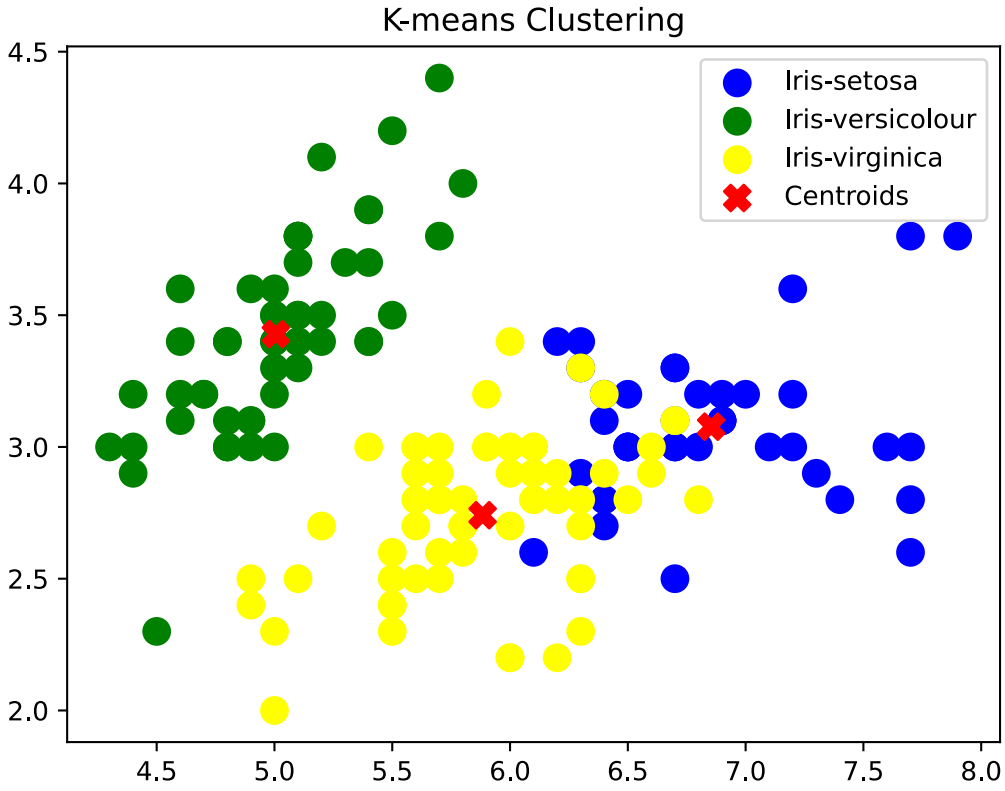

Diagram 3: A scatterplot for K means clustering

```
--------------------------------------------------------------------------------
The algorithm for K-means used   0.2 seconds and 154.8 megabyte
--------------------------------------------------------------------------------
```

Computational time:
The computational time for Part 2a implementation is 0.2 seconds running on the iris dataset, if we only produce the scatterplot without plt.show(). Given that the algorithm will run at a fixed maximum iteration of 150, in the worst case, the algorithm will run at O(r*n*c), where r is the observed labels, n is the number of features and c is the number of centroids of clusters

Exercise 3 Part 2b
The value k can be evaluated using the elbow method and the silhouette analysis to provide the optimal model performance based on different k number of clusters.

k represents the number of clusters that is used to determine the number of random points for constructing centroids to assign the nearest cluster centroids through user input, hence, comparing the inertia, or the sum of squared distance between the data points and their cluster centroids in the elbow method allow us to better determine the k number of cluster. By iterating a range of k values from 1 to 10 to avoid having too many clusters for centroid construction and calculate the values of distortions, which is the average of squared distance

from the centroid cluster for each k in visualisations, we can obtain the best k using the point where the plotted curve starts to flat out and where either the distortion or inertia decline in a linear pattern.

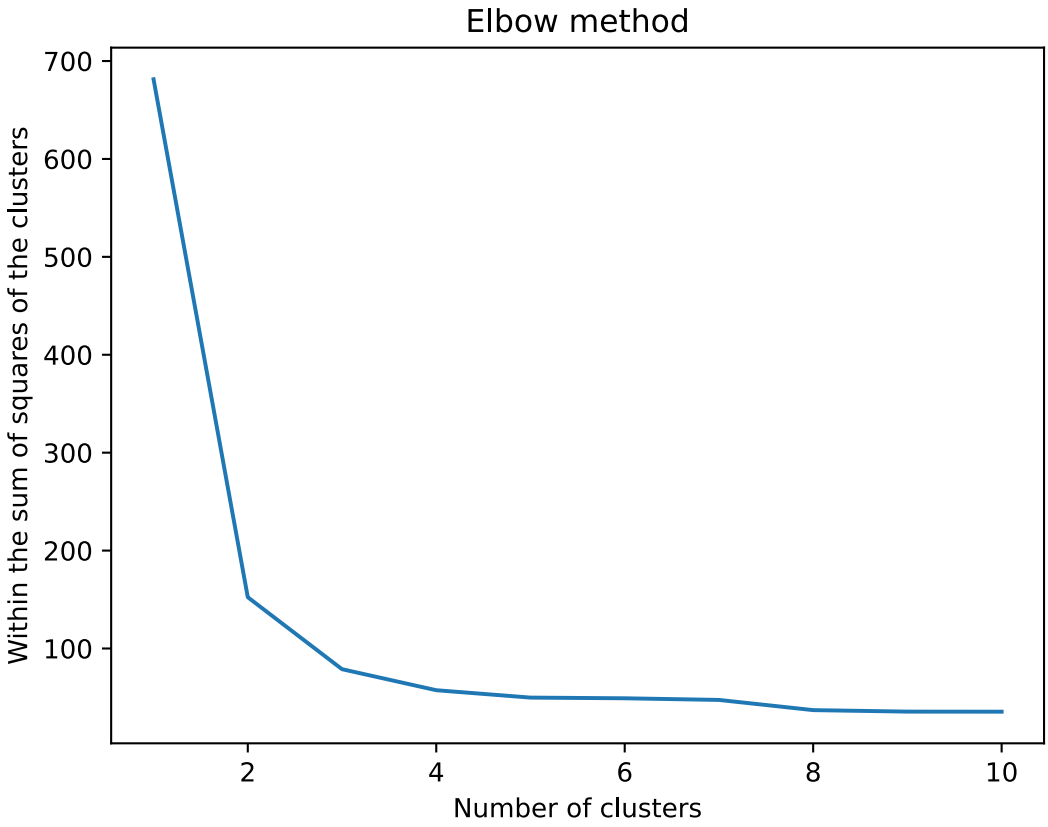

Diagram 4: Line graph for elbow method using inertia. Notice that 3 provides a clear elbow in a range from 1 to 10, which indicates that 3 is the best number of clusters and k = 3 is used for K-means clustering for iris dataset.

Based on Diagram 4, k = 3 can provides as with the optimal inertia which we can segmentize into 3 clusters with minimal error

However, this approach can be inadequate for evaluation as the error function is decreasing for all k. Therefore, an alternative approach that is also considered, but may be implemented in the future, is the silhouette analysis, which is used to determine the degree of separation between the clusters. We first calculated the mean intra-cluster distance as a, then calculate the meanest nearest-cluster as b, and finally get the coefficient using the formula (b-a) / max(a,b). The closer the coefficient score is towards 1, the better because this means that the sample is more distinguishable between the neighbouring clusters.