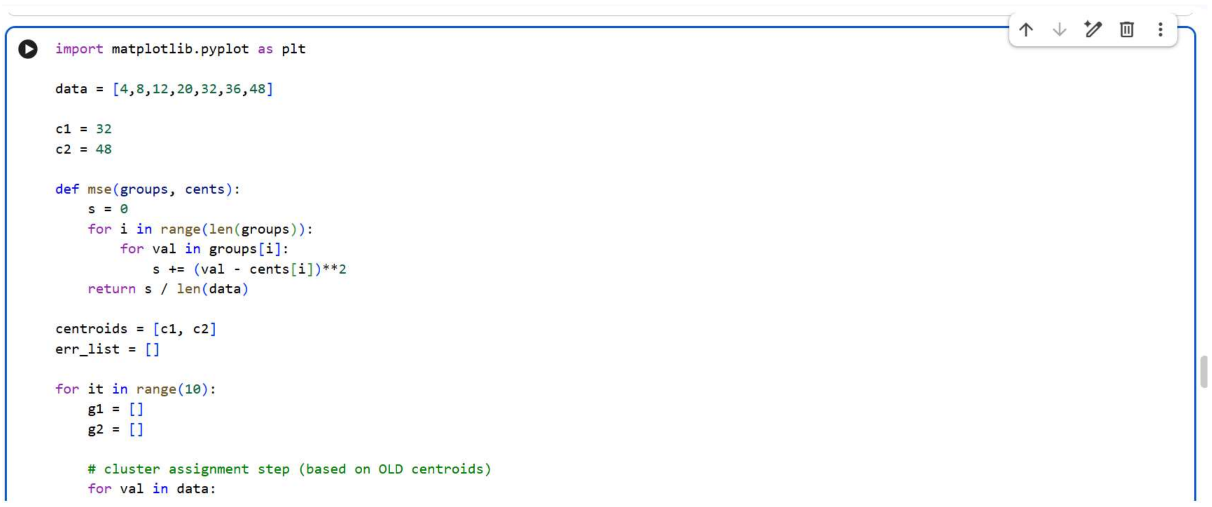
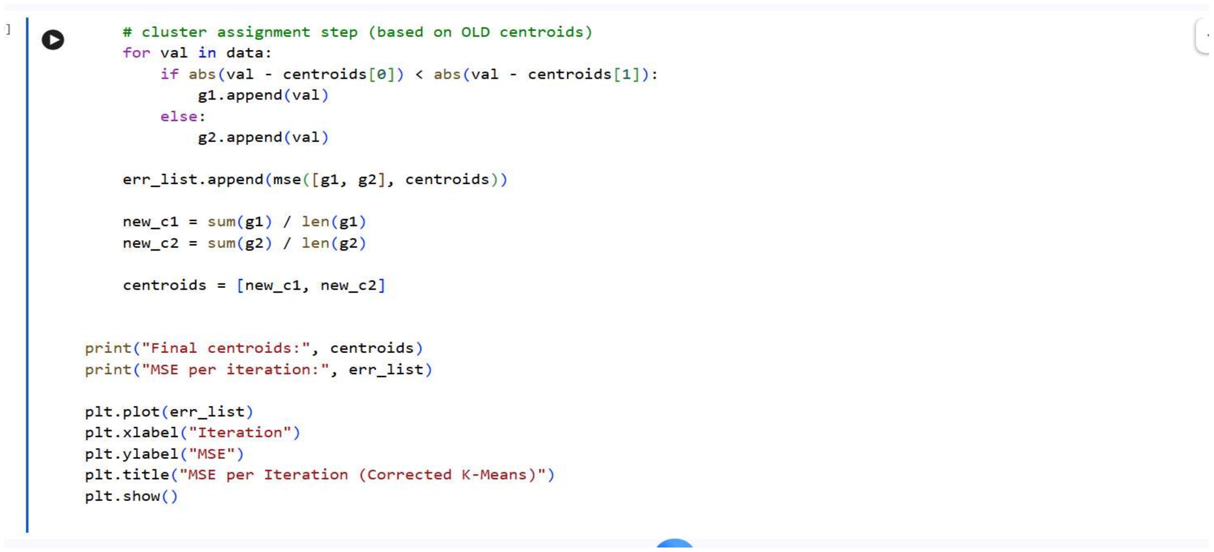
PRACTICAL-5

Apply simple K-means algorithm for clustering any dataset. Compare the performance of clusters by varying the algorithm parameters. For a given set of parameters, plot a line graph depicting MSE obtained after each iteration.

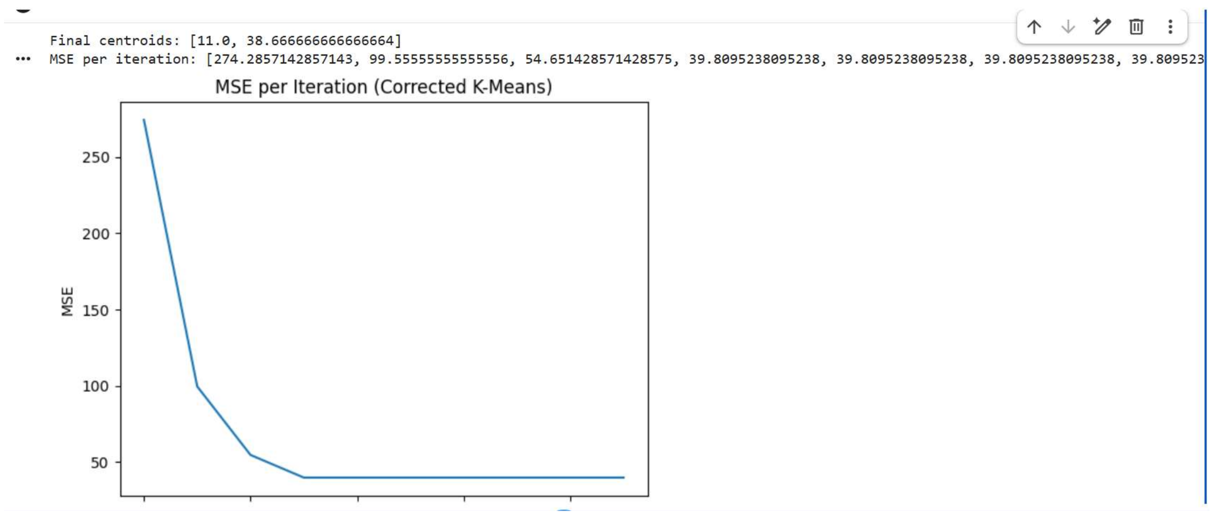
Output:



```python
import matplotlib.pyplot as plt

data = [4,8,12,20,32,36,48]

c1 = 32
c2 = 48

def mse(groups, cents):
    s = 0
    for i in range(len(groups)):
        for val in groups[i]:
            s += (val - cents[i])**2
    return s / len(data)

centroids = [c1, c2]
err_list = []

for it in range(10):
    g1 = []
    g2 = []

    # cluster assignment step (based on OLD centroids)
    for val in data:
```



```python
    # cluster assignment step (based on OLD centroids)
    for val in data:
        if abs(val - centroids[0]) < abs(val - centroids[1]):
            g1.append(val)
        else:
            g2.append(val)

    err_list.append(mse([g1, g2], centroids))

    new_c1 = sum(g1) / len(g1)
    new_c2 = sum(g2) / len(g2)

    centroids = [new_c1, new_c2]


print("Final centroids:", centroids)
print("MSE per iteration:", err_list)

plt.plot(err_list)
plt.xlabel("Iteration")
plt.ylabel("MSE")
plt.title("MSE per Iteration (Corrected K-Means)")
plt.show()
```



```
Final centroids: [11.0, 38.666666666666664]
MSE per iteration: [274.2857142857143, 99.55555555555556, 54.651428571428575, 39.8095238095238, 39.8095238095238, 39.8095238095238, 39.809523
```