Exercise 4: How many clusters of grain?

In the video, you learnt how to choose a good number of clusters for a dataset using the k-means inertia graph. You are given a dataset of the measurements of samples of grain. What's a good number of clusters in this case?

This dataset was obtained from the UCI.

From the course *Transition to Data Science*. Buy the entire course for just $10 for many more exercises and helpful video lectures.

Step 1: Load the dataset *(written for you)*.

```
In [6]: import pandas as pd

        seeds_df = pd.read_csv('../datasets/seeds.csv')
        # forget about the grain variety for the moment - we'll use this later
        del seeds_df['grain_variety']
```

Step 2: Display the DataFrame to inspect the data. Notice that there are 7 columns - so each grain sample (row) is a point in 7D space! Scatter plots can't help us here.

```
In [7]: seeds_df
```

Out[7]:

	area	perimeter	compactness	length	width	asymmetry_coefficient	\
0	15.26	14.84	0.8710	5.763	3.312	2.2210	
1	14.88	14.57	0.8811	5.554	3.333	1.0180	
2	14.29	14.09	0.9050	5.291	3.337	2.6990	
3	13.84	13.94	0.8955	5.324	3.379	2.2590	
4	16.14	14.99	0.9034	5.658	3.562	1.3550	
5	14.38	14.21	0.8951	5.386	3.312	2.4620	
6	14.69	14.49	0.8799	5.563	3.259	3.5860	
7	14.11	14.10	0.8911	5.420	3.302	2.7000	
8	16.63	15.46	0.8747	6.053	3.465	2.0400	
9	16.44	15.25	0.8880	5.884	3.505	1.9690	
10	15.26	14.85	0.8696	5.714	3.242	4.5430	
11	14.03	14.16	0.8796	5.438	3.201	1.7170	
12	13.89	14.02	0.8880	5.439	3.199	3.9860	
13	13.78	14.06	0.8759	5.479	3.156	3.1360	
14	13.74	14.05	0.8744	5.482	3.114	2.9320	
15	14.59	14.28	0.8993	5.351	3.333	4.1850	
16	13.99	13.83	0.9183	5.119	3.383	5.2340	
17	15.69	14.75	0.9058	5.527	3.514	1.5990	
18	14.70	14.21	0.9153	5.205	3.466	1.7670	
19	12.72	13.57	0.8686	5.226	3.049	4.1020	
20	14.16	14.40	0.8584	5.658	3.129	3.0720	

```
21   14.11      14.26       0.8722   5.520  3.168              2.6880
22   15.88      14.90       0.8988   5.618  3.507              0.7651
23   12.08      13.23       0.8664   5.099  2.936              1.4150
24   15.01      14.76       0.8657   5.789  3.245              1.7910
25   16.19      15.16       0.8849   5.833  3.421              0.9030
26   13.02      13.76       0.8641   5.395  3.026              3.3730
27   12.74      13.67       0.8564   5.395  2.956              2.5040
28   14.11      14.18       0.8820   5.541  3.221              2.7540
29   13.45      14.02       0.8604   5.516  3.065              3.5310
..     ...        ...          ...     ...    ...                 ...
180  11.41      12.95       0.8560   5.090  2.775              4.9570
181  12.46      13.41       0.8706   5.236  3.017              4.9870
182  12.19      13.36       0.8579   5.240  2.909              4.8570
183  11.65      13.07       0.8575   5.108  2.850              5.2090
184  12.89      13.77       0.8541   5.495  3.026              6.1850
185  11.56      13.31       0.8198   5.363  2.683              4.0620
186  11.81      13.45       0.8198   5.413  2.716              4.8980
187  10.91      12.80       0.8372   5.088  2.675              4.1790
188  11.23      12.82       0.8594   5.089  2.821              7.5240
189  10.59      12.41       0.8648   4.899  2.787              4.9750
190  10.93      12.80       0.8390   5.046  2.717              5.3980
191  11.27      12.86       0.8563   5.091  2.804              3.9850
192  11.87      13.02       0.8795   5.132  2.953              3.5970
193  10.82      12.83       0.8256   5.180  2.630              4.8530
194  12.11      13.27       0.8639   5.236  2.975              4.1320
195  12.80      13.47       0.8860   5.160  3.126              4.8730
196  12.79      13.53       0.8786   5.224  3.054              5.4830
197  13.37      13.78       0.8849   5.320  3.128              4.6700
198  12.62      13.67       0.8481   5.410  2.911              3.3060
199  12.76      13.38       0.8964   5.073  3.155              2.8280
200  12.38      13.44       0.8609   5.219  2.989              5.4720
201  12.67      13.32       0.8977   4.984  3.135              2.3000
202  11.18      12.72       0.8680   5.009  2.810              4.0510
203  12.70      13.41       0.8874   5.183  3.091              8.4560
204  12.37      13.47       0.8567   5.204  2.960              3.9190
205  12.19      13.20       0.8783   5.137  2.981              3.6310
206  11.23      12.88       0.8511   5.140  2.795              4.3250
207  13.20      13.66       0.8883   5.236  3.232              8.3150
208  11.84      13.21       0.8521   5.175  2.836              3.5980
209  12.30      13.34       0.8684   5.243  2.974              5.6370

     groove_length
0            5.220
```


1	4.956
2	4.825
3	4.805
4	5.175
5	4.956
6	5.219
7	5.000
8	5.877
9	5.533
10	5.314
11	5.001
12	4.738
13	4.872
14	4.825
15	4.781
16	4.781
17	5.046
18	4.649
19	4.914
20	5.176
21	5.219
22	5.091
23	4.961
24	5.001
25	5.307
26	4.825
27	4.869
28	5.038
29	5.097
..	...
180	4.825
181	5.147
182	5.158
183	5.135
184	5.316
185	5.182
186	5.352
187	4.956
188	4.957
189	4.794
190	5.045
191	5.001
192	5.132


```
193          5.089
194          5.012
195          4.914
196          4.958
197          5.091
198          5.231
199          4.830
200          5.045
201          4.745
202          4.828
203          5.000
204          5.001
205          4.870
206          5.003
207          5.056
208          5.044
209          5.063

[210 rows x 7 columns]
```

Step 3: Extract the measurements from the DataFrame using its `.values` attribute:

```
In [8]: samples = seeds_df.values
```

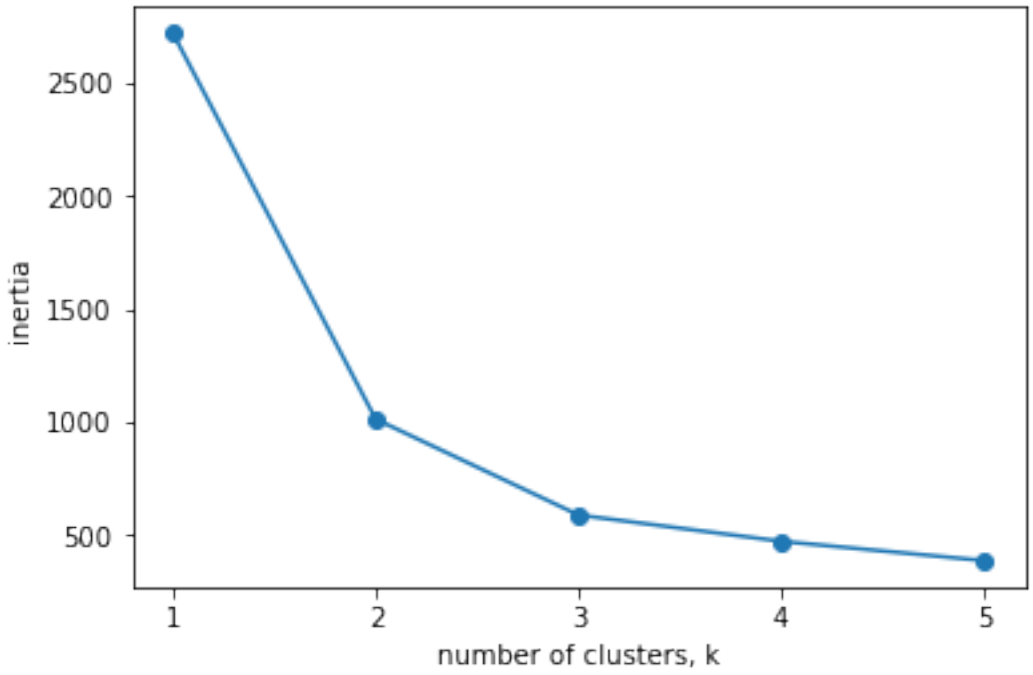
Step 4: *(Written for you)*. Measure the quality of clusterings with different numbers of clusters using the inertia. For each of the given values of `k`, perform the following steps:

- Create a `KMeans` instance called `model` with `k` clusters.
- Fit the model to the grain data `samples`.
- Append the value of the `inertia_` attribute of `model` to the list `inertias`.

```
In [9]: from sklearn.cluster import KMeans

        ks = range(1, 6)
        inertias = []

        for k in ks:
            # Create a KMeans instance with k clusters: model
            model = KMeans(n_clusters=k)

            # Fit model to samples
            model.fit(samples)

            # Append the inertia to the list of inertias
            inertias.append(model.inertia_)
```


Step 5: Plot the inertia to see which number of clusters is best. Remember: lower numbers are better!

In [10]:
```
import matplotlib.pyplot as plt

# Plot ks vs inertias
plt.plot(ks, inertias, '-o')
plt.xlabel('number of clusters, k')
plt.ylabel('inertia')
plt.xticks(ks)
plt.show()
```



Excellent work! You can see from the graph that 3 is a good number of clusters, since these are points where the inertia begins to decrease more slowly.

In [ ]:

