```python
# imports
import numpy as np
import matplotlib.pyplot as plt
import pandas as pd
%matplotlib inline
```

Load Dataset

In [3]:
```python
dataset = pd.read_csv("Iris.csv")
dataset.head()
```

Out[3]:

	Id	SepalLengthCm	SepalWidthCm	PetalLengthCm	PetalWidthCm	Species
0	1	5.1	3.5	1.4	0.2	Iris-setosa
1	2	4.9	3.0	1.4	0.2	Iris-setosa
2	3	4.7	3.2	1.3	0.2	Iris-setosa
3	4	4.6	3.1	1.5	0.2	Iris-setosa
4	5	5.0	3.6	1.4	0.2	Iris-setosa

In [4]:
```python
dataset.shape
```

Out[4]:
```
(150, 6)
```

Data Cleaning

In [5]:
```python
# species -> target variable which is not required in the clustering problem
# id -> also not useful
# so we eliminate the same
x = dataset.iloc[:,[1,2,3,4]].values
```

In [6]:
```python
x[:5]
```

Out[6]:
```
array([[5.1, 3.5, 1.4, 0.2],
       [4.9, 3. , 1.4, 0.2],
       [4.7, 3.2, 1.3, 0.2],
       [4.6, 3.1, 1.5, 0.2],
       [5. , 3.6, 1.4, 0.2]])
```

In [7]:
```python
target = dataset['Species']
```

In [8]:
```python
target
```

Out[8]:
```
0         Iris-setosa
1         Iris-setosa
2         Iris-setosa
3         Iris-setosa
4         Iris-setosa
            ...
145    Iris-virginica
146    Iris-virginica
147    Iris-virginica
148    Iris-virginica
149    Iris-virginica
Name: Species, Length: 150, dtype: object
```

Elbow Method and KMeans

In [9]:
```python
# Apply Kmeans to the dataset
from sklearn.cluster import KMeans
```

at any given number of clusters (k) we can find sum of squared distances.

the k with minimum sum of squared distances is the optimum value for the algorithm

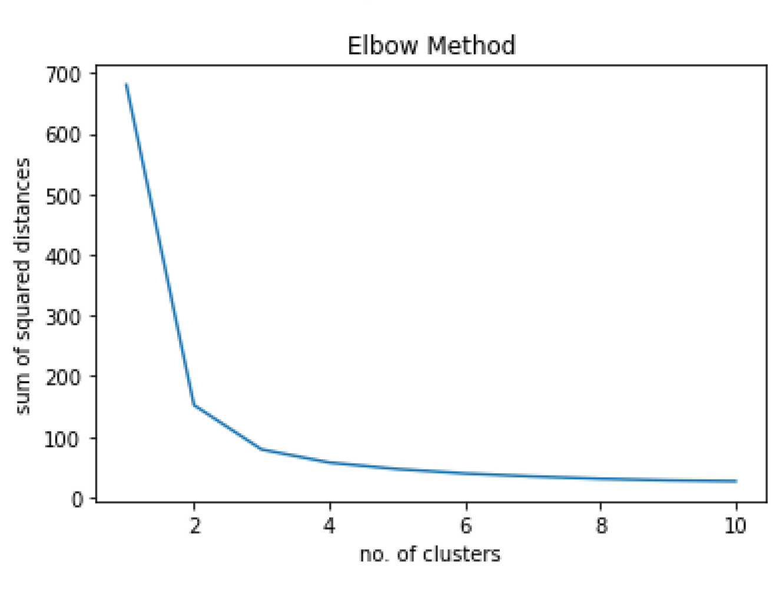
In [10]:
```python
sum_of_squared_distances = []
for i in range(1, 11):
    kmeans = KMeans(n_clusters = i)
    kmeans.fit(x)
    sum_of_squared_distances.append(kmeans.inertia_) #inertia_ attribute gives the sum of squared distances of

sum_of_squared_distances
```

```
C:\Users\Asus\anaconda3\lib\site-packages\sklearn\cluster\_kmeans.py:881: UserWarning: KMeans is known to have
a memory leak on Windows with MKL, when there are less chunks than available threads. You can avoid it by setti
ng the environment variable OMP_NUM_THREADS=1.
  warnings.warn(
```

Out[10]:
```
[680.8243999999996,
 152.36870647733915,
 78.94084142614601,
 57.317873214285726,
 46.535582051282034,
 39.251830892636775,
 34.299985543844244,
 30.496937562437576,
 27.8873865914787,
 26.436058441558448]
```

In [11]:
```python
# plot a graph between sum of squared distances and number of clusters
plt.plot(range(1,11), sum_of_squared_distances)
plt.title("Elbow Method")
plt.xlabel("no. of clusters")
plt.ylabel("sum of squared distances")
```

Out[11]:
```
Text(0, 0.5, 'sum of squared distances')
```



The optimum value is where the elbow occurs i.e, `3`, after this clusters sum doesnot decrease significantly

And we might also get extra labels for Y which is not necessary

In [12]:
```python
kmeans = KMeans(n_clusters = 3)
pred_y = kmeans.fit_predict(x) # this will fit and then predict
set(pred_y) # we have 3 classes and our clustering algorithm also predicted 3 labels
```

Out[12]:
```
{0, 1, 2}
```

In [13]:
```python
pred_y
```

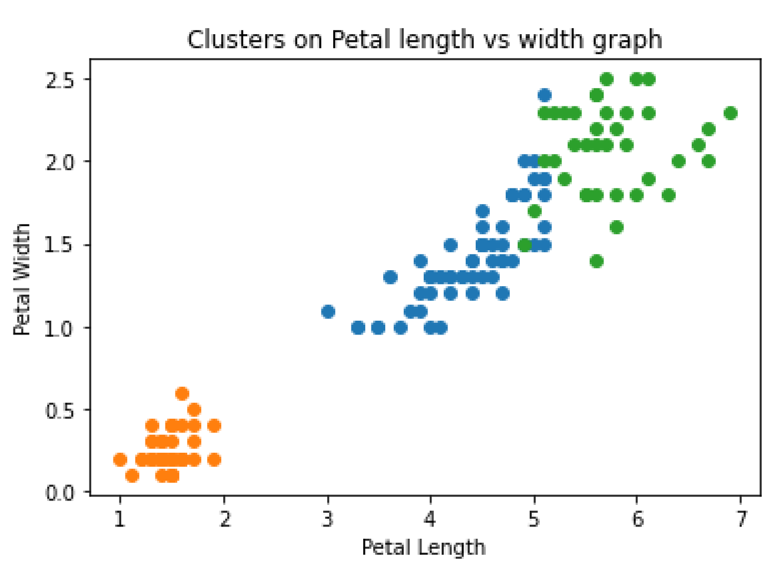
Out[13]:
```
array([1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1,
       1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1,
       1, 1, 1, 1, 1, 1, 0, 0, 2, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
       0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 2, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
       0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 2, 0, 2, 2, 2, 2, 0, 2, 2, 2,
       2, 2, 2, 0, 0, 2, 2, 2, 2, 0, 2, 0, 2, 0, 2, 2, 0, 0, 2, 2, 2, 2,
       2, 0, 2, 2, 2, 2, 0, 2, 2, 2, 0, 2, 2, 2, 0, 2, 2, 0])
```

In [14]:
```python
x[pred_y == 0, 2]
```

Out[14]:
```
array([4.7, 4.5, 4. , 4.6, 4.5, 4.7, 3.3, 4.6, 3.9, 3.5, 4.2, 4. , 4.7,
       3.6, 4.4, 4.5, 4.1, 4.5, 3.9, 4.8, 4. , 4.9, 4.7, 4.3, 4.4, 4.8,
       4.5, 3.5, 3.8, 3.7, 3.9, 5.1, 4.5, 4.5, 4.7, 4.4, 4.1, 4. , 4.4,
       4.6, 4. , 3.3, 4.2, 4.2, 4.2, 4.3, 3. , 4.1, 5.1, 4.5, 5. , 5.1,
       5. , 4.9, 4.9, 4.8, 4.9, 5.1, 4.8, 5.1, 5. , 5.1])
```

In [15]:
```python
# plot values between petal length and petal width
for i in set(pred_y):
    plt.scatter(x[pred_y == i, 2], x[pred_y == i, 3])
plt.xlabel("Petal Length")
plt.ylabel("Petal Width")
plt.title("Clusters on Petal length vs width graph")
```

Out[15]:
```
Text(0.5, 1.0, 'Clusters on Petal length vs width graph')
```



Hierarchical Clustering

Which linkage criterion to use.

    The linkage criterion determines which distance to use between sets of observation.

    The algorithm will merge the pairs of cluster that minimize this criterion.

'ward' minimizes the variance of the clusters being merged.

'average' uses the average of the distances of each observation of the two sets.

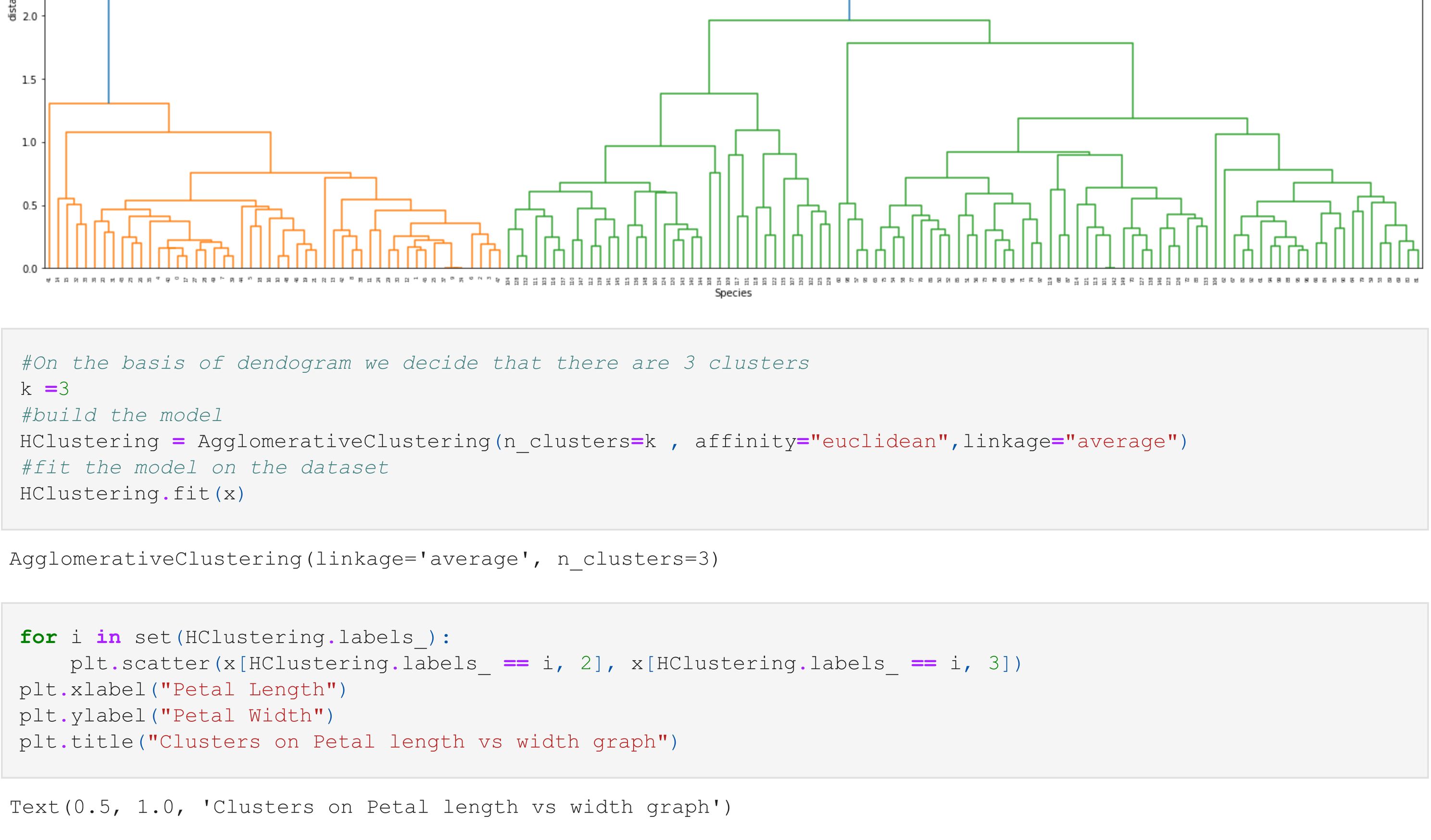
'complete' or 'maximum' linkage uses the maximum distances between all observations of the two sets.

single' uses the minimum of the distances between all observations of the two sets.

In [16]:
```python
from scipy.cluster.hierarchy import dendrogram, linkage
from sklearn.cluster import AgglomerativeClustering
import sklearn.metrics as sm
# generate the linkage matrix
Z = linkage(x, 'average')

Z
```

Out[16]:
```
array([[9.00000000e+00, 3.40000000e+01, 0.00000000e+00, 2.00000000e+00],
       [3.70000000e+01, 1.50000000e+02, 0.00000000e+00, 3.00000000e+00],
       [1.01000000e+02, 1.42000000e+02, 0.00000000e+00, 2.00000000e+00],
       [7.00000000e+00, 3.90000000e+01, 1.00000000e-01, 2.00000000e+00],
       [0.00000000e+00, 1.70000000e+01, 1.00000000e-01, 2.00000000e+00],
       [1.28000000e+02, 1.32000000e+02, 1.00000000e-01, 2.00000000e+00],
       [1.00000000e+01, 4.80000000e+01, 1.00000000e-01, 2.00000000e+00],
       [1.90000000e+01, 2.10000000e+01, 1.41421356e-01, 2.00000000e+00],
       [2.90000000e+01, 3.00000000e+01, 1.41421356e-01, 2.00000000e+00],
       [5.70000000e+01, 9.30000000e+01, 1.41421356e-01, 2.00000000e+00],
       [8.00000000e+01, 8.10000000e+01, 1.41421356e-01, 2.00000000e+00],
       [1.16000000e+02, 1.37000000e+02, 1.41421356e-01, 2.00000000e+00],
       [8.00000000e+00, 3.80000000e+01, 1.41421356e-01, 2.00000000e+00],
       [3.00000000e+00, 4.70000000e+01, 1.41421356e-01, 2.00000000e+00],
       [2.70000000e+01, 2.80000000e+01, 1.41421356e-01, 2.00000000e+00],
       [8.20000000e+01, 9.20000000e+01, 1.41421356e-01, 2.00000000e+00],
       [9.50000000e+01, 9.60000000e+01, 1.41421356e-01, 2.00000000e+00],
       [1.27000000e+02, 1.38000000e+02, 1.41421356e-01, 2.00000000e+00],
       [1.00000000e+00, 4.50000000e+01, 1.41421356e-01, 2.00000000e+00],
       [6.30000000e+01, 9.10000000e+01, 1.41421356e-01, 2.00000000e+00],
       [6.50000000e+01, 7.50000000e+01, 1.41421356e-01, 2.00000000e+00],
       [4.00000000e+01, 1.54000000e+02, 1.57313218e-01, 3.00000000e+00],
       [4.90000000e+01, 1.53000000e+02, 1.57313218e-01, 3.00000000e+00],
       [4.00000000e+00, 1.71000000e+02, 1.62610506e-01, 4.00000000e+00],
       [1.20000000e+01, 1.68000000e+02, 1.70710678e-01, 3.00000000e+00],
       [1.23000000e+02, 1.26000000e+02, 1.73205081e-01, 2.00000000e+00],
       [1.12000000e+02, 1.39000000e+02, 1.73205081e-01, 2.00000000e+00],
       [9.40000000e+01, 9.90000000e+01, 1.73205081e-01, 2.00000000e+00],
       [8.80000000e+01, 1.66000000e+02, 1.73205081e-01, 3.00000000e+00],
       [4.60000000e+01, 1.57000000e+02, 1.93185165e-01, 3.00000000e+00],
       [2.00000000e+00, 1.63000000e+02, 1.93185165e-01, 3.00000000e+00],
       [6.60000000e+01, 8.40000000e+01, 2.00000000e-01, 2.00000000e+00],
       [2.30000000e+01, 2.60000000e+01, 2.00000000e-01, 2.00000000e+00],
       [2.50000000e+01, 1.51000000e+02, 2.00000000e-01, 4.00000000e+00],
       [5.30000000e+01, 8.90000000e+01, 2.00000000e-01, 2.00000000e+00],
       [7.40000000e+01, 9.70000000e+01, 2.00000000e-01, 2.00000000e+00],
       [1.64000000e+02, 1.72000000e+02, 2.08943851e-01, 5.00000000e+00],
       [6.90000000e+01, 1.60000000e+02, 2.18890106e-01, 3.00000000e+00],
       [1.73000000e+02, 1.86000000e+02, 2.20051792e-01, 9.00000000e+00],
       [1.74000000e+02, 1.83000000e+02, 2.21380223e-01, 7.00000000e+00],
       [7.80000000e+01, 1.69000000e+02, 2.22474487e-01, 3.00000000e+00],
       [1.10000000e+02, 1.47000000e+02, 2.23606798e-01, 2.00000000e+00],
       [1.20000000e+02, 1.43000000e+02, 2.23606798e-01, 2.00000000e+00],
       [4.30000000e+01, 1.82000000e+02, 2.44090964e-01, 3.00000000e+00],
       [1.36000000e+02, 1.48000000e+02, 2.44948974e-01, 2.00000000e+00],
       [5.40000000e+01, 5.80000000e+01, 2.44948974e-01, 2.00000000e+00],
       [1.40000000e+02, 1.44000000e+02, 2.44948974e-01, 2.00000000e+00],
       [1.03000000e+02, 1.61000000e+02, 2.44948974e-01, 3.00000000e+00],
       [1.41000000e+02, 1.45000000e+02, 2.44948974e-01, 2.00000000e+00],
       [1.77000000e+02, 1.78000000e+02, 2.46912362e-01, 5.00000000e+00],
       [1.58000000e+02, 1.89000000e+02, 2.57943941e-01, 9.00000000e+00],
       [4.20000000e+01, 1.62000000e+02, 2.58113883e-01, 3.00000000e+00],
       [7.00000000e+01, 1.67000000e+02, 2.61803399e-01, 3.00000000e+00],
       [6.70000000e+01, 1.65000000e+02, 2.63895843e-01, 3.00000000e+00],
       [6.80000000e+01, 8.70000000e+01, 2.64575131e-01, 2.00000000e+00],
       [1.13000000e+02, 1.52000000e+02, 2.64575131e-01, 3.00000000e+00],
       [5.00000000e+01, 5.20000000e+01, 2.64575131e-01, 2.00000000e+00],
       [5.10000000e+01, 5.60000000e+01, 2.64575131e-01, 2.00000000e+00],
       [1.07000000e+02, 1.30000000e+02, 2.64575131e-01, 2.00000000e+00],
       [1.05000000e+02, 1.22000000e+02, 2.64575131e-01, 2.00000000e+00],
       [6.00000000e+00, 1.80000000e+02, 2.73281469e-01, 4.00000000e+00],
       [2.00000000e+01, 3.10000000e+01, 2.82842712e-01, 2.00000000e+00],
       [1.56000000e+02, 1.79000000e+02, 2.99436791e-01, 5.00000000e+00],
       [1.10000000e+01, 2.40000000e+01, 3.00000000e-01, 2.00000000e+00],
       [1.30000000e+01, 2.01000000e+02, 3.02528967e-01, 4.00000000e+00],
       [7.30000000e+01, 1.90000000e+02, 3.03635044e-01, 4.00000000e+00],
       [1.92000000e+02, 1.96000000e+02, 3.06803265e-01, 4.00000000e+00],
       [8.60000000e+01, 2.06000000e+02, 3.07252596e-01, 3.00000000e+00],
       [1.04000000e+02, 1.55000000e+02, 3.08113883e-01, 3.00000000e+00],
       [1.46000000e+02, 1.75000000e+02, 3.16123654e-01, 3.00000000e+00],
       [5.50000000e+01, 9.00000000e+01, 3.16227766e-01, 2.00000000e+00],
       [1.49000000e+02, 2.02000000e+02, 3.19875202e-01, 4.00000000e+00],
       [1.21000000e+02, 2.05000000e+02, 3.21372670e-01, 4.00000000e+00],
       [1.70000000e+02, 1.95000000e+02, 3.25417144e-01, 4.00000000e+00],
       [8.30000000e+01, 1.33000000e+02, 3.31662479e-01, 2.00000000e+00],
       [5.00000000e+00, 1.80000000e+01, 3.31662479e-01, 2.00000000e+00],
       [1.84000000e+02, 1.87000000e+02, 3.38826088e-01, 5.00000000e+00],
       [1.15000000e+02, 1.94000000e+02, 3.43649167e-01, 3.00000000e+00],
       [3.20000000e+01, 3.30000000e+01, 3.46410162e-01, 2.00000000e+00],
       [1.25000000e+02, 1.29000000e+02, 3.46410162e-01, 2.00000000e+00],
       [1.24000000e+02, 2.16000000e+02, 3.47598376e-01, 5.00000000e+00],
       [2.00000000e+02, 2.10000000e+02, 3.54359194e-01, 1.30000000e+01],
       [3.50000000e+01, 1.88000000e+02, 3.68235505e-01, 1.00000000e+01],
       [3.60000000e+01, 2.11000000e+02, 3.70245917e-01, 3.00000000e+00],
       [6.10000000e+01, 1.99000000e+02, 3.72863514e-01, 6.00000000e+00],
       [7.60000000e+01, 2.17000000e+02, 3.83692631e-01, 4.00000000e+00],
       [7.10000000e+01, 1.85000000e+02, 3.83776187e-01, 3.00000000e+00],
       [9.80000000e+01, 1.59000000e+02, 3.87298335e-01, 3.00000000e+00],
       [1.76000000e+02, 1.98000000e+02, 3.91079392e-01, 4.00000000e+00],
       [7.20000000e+01, 2.24000000e+02, 3.92409598e-01, 3.00000000e+00],
       [1.60000000e+01, 2.12000000e+02, 3.97231468e-01, 6.00000000e+00],
       [1.11000000e+02, 1.97000000e+02, 4.10951413e-01, 4.00000000e+00],
       [2.03000000e+02, 2.34000000e+02, 4.11551824e-01, 9.00000000e+00],
       [1.17000000e+02, 1.31000000e+02, 4.12310563e-01, 2.00000000e+00],
       [1.93000000e+02, 2.32000000e+02, 4.14095602e-01, 1.30000000e+01],
       [8.50000000e+01, 2.07000000e+02, 4.16211654e-01, 3.00000000e+00],
       [7.70000000e+01, 2.35000000e+02, 4.17858502e-01, 5.00000000e+00],
       [2.19000000e+02, 2.39000000e+02, 4.28466073e-01, 6.00000000e+00],
       [1.81000000e+02, 2.20000000e+02, 4.37885262e-01, 4.00000000e+00],
       [6.40000000e+01, 7.90000000e+01, 4.47213595e-01, 2.00000000e+00],
       [1.02000000e+02, 2.29000000e+02, 4.53456788e-01, 3.00000000e+00],
       [2.13000000e+02, 2.31000000e+02, 4.58484122e-01, 1.50000000e+01],
       [2.18000000e+02, 2.41000000e+02, 4.62587845e-01, 7.00000000e+00],
       [2.25000000e+02, 2.40000000e+02, 4.63899890e-01, 8.00000000e+00],
       [2.33000000e+02, 2.44000000e+02, 4.68197902e-01, 1.60000000e+01],
       [1.91000000e+02, 2.38000000e+02, 4.73325708e-01, 6.00000000e+00],
       [1.18000000e+02, 2.09000000e+02, 4.80016560e-01, 3.00000000e+00],
       [4.40000000e+01, 2.53000000e+02, 4.90701246e-01, 9.00000000e+00],
       [2.23000000e+02, 2.46000000e+02, 4.95121073e-01, 9.00000000e+00],
       [2.08000000e+02, 2.50000000e+02, 4.96264997e-01, 5.00000000e+00],
       [1.14000000e+02, 2.22000000e+02, 5.07329273e-01, 5.00000000e+00],
       [1.50000000e+01, 2.28000000e+02, 5.08149490e-01, 3.00000000e+00],
       [2.15000000e+02, 2.36000000e+02, 5.11495126e-01, 7.00000000e+00],
       [6.00000000e+01, 2.37000000e+02, 5.13307651e-01, 4.00000000e+00],
       [5.90000000e+01, 2.26000000e+02, 5.18204344e-01, 6.00000000e+00],
       [2.42000000e+02, 2.48000000e+02, 5.25474630e-01, 1.30000000e+01],
       [2.54000000e+02, 2.57000000e+02, 5.39202268e-01, 2.50000000e+01],
       [2.14000000e+02, 2.51000000e+02, 5.47272493e-01, 1.90000000e+01],
       [1.40000000e+01, 2.61000000e+02, 5.48532860e-01, 4.00000000e+00],
       [2.21000000e+02, 2.47000000e+02, 5.49450670e-01, 1.00000000e+01],
       [2.49000000e+02, 2.64000000e+02, 5.70834015e-01, 8.00000000e+00],
       [2.45000000e+02, 2.62000000e+02, 5.94899385e-01, 1.00000000e+01],
       [1.00000000e+02, 2.30000000e+02, 6.02494379e-01, 6.00000000e+00],
       [2.52000000e+02, 2.55000000e+02, 6.03519420e-01, 1.30000000e+01],
       [2.27000000e+02, 2.72000000e+02, 6.06291352e-01, 9.00000000e+00],
       [1.19000000e+02, 2.04000000e+02, 6.22811631e-01, 3.00000000e+00],
       [2.60000000e+02, 2.69000000e+02, 6.41198366e-01, 1.50000000e+01],
       [2.73000000e+02, 2.74000000e+02, 6.74792578e-01, 2.20000000e+01],
       [2.65000000e+02, 2.70000000e+02, 6.77553240e-01, 2.10000000e+01],
       [1.35000000e+02, 2.59000000e+02, 7.06547529e-01, 6.00000000e+00],
       [2.58000000e+02, 2.71000000e+02, 7.14843442e-01, 1.90000000e+01],
       [2.20000000e+01, 2.67000000e+02, 7.20501100e-01, 2.00000000e+01],
       [1.08000000e+02, 1.34000000e+02, 7.54983444e-01, 2.00000000e+00],
       [2.66000000e+02, 2.81000000e+02, 7.55184930e-01, 4.50000000e+01],
       [6.20000000e+01, 2.78000000e+02, 7.83078008e-01, 2.20000000e+01],
       [2.75000000e+02, 2.76000000e+02, 8.93789321e-01, 1.80000000e+01],
       [1.09000000e+02, 2.43000000e+02, 8.96485216e-01, 3.00000000e+00],
       [2.56000000e+02, 2.79000000e+02, 9.03309021e-01, 9.00000000e+00],
       [2.80000000e+02, 2.85000000e+02, 9.17272544e-01, 3.70000000e+01],
       [2.77000000e+02, 2.82000000e+02, 9.68739482e-01, 2.40000000e+01],
       [1.06000000e+02, 2.84000000e+02, 1.05978781e+00, 2.30000000e+01],
       [2.68000000e+02, 2.83000000e+02, 1.07787832e+00, 4.90000000e+01],
       [2.86000000e+02, 2.87000000e+02, 1.09005269e+00, 1.20000000e+01],
       [2.88000000e+02, 2.90000000e+02, 1.18677850e+00, 6.00000000e+01],
       [4.10000000e+01, 2.91000000e+02, 1.30553087e+00, 5.00000000e+01],
       [2.89000000e+02, 2.92000000e+02, 1.38099374e+00, 3.60000000e+01],
       [2.63000000e+02, 2.93000000e+02, 1.78556648e+00, 6.40000000e+01],
       [2.95000000e+02, 2.96000000e+02, 1.96361409e+00, 1.00000000e+02],
       [2.94000000e+02, 2.97000000e+02, 4.06041346e+00, 1.50000000e+02]])
```

In [17]:
```python
# set cut-off to 150
max_d = 7.08                # max_d as in max_distance

plt.figure(figsize=(25, 10))
plt.title('Iris Hierarchical Clustering Dendrogram')
plt.xlabel('Species')
plt.ylabel('distance')
dendrogram(
    Z      # font size for the x axis labels
)
plt.axhline(y=max_d, c='k')
plt.show()
```

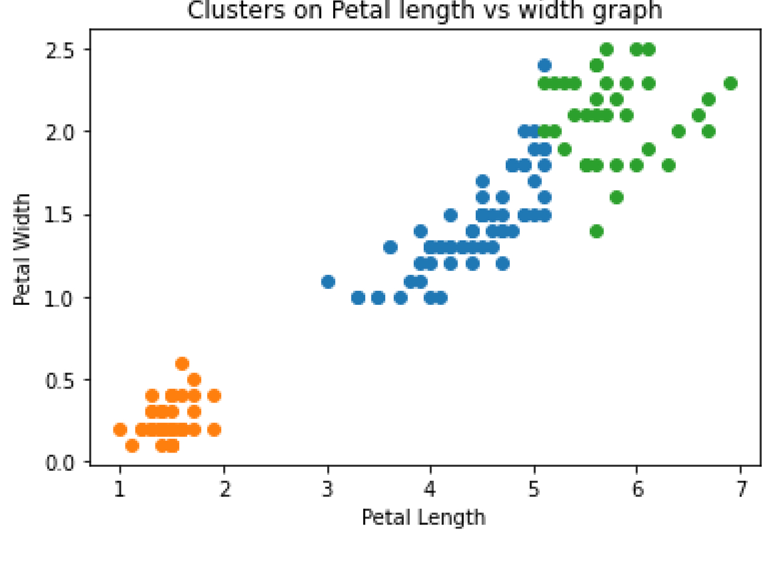


In [18]:
```python
#On the basis of dendogram we decide that there are 3 clusters
k =3
#build the model
HClustering = AgglomerativeClustering(n_clusters=k , affinity="euclidean",linkage="average")
#fit the model on the dataset
HClustering.fit(x)
```

Out[18]:
```
AgglomerativeClustering(linkage='average', n_clusters=3)
```

In [19]:
```python
for i in set(HClustering.labels_):
    plt.scatter(x[HClustering.labels_ == i, 2], x[HClustering.labels_ == i, 3])
plt.xlabel("Petal Length")
plt.ylabel("Petal Width")
plt.title("Clusters on Petal length vs width graph")
```

Out[19]:
```
Text(0.5, 1.0, 'Clusters on Petal length vs width graph')
```



*** HAPPY LEARNING **