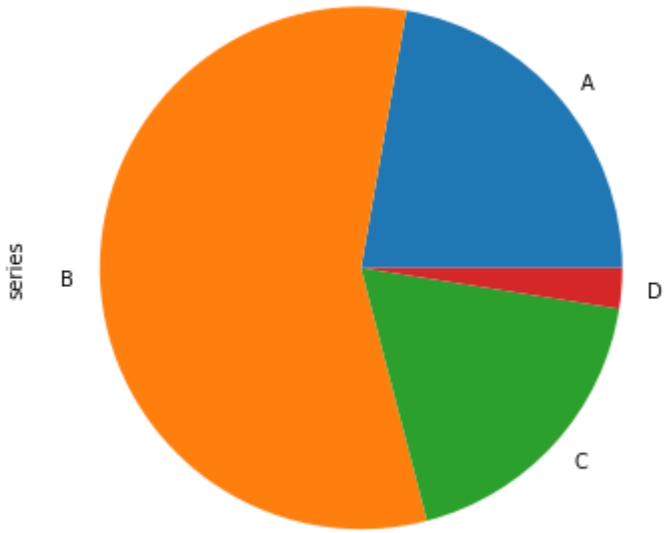
In [1]:

```
# Pie Charts
# you will learn how to create pie charts to visualize datasets.

%matplotlib inline
import matplotlib.pyplot as plt
import pandas as pd
import numpy as np
series = pd.Series(3 * np.random.rand(4),
                   index=['A', 'B', 'C', 'D'],
                   name='series')
print(series)
# You can visualize it as follows:
plt.figure()
series.plot.pie(figsize=(6, 6))
plt.show()
```

```
A    0.958683
B    2.451032
C    0.795151
D    0.107733
Name: series, dtype: float64
```

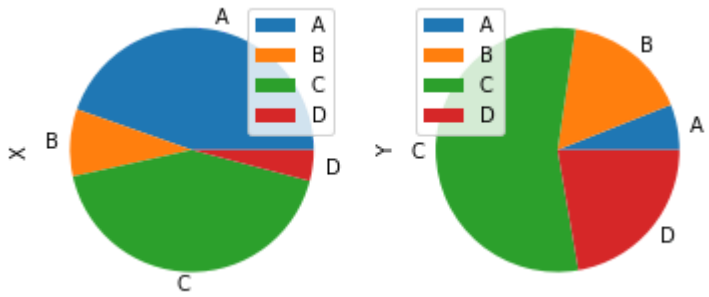


In [5]:

```
%matplotlib inline
import matplotlib.pyplot as plt
import pandas as pd
import numpy as np
# Let's create a dataset with two columns as follows:
df = pd.DataFrame(3 * np.random.rand(4, 2),
                  index=['A', 'B', 'C', 'D'],
                  columns=['X', 'Y'])
print(df)
# You can visualize it as follows:
plt.figure()
df.plot.pie(figsize=(6, 6), subplots=True)
plt.show()
```

```
          X         Y
A  2.118874  0.239624
B  0.422719  0.660457
C  2.014499  2.187678
D  0.194507  0.884079
```

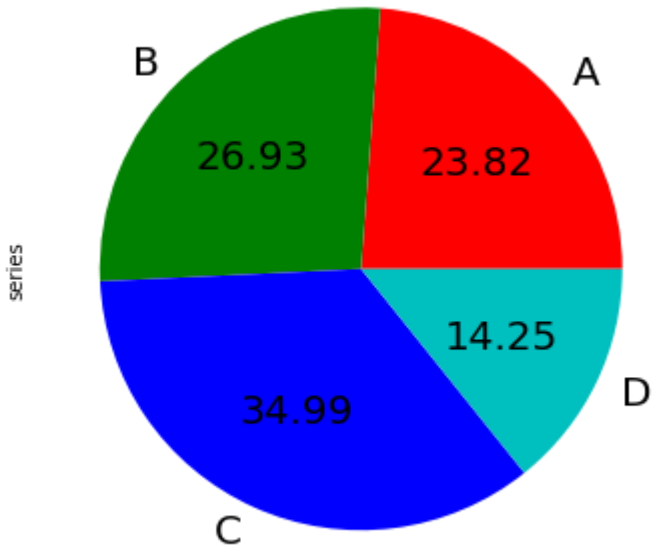
```
<Figure size 432x288 with 0 Axes>
```



In [6]:

```
%matplotlib inline
import matplotlib.pyplot as plt
import pandas as pd
import numpy as np
series = pd.Series(3 * np.random.rand(4),
                   index=['A', 'B', 'C', 'D'],
                   name='series')
print(series)
# You can customize pie charts. Specifically, you can customize the font,
# colors, and labels as follows:
plt.figure()
series.plot.pie(labels=['A', 'B', 'C', 'D'],
                colors=['r', 'g', 'b', 'c'],
                autopct='%.2f', fontsize=20,
figsize=(6, 6))
plt.show()
```

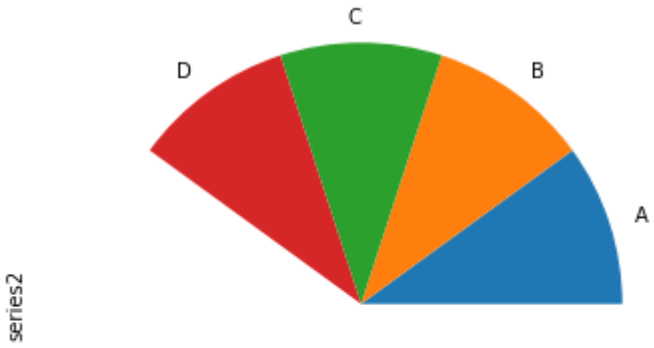
```
A    1.742623
B    1.970329
C    2.559763
D    1.042423
Name: series, dtype: float64
```



In [7]:

```
# Let's create a partial pie chart by passing values whose sum is less
# than 1.0. The following is the data for that:

%matplotlib inline
import matplotlib.pyplot as plt
import pandas as pd
import numpy as np
series = pd.Series([0.1] * 4,
                   index=['A', 'B', 'C', 'D'],
                   name='series2')
print(series)
# The partial pie chart can be visualized as follows:
plt.figure()
series.plot.pie(figsize=(6, 6))
plt.show()
```

```
A    0.1
B    0.1
C    0.1
D    0.1
Name: series2, dtype: float64
```



In [9]:

```
# Data Visualization with Seaborn
# What is seaborn?
# Seaborn is based on and built on top of Matplotlib. It provides a lot
# of functionality for drawing attractive graphics. It has built-in
# support for the series and dataframe data structures in Pandas
# and for Ndarrays in NumPy.The following command returns the list of
# all the built-in sample dataframes:

import seaborn as sns
sns.get_dataset_names()
```

Out[9]:

```
['anagrams',
 'anscombe',
 'attention',
 'brain_networks',
 'car_crashes',
 'diamonds',
 'dots',
 'exercise',
 'flights',
 'fmri',
 'gammas',
 'geyser',
 'iris',
 'mpg',
 'penguins',
 'planets',
 'taxis',
 'tips',
 'titanic']
```

In [10]:

```
# You can load these dataframes into Python variables as follows:
iris = sns.load_dataset('iris')
```

In [11]:

```
# Let's see the data stored in the iris dataset with the following
# statement:
iris
```

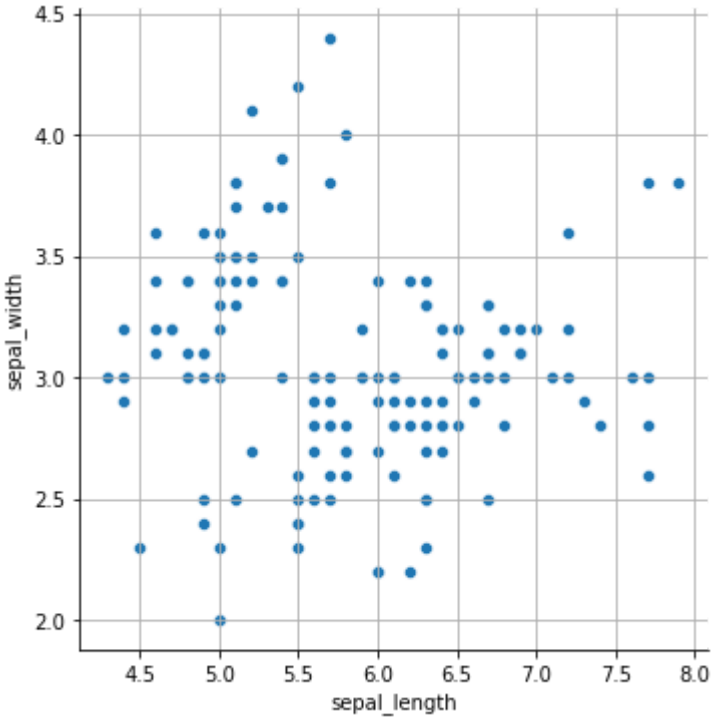
Out[11]:

	sepal_length	sepal_width	petal_length	petal_width	species
0	5.1	3.5	1.4	0.2	setosa
1	4.9	3.0	1.4	0.2	setosa
2	4.7	3.2	1.3	0.2	setosa
3	4.6	3.1	1.5	0.2	setosa
4	5.0	3.6	1.4	0.2	setosa
...	...	...	...	...	...
145	6.7	3.0	5.2	2.3	virginica
146	6.3	2.5	5.0	1.9	virginica
147	6.5	3.0	5.2	2.0	virginica
148	6.2	3.4	5.4	2.3	virginica
149	5.9	3.0	5.1	1.8	virginica

150 rows × 5 columns

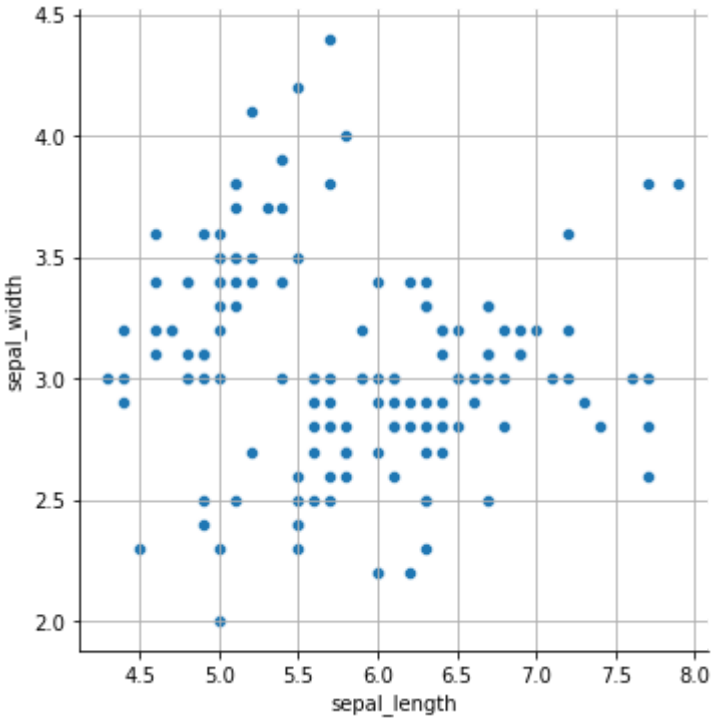
In [13]:

```
# Plotting Statistical Relationships
# You can plot the statistical relationship between two variables with
# various functions in Seaborn. The general plotting function to do this
# is relplot(). You can plot various types of data with this function.
# By default, the relplot() function plots a scatter plot.
# Here is an example:

import matplotlib.pyplot as plt
import seaborn as sns
sns.relplot(x='sepal_length',
            y='sepal_width',
            data=iris)
plt.grid('on')
plt.show()
```



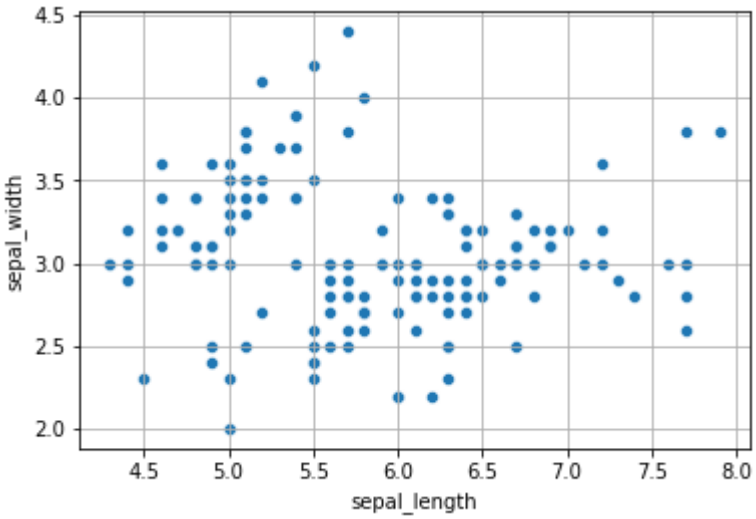
In [14]:

```
# You can explicitly specify the type of plot as follows:

import matplotlib.pyplot as plt
import seaborn as sns
sns.relplot(x='sepal_length',
            y='sepal_width',
            data=iris,
            kind='scatter')
plt.grid('on')
plt.show()
```



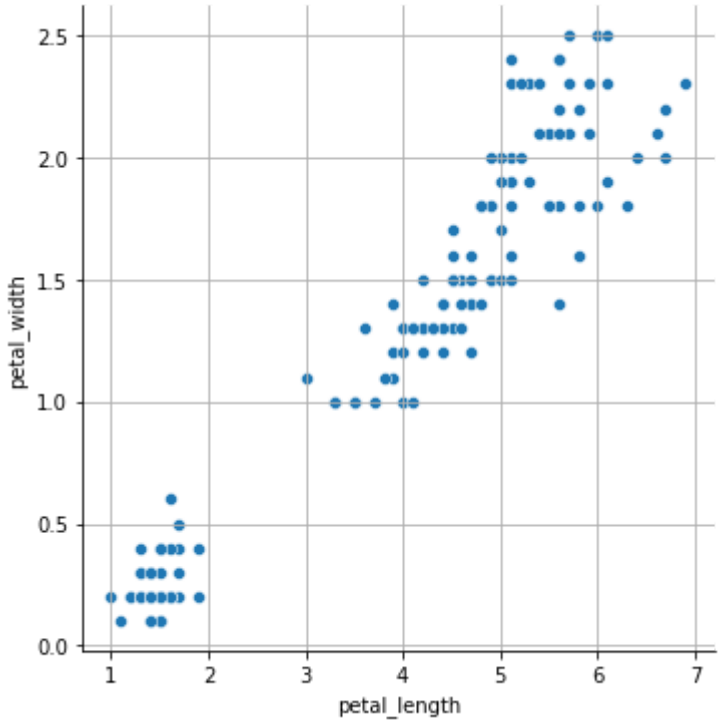
In [15]:

```
# The function replot() is a generic function where you can pass an
# argument to specify the type of plot. You can also create a scatter
# plot with the function scatterplot().

import matplotlib.pyplot as plt
import seaborn as sns
sns.scatterplot(x='sepal_length',
                y='sepal_width',
                data=iris)
plt.grid('on')
plt.show()
```



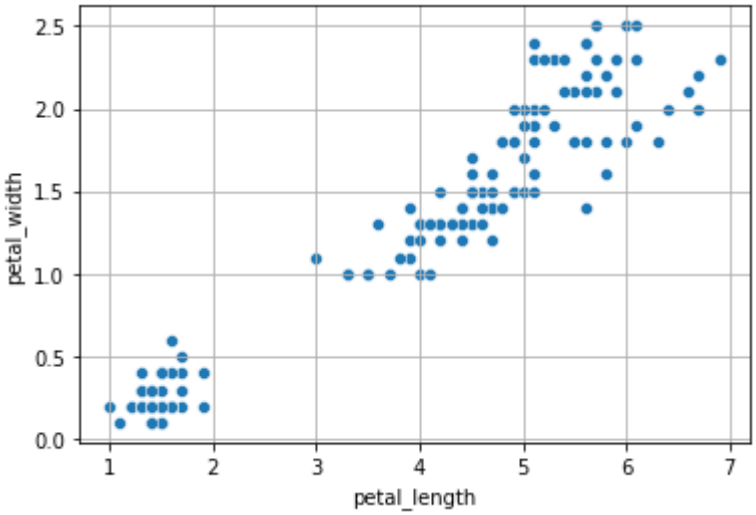
In [16]:

```
# You can feed some other columns of the dataset to the plotting
# function as follows:

import matplotlib.pyplot as plt
import seaborn as sns
sns.relplot(x='petal_length',
            y='petal_width',
            data=iris)
plt.grid('on')
plt.show()
```



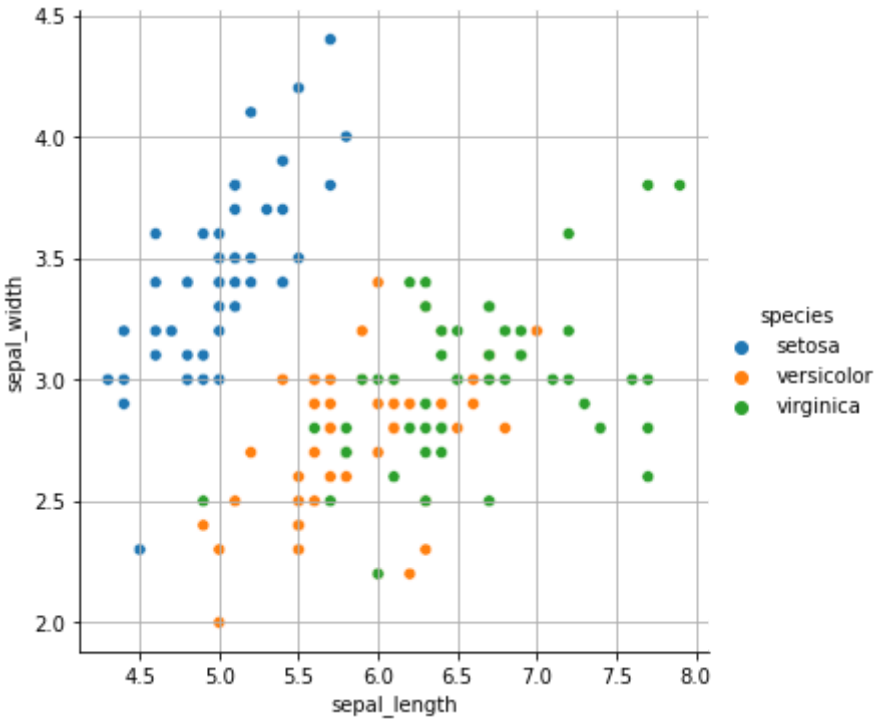
In [17]:

```
# You can also write this with the scatterplot() function as follows:

import matplotlib.pyplot as plt
import seaborn as sns
sns.scatterplot(x='petal_length',
                y='petal_width',
                data=iris)
plt.grid('on')
plt.show()
```



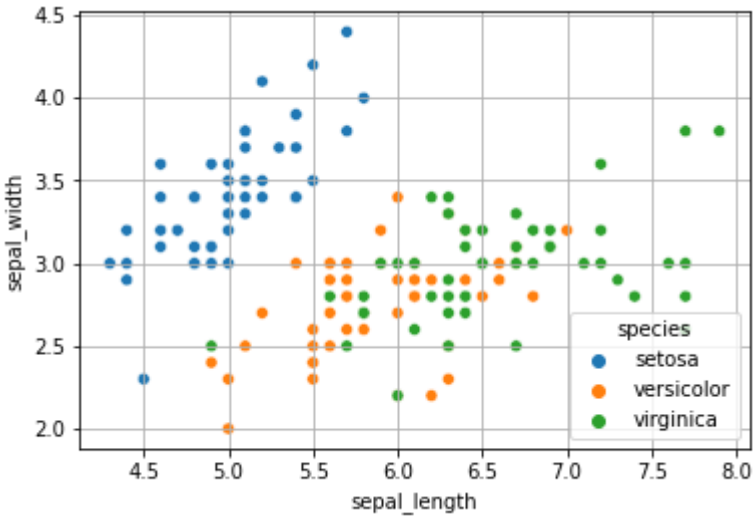
In [18]:

```
# You can customize the plot and show an additional column with color
# coding as follows:

import matplotlib.pyplot as plt
import seaborn as sns
sns.relplot(x='sepal_length',
            y='sepal_width',
            hue='species',
            data=iris)
plt.grid('on')
plt.show()
```



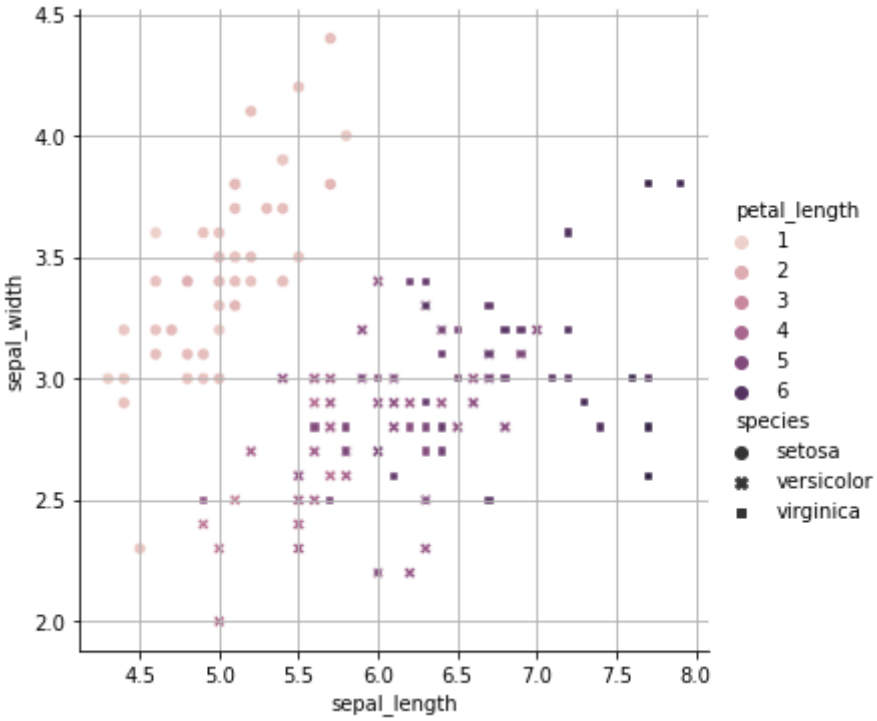
In [19]:

```
# You get the same result as;

import matplotlib.pyplot as plt
import seaborn as sns
sns.scatterplot(x='sepal_length',
                y='sepal_width',
                hue='species',
                data=iris)
plt.grid('on')
plt.show()
```



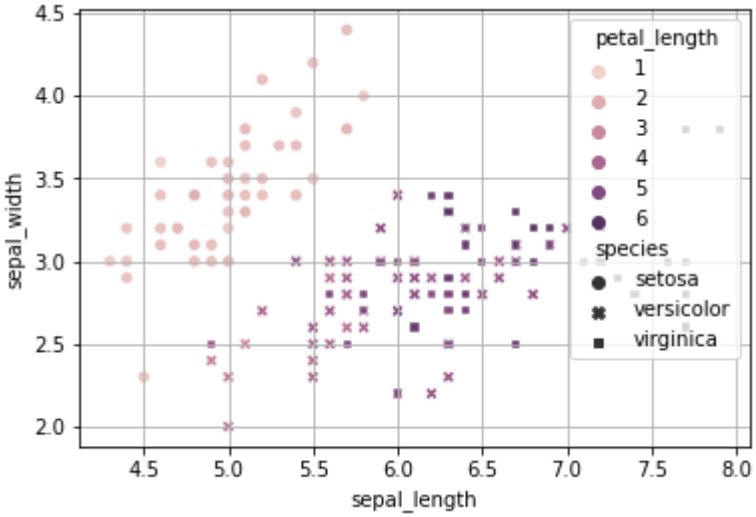
In [20]:

```
# You can also assign the styles to the scatter plot data points
# (markers) as follows:

import matplotlib.pyplot as plt
import seaborn as sns
sns.relplot(x='sepal_length', y='sepal_width',
            hue='petal_length', style='species',
            data=iris)
plt.grid('on')
plt.show()
```



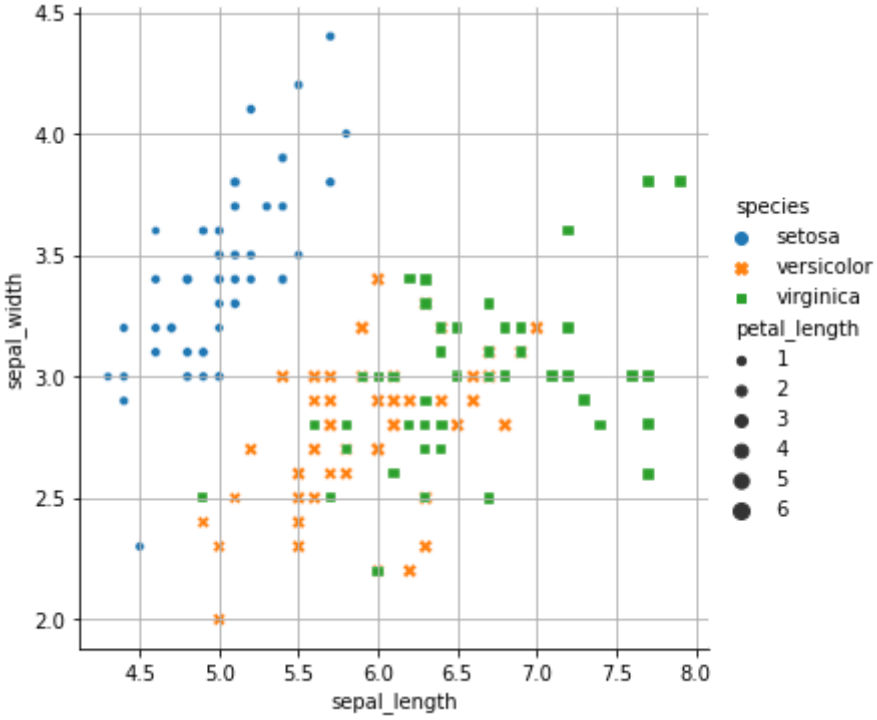
In [21]:

```
# The following code produces the same output:

import matplotlib.pyplot as plt
import seaborn as sns
sns.scatterplot(x='sepal_length', y='sepal_width',
                hue='petal_length', style='species',
                data=iris)
plt.grid('on')
plt.show()
```



In [22]:

```
# You can also adjust the sizes of the markers as follows:

import matplotlib.pyplot as plt
import seaborn as sns
sns.relplot(x='sepal_length', y='sepal_width',
            size='petal_length', style='species',
            hue='species', data=iris)
plt.grid('on')
plt.show()
```



In [23]:

```
# The following code produces the same result:

import matplotlib.pyplot as plt
import seaborn as sns
sns.scatterplot(x='sepal_length', y='sepal_width',
                size='petal_length', style='species',
                hue='species', data=iris)
plt.grid('on')
plt.show()
```