In [2]:
```
pip install pandas
```

```
Requirement already satisfied: pandas in c:\users\sanka\anaconda3\lib\site-pa
ckages (1.1.3)Note: you may need to restart the kernel to use updated package
s.
Requirement already satisfied: numpy>=1.15.4 in c:\users\sanka\anaconda3\lib
\site-packages (from pandas) (1.19.2)
Requirement already satisfied: python-dateutil>=2.7.3 in c:\users\sanka\anaco
nda3\lib\site-packages (from pandas) (2.8.1)
Requirement already satisfied: pytz>=2017.2 in c:\users\sanka\anaconda3\lib\s
ite-packages (from pandas) (2020.1)
Requirement already satisfied: six>=1.5 in c:\users\sanka\anaconda3\lib\site-
packages (from python-dateutil>=2.7.3->pandas) (1.15.0)
```

In [3]:
```
pip install numpy
```

```
Requirement already satisfied: numpy in c:\users\sanka\anaconda3\lib\site-pac
kages (1.19.2)
Note: you may need to restart the kernel to use updated packages.
```

In [4]:
```
import pandas as pd
from pandas import read_csv
import numpy as np
import seaborn as sns
import matplotlib as plt
%matplotlib inline
```

In [5]:
```
df=pd.read_csv("IRISS.csv")
df
```

Out[5]:

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

150 rows × 5 columns

In [6]:
```
df.shape #structure od dataset
```

Out[6]:
```
(150, 5)
```

In [7]:
```
df.tail()
```

Out[7]:

	sepal_length	sepal_width	petal_length	petal_width	species
145	6.7	3.0	5.2	2.3	Iris-virginica
146	6.3	2.5	5.0	1.9	Iris-virginica
147	6.5	3.0	5.2	2.0	Iris-virginica
148	6.2	3.4	5.4	2.3	Iris-virginica
149	5.9	3.0	5.1	1.8	Iris-virginica

In [8]:
```
df.head()  #displays first 5 rows
```

Out[8]:

	sepal_length	sepal_width	petal_length	petal_width	species
0	5.1	3.5	1.4	0.2	Iris-setosa
1	4.9	3.0	1.4	0.2	Iris-setosa
2	4.7	3.2	1.3	0.2	Iris-setosa
3	4.6	3.1	1.5	0.2	Iris-setosa
4	5.0	3.6	1.4	0.2	Iris-setosa

In [9]:
```
df.info() #provide summary of dataframe
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 150 entries, 0 to 149
Data columns (total 5 columns):
 #   Column        Non-Null Count  Dtype  
---  ------        --------------  -----  
 0   sepal_length  150 non-null    float64
 1   sepal_width   150 non-null    float64
 2   petal_length  150 non-null    float64
 3   petal_width   150 non-null    float64
 4   species       150 non-null    object 
dtypes: float64(4), object(1)
memory usage: 6.0+ KB
```

In [10]:
```
df.describe()
```

Out[10]:

	sepal_length	sepal_width	petal_length	petal_width
count	150.000000	150.000000	150.000000	150.000000
mean	5.843333	3.054000	3.758667	1.198667
std	0.828066	0.433594	1.764420	0.763161
min	4.300000	2.000000	1.000000	0.100000
25%	5.100000	2.800000	1.600000	0.300000
50%	5.800000	3.000000	4.350000	1.300000
75%	6.400000	3.300000	5.100000	1.800000
max	7.900000	4.400000	6.900000	2.500000

In [12]:
```
df["sepal_length"].count()
```

Out[12]:
```
150
```

In [19]:
```
#creating subsets
df1 = df[['sepal_length','petal_length']].loc[0:15]
df1
```

Out[19]:

	sepal_length	petal_length
0	5.1	1.4
1	4.9	1.4
2	4.7	1.3
3	4.6	1.5
4	5.0	1.4
5	5.4	1.7
6	4.6	1.4
7	5.0	1.5
8	4.4	1.4
9	4.9	1.5
10	5.4	1.5
11	4.8	1.6
12	4.8	1.4
13	4.3	1.1
14	5.8	1.2
15	5.7	1.5

In [20]:
```
trans=df.transpose()
trans
```

Out[20]:

	0	1	2	3	4	5	6	7	8	9	...	
sepal_length	5.1	4.9	4.7	4.6	5	5.4	4.6	5	4.4	4.9	...	
sepal_width	3.5	3	3.2	3.1	3.6	3.9	3.4	3.4	2.9	3.1	...	
petal_length	1.4	1.4	1.3	1.5	1.4	1.7	1.4	1.5	1.4	1.5	...	
petal_width	0.2	0.2	0.2	0.2	0.2	0.4	0.3	0.2	0.2	0.1	...	
species	Iris-setosa	Iris-setosa	Iris-setosa	Iris-setosa	Iris-setosa	Iris-setosa	Iris-setosa	Iris-setosa	Iris-setosa	Iris-setosa	...	virg

5 rows × 150 columns

In [22]:
```
df['sepal_length'].unique()
```

Out[22]:
```
array([5.1, 4.9, 4.7, 4.6, 5. , 5.4, 4.4, 4.8, 4.3, 5.8, 5.7, 5.2, 5.5,
       4.5, 5.3, 7. , 6.4, 6.9, 6.5, 6.3, 6.6, 5.9, 6. , 6.1, 5.6, 6.7,
       6.2, 6.8, 7.1, 7.6, 7.3, 7.2, 7.7, 7.4, 7.9])
```

In [23]:
```
df['petal_length'].unique()
```

Out[23]:
```
array([1.4, 1.3, 1.5, 1.7, 1.6, 1.1, 1.2, 1. , 1.9, 4.7, 4.5, 4.9, 4. ,
       4.6, 3.3, 3.9, 3.5, 4.2, 3.6, 4.4, 4.1, 4.8, 4.3, 5. , 3.8, 3.7,
       5.1, 3. , 6. , 5.9, 5.6, 5.8, 6.6, 6.3, 6.1, 5.3, 5.5, 6.7, 6.9,
       5.7, 6.4, 5.4, 5.2])
```

In [24]:
```
df.isnull().sum() # checking missing values
```

Out[24]:
```
sepal_length    0
sepal_width     0
petal_length    0
petal_width     0
species         0
dtype: int64
```

In [25]:
```
df.describe().style.background_gradient(cmap="Greens")
```

Out[25]:

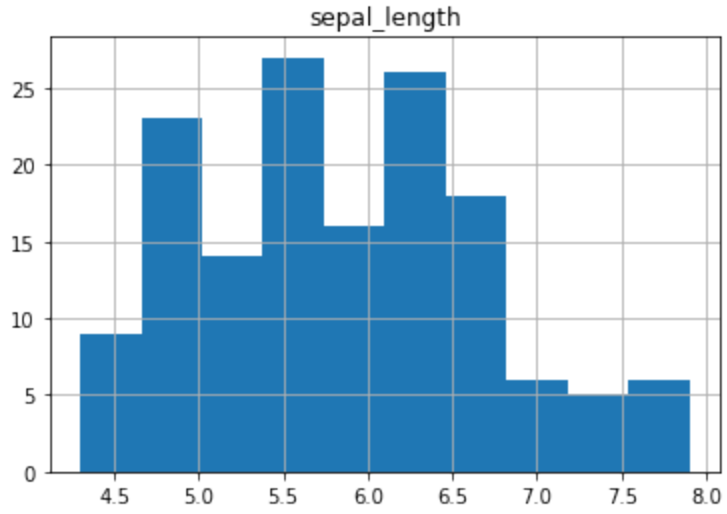
	sepal_length	sepal_width	petal_length	petal_width
count	150.000000	150.000000	150.000000	150.000000
mean	5.843333	3.054000	3.758667	1.198667
std	0.828066	0.433594	1.764420	0.763161
min	4.300000	2.000000	1.000000	0.100000
25%	5.100000	2.800000	1.600000	0.300000
50%	5.800000	3.000000	4.350000	1.300000
75%	6.400000	3.300000	5.100000	1.800000
max	7.900000	4.400000	6.900000	2.500000

In [28]:
```
df.describe(include="all")
```

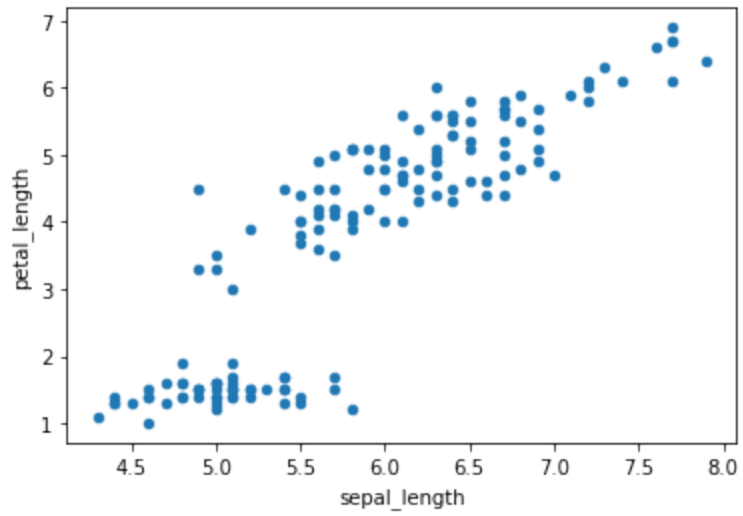
Out[28]:

	sepal_length	sepal_width	petal_length	petal_width	species
count	150.000000	150.000000	150.000000	150.000000	150
unique	NaN	NaN	NaN	NaN	3
top	NaN	NaN	NaN	NaN	Iris-virginica
freq	NaN	NaN	NaN	NaN	50
mean	5.843333	3.054000	3.758667	1.198667	NaN
std	0.828066	0.433594	1.764420	0.763161	NaN
min	4.300000	2.000000	1.000000	0.100000	NaN
25%	5.100000	2.800000	1.600000	0.300000	NaN
50%	5.800000	3.000000	4.350000	1.300000	NaN
75%	6.400000	3.300000	5.100000	1.800000	NaN
max	7.900000	4.400000	6.900000	2.500000	NaN

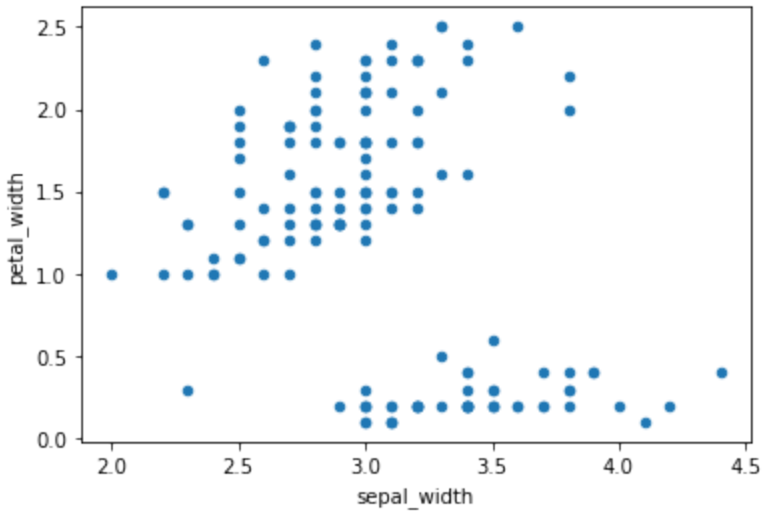
In [32]:
```
import matplotlib.pyplot as plt
from matplotlib import pyplot as plt
df.hist('sepal_length') #histogram
plt.show()
```



In [34]:
```
# Scatter plot
df.plot(kind='scatter', x='sepal_length', y='petal_length')
plt.show()
```



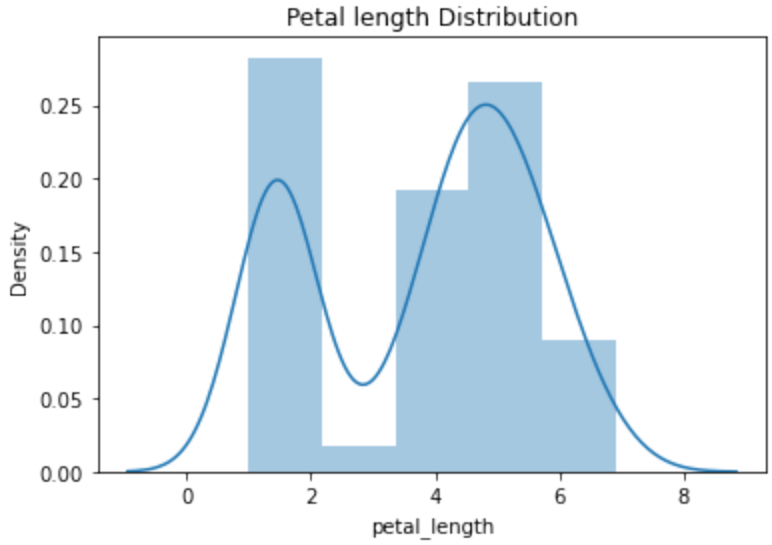
In [36]:
```
# Scatter plot for numerical columns
df.plot(kind='scatter', x='sepal_width', y='petal_width')
plt.show()
```



In [39]:
```
sns.distplot(df['petal_length']).set_title('Petal length Distribution')
```

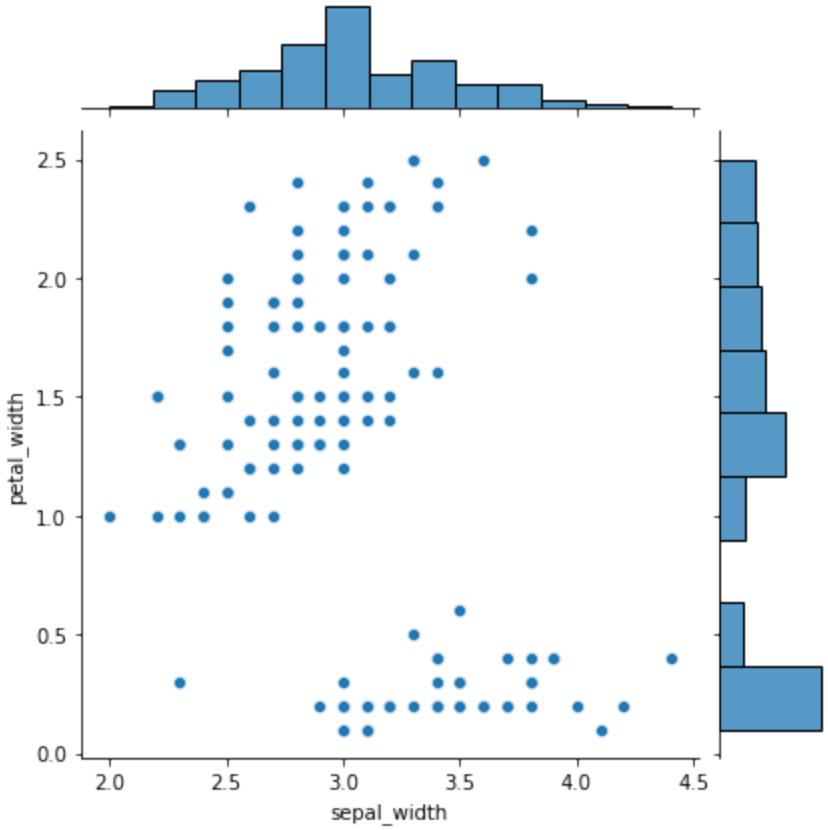
```
C:\Users\sanka\anaconda3\lib\site-packages\seaborn\distributions.py:2551: Fut
ureWarning: `distplot` is a deprecated function and will be removed in a futu
re version. Please adapt your code to use either `displot` (a figure-level fu
nction with similar flexibility) or `histplot` (an axes-level function for hi
stograms).
  warnings.warn(msg, FutureWarning)
```

Out[39]:
```
Text(0.5, 1.0, 'Petal length Distribution')
```



In [41]:
```
sns.jointplot(x = 'sepal_width', y = 'petal_width', data = df)
```

Out[41]:
```
<seaborn.axisgrid.JointGrid at 0x23d7a086160>
```



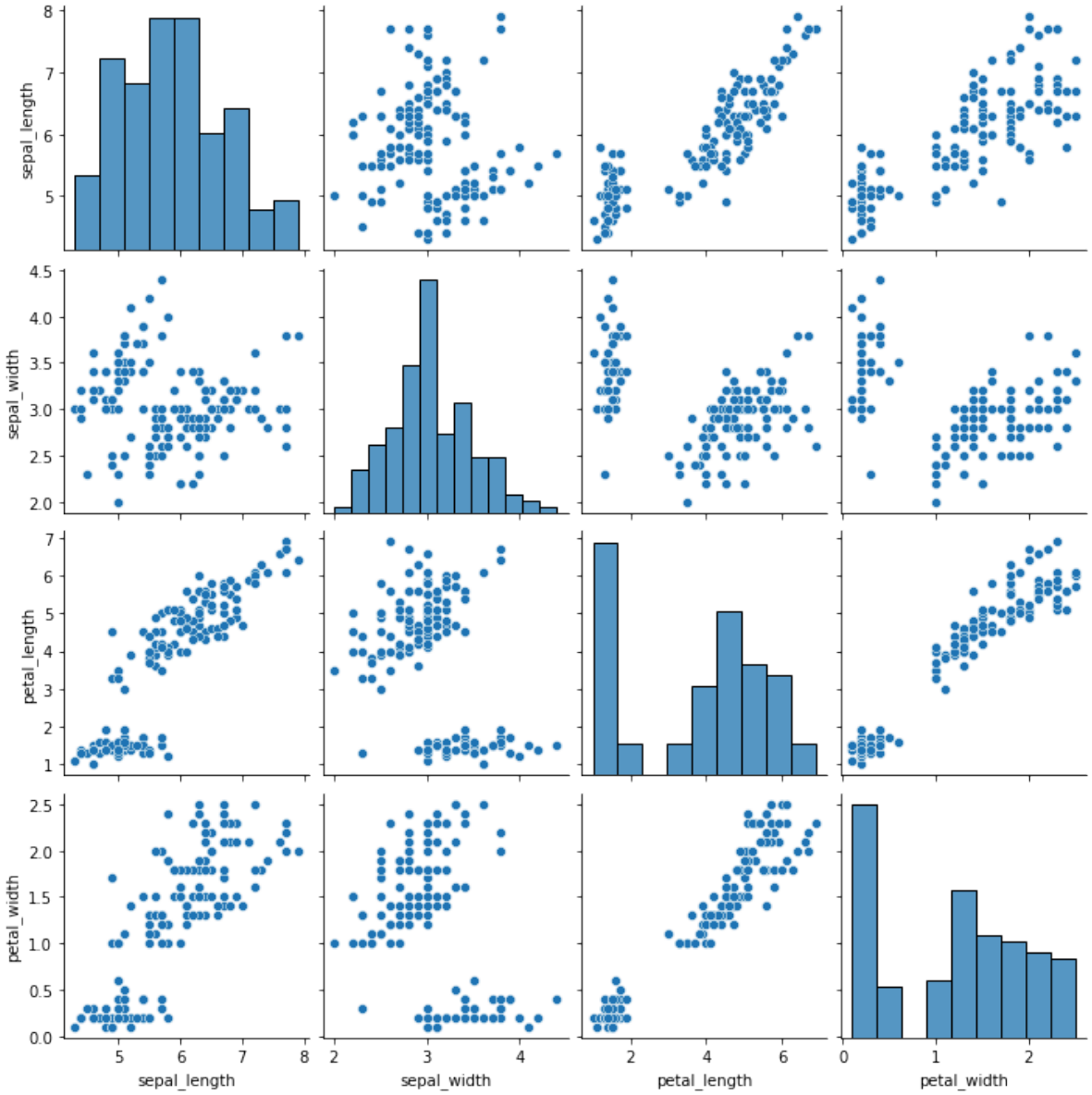
In [42]:
```
df.corr()
```

Out[42]:

	sepal_length	sepal_width	petal_length	petal_width
sepal_length	1.000000	-0.109369	0.871754	0.817954
sepal_width	-0.109369	1.000000	-0.420516	-0.356544
petal_length	0.871754	-0.420516	1.000000	0.962757
petal_width	0.817954	-0.356544	0.962757	1.000000

In [43]:
```
sns.pairplot(df)
```

Out[43]:
```
<seaborn.axisgrid.PairGrid at 0x23d79c399d0>
```



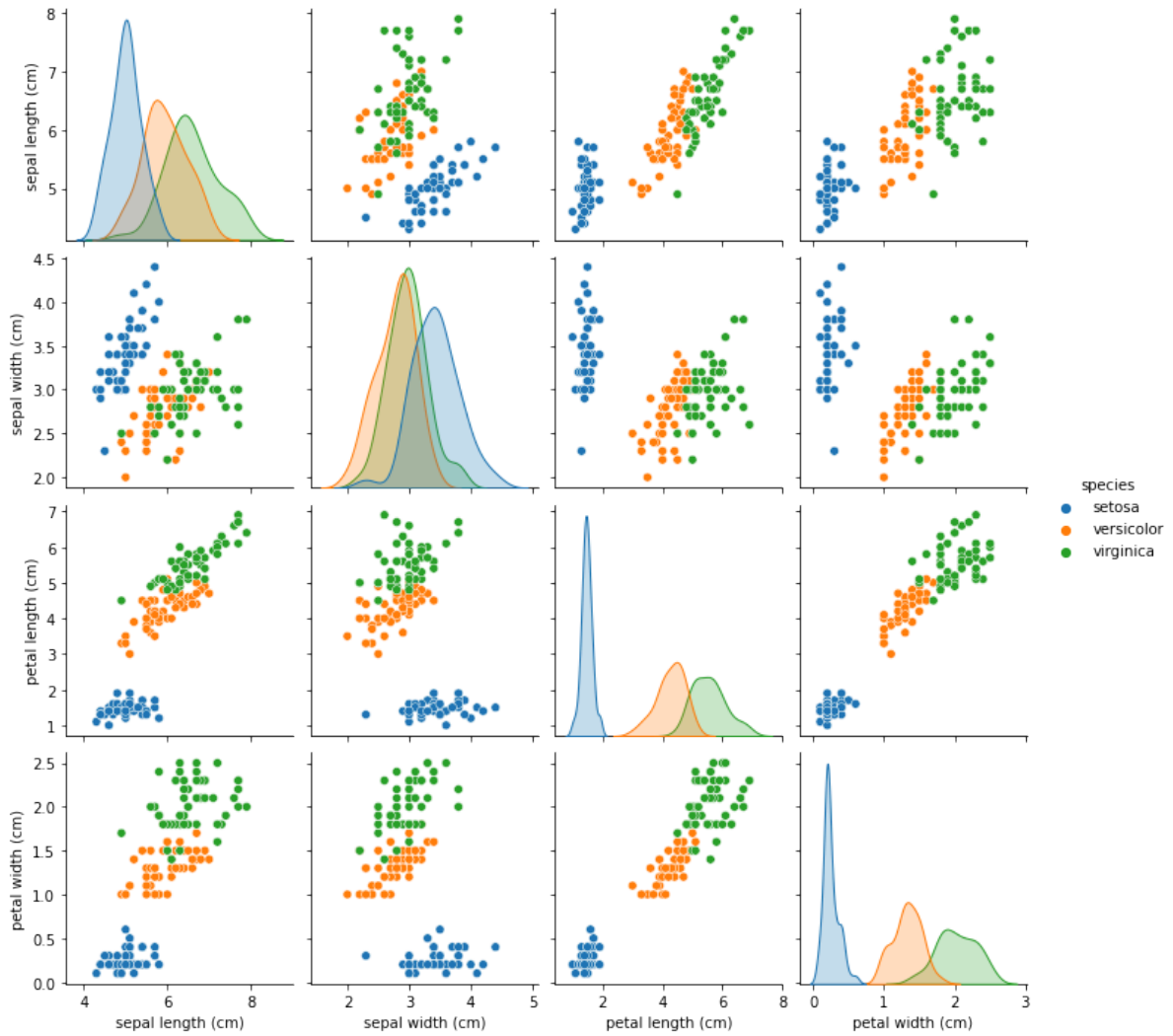
In [45]:
```
from sklearn.datasets import load_iris
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import StandardScaler
from sklearn.ensemble import RandomForestClassifier
from sklearn.metrics import classification_report, confusion_matrix, accuracy_

# Load the dataset
iris = load_iris()
df = pd.DataFrame(data=iris.data, columns=iris.feature_names)
df['species'] = iris.target
df['species'] = df['species'].map({0: 'setosa', 1: 'versicolor', 2: 'virginica

print(df.head())
```

```
   sepal length (cm)  sepal width (cm)  petal length (cm)  petal width (cm)
\
0                5.1               3.5                1.4               0.2
1                4.9               3.0                1.4               0.2
2                4.7               3.2                1.3               0.2
3                4.6               3.1                1.5               0.2
4                5.0               3.6                1.4               0.2

  species
0  setosa
1  setosa
2  setosa
3  setosa
4  setosa
```

In [46]:
```
print(df.describe())
```

```
       sepal length (cm)  sepal width (cm)  petal length (cm)  \
count         150.000000        150.000000         150.000000
mean            5.843333          3.057333           3.758000
std             0.828066          0.435866           1.765298
min             4.300000          2.000000           1.000000
25%             5.100000          2.800000           1.600000
50%             5.800000          3.000000           4.350000
75%             6.400000          3.300000           5.100000
max             7.900000          4.400000           6.900000

       petal width (cm)
count        150.000000
mean           1.199333
std            0.762238
min            0.100000
25%            0.300000
50%            1.300000
75%            1.800000
max            2.500000
```

In [47]:
```
sns.pairplot(df, hue='species')
plt.show()
```



In [48]:
```
# Feature and target separation
X = df.drop('species', axis=1)
y = df['species']
```

In [49]:
```
#Standardize the features
scaler = StandardScaler()
X_scaled = scaler.fit_transform(X)
```

In [50]:
```
X_train, X_test, y_train, y_test = train_test_split(X_scaled, y, test_size=0.2
```

In [51]:
```
# Initialize and train the RandomForestClassifier
model = RandomForestClassifier(random_state=42)
model.fit(X_train, y_train)
```

Out[51]:
```
RandomForestClassifier
RandomForestClassifier(random_state=42)
```

In [52]:
```
# Make predictions
y_pred = model.predict(X_test)
```

In [53]:
```
# Evaluation metrics
print("Confusion Matrix:")
print(confusion_matrix(y_test, y_pred))
```

```
Confusion Matrix:
[[10  0  0]
 [ 0  9  0]
 [ 0  0 11]]
```

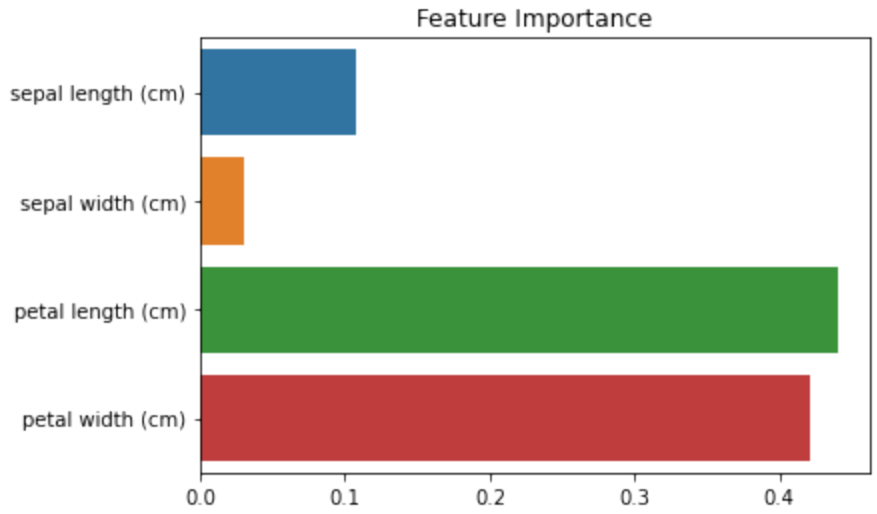
In [54]:
```
print("\nClassification Report:")
print(classification_report(y_test, y_pred))
```

```
Classification Report:
              precision    recall  f1-score   support

      setosa       1.00      1.00      1.00        10
  versicolor       1.00      1.00      1.00         9
   virginica       1.00      1.00      1.00        11

    accuracy                           1.00        30
   macro avg       1.00      1.00      1.00        30
weighted avg       1.00      1.00      1.00        30
```

In [55]:
```
print("\nAccuracy Score:")
print(accuracy_score(y_test, y_pred))
```

```
Accuracy Score:
1.0
```

In [57]:
```
# Visualizing feature importance
importances = model.feature_importances_
feature_names = iris.feature_names
sns.barplot(x=importances, y=feature_names)
plt.title('Feature Importance')
plt.show()
```



In [ ]:
```

```