```
import pandas as pd
from sklearn.datasets import load_iris
iris=load_iris()
dir(iris)
```

The line you provided performs the following actions:

It imports the pandas library and assigns it the alias pd.
pandas is a popular library in Python used for data manipulation and analysis.

It imports the load_iris function from the sklearn.datasets module.
sklearn (scikit-learn) is a widely used library for machine learning in Python.

It creates a variable named iris and assigns it the result of calling the load_iris function.
load_iris is a function provided that loads the Iris dataset, which is a popular dataset used for classificatio n tasks.
The dir() function returns a list of names in the specified object. In this case,
it will return a list of names (attributes and methods) available in the iris object, allowing you to explore and understand its structure and functionality.

```
df=pd.DataFrame(iris.data,columns=iris.feature_names)
```

The line you provided creates a Pandas DataFrame named df using the data from the Iris dataset.
pd.DataFrame() creates a new DataFrame object from the data passed as the first argument.
In this case, the data is iris.data, which represents the feature values of the Iris dataset.

The columns parameter specifies the column names for the DataFrame.
In this case, it is set to iris.feature_names, which represents the names of the features in the Iris dataset.

```
df['target']=iris.target
iris.feature_names
```

```
df['target'] = iris.target
```

This line adds a new column named 'target' to the DataFrame df. The values for this column are assigned from iris.target

```
iris.feature_names
```

This line returns a list of feature names from the Iris dataset.
These feature names represent the column names that can be used to label the columns in the DataFram e df.
Each feature name corresponds to a specific attribute of the iris flowers, such as 'sepal length', 'sepal widt h', 'petal length', and 'petal width'.

```
iris.target_names
df['species']=df.target.apply(lambda x:iris.target_names[x])
```

iris.target_names
represents the names of the classes or species of the iris flowers in the dataset.
In the case of the Iris dataset, there are three classes: 'setosa', 'versicolor', and 'virginica'.

df['species'] = df.target.apply(lambda x: iris.target_names[x])

This line adds a new column named 'species' to the DataFrame df.
It assigns values to this column using the apply() function along with a lambda function.

The apply() function in Pandas allows you to apply a function to each element of a DataFrame column. In this case,
the code creates a new column 'species' in the DataFrame df and populates it with the species names cor responding to the target values.

iris.target_names

```
from matplotlib import pyplot as plt
%matplotlib inline
df0=df[df.target==0]
df1=df[df.target==1]
df2=df[df.target==2]
plt.xlabel('petal length')
plt.ylabel('petal width')
plt.scatter(df0['petal length (cm)'],df0['petal width (cm)'],color='red')
plt.scatter(df1['petal length (cm)'],df1['petal width (cm)'],color='green')
plt.scatter(df2['petal length (cm)'],df2['petal width (cm)'],color='blue')
```

df0 = df[df.target == 0]
This line creates a new DataFrame df0 by filtering df based on a condition.
it selects only the rows in df where the 'target' column has a value of 0.

df1 = df[df.target == 1]
This line creates a new DataFrame df1 by filtering df based on a condition.
It selects only the rows in df where the 'target' column has a value of 1

df2 = df[df.target == 2]
This line creates a new DataFrame df2 by filtering df based on a condition.
It selects only the rows in df where the 'target' column has a value of 2.

from sklearn.model_selection import train_test_split
splits the training data and testing data

X=df.drop(['target','species'],axis='columns')
This line creates a new DataFrame X by dropping the specified columns from the original DataFrame df.
The drop() function is used to remove columns from a DataFrame. In this case, the columns being droppe d are 'target' and 'species'
axis='columns' indicates that the specified columns should be dropped along the column axis (i.e., horizo ntally).

By removing the 'target' and 'species' columns,
X will contain only the feature columns (i.e., the independent variables) of the dataset.

This is typically done to separate the features from the target variable before training a machine learning model.

```
y=df.target
X_train,X_test,y_train,y_test=train_test_split(X,y,test_size=0.2)
```

this line creates a new variable y and assigns it the values from the ’target’

```
  X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2)
```

This line uses the train_test_split function to split the feature and target data into training and testing subs ets.
It assigns the resulting subsets to the variables X_train, X_test, y_train, and y_test.

```
from sklearn.svm import SVC
model=SVC()
model.fit(X_train,y_train)
```

```
  from sklearn.svm import SVC
```

This line imports the SVC class from the sklearn.svm module. SVC stands for Support Vector Classifier, which is a popular implementation of Support Vector Machines (SVM) for classification tasks.

```
  model = SVC()
```

This line creates an instance of the SVC class and assigns it to the variable model.

```
  model.fit(X_train, y_train)
```

This line trains the SVM classifier (model) using the fit method.
The fit method takes the feature data X_train and the corresponding target data y_train as arguments.
It fits the SVM model to the training data, allowing it to learn the patterns and relationships between the fe atures and target variable.

During the training process, the SVM classifier optimizes its internal parameters to find the decision boun daries that best separate the
different classes in the training data.

Once the fit method is executed, the SVM classifier (model) is trained and ready to make predictions on n ew, unseen data.