Introduction to Python

Learn how read a file, shuffle data, filter data, plot data, split file into training and test sets

In [3]:
```
%matplotlib inline
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
```

In [4]:
```
# Modified IRIS Dataset - Introduced NaNs for some of the observations
# Dataset: https://archive.ics.uci.edu/ml/datasets/Iris/
```

In [23]:
```
df = pd.read_csv('IrisMissingData.csv')
```

In [6]:
```
df.head(10)
```

Out[6]:

	sepal_length	sepal_width	petal_length	petal_width	class
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
5	5.4	3.9	1.7	0.4	Iris-setosa
6	4.6	NaN	1.4	0.3	Iris-setosa
7	5.0	3.4	NaN	0.2	Iris-setosa
8	4.4	2.9	1.4	0.2	Iris-setosa
9	4.9	3.1	1.5	0.1	Iris-setosa

In [7]:
```
df.loc[0:2]
```

Out[7]:

	sepal_length	sepal_width	petal_length	petal_width	class
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

In [8]:
```
df.loc[7]
```

Out[8]:
```
sepal_length            5.0
sepal_width             3.4
petal_length            NaN
petal_width             0.2
class           Iris-setosa
Name: 7, dtype: object
```

In [9]:
```
df['sepal_length']
```

Out[9]:
```
0      5.1
1      4.9
2      4.7
3      4.6
4      5.0
      ...
145    NaN
146    6.3
147    6.5
148    6.2
149    5.9
Name: sepal_length, Length: 150, dtype: float64
```

In [10]:
```
df[['sepal_length','sepal_width']]
```

Out[10]:

	sepal_length	sepal_width
0	5.1	3.5
1	4.9	3.0
2	4.7	3.2
3	4.6	3.1
4	5.0	3.6
...	...	...
145	NaN	3.0
146	6.3	2.5
147	6.5	3.0
148	6.2	3.4
149	5.9	3.0

150 rows × 2 columns

In [11]:
```
df.head(10)
```

Out[11]:

	sepal_length	sepal_width	petal_length	petal_width	class
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
5	5.4	3.9	1.7	0.4	Iris-setosa
6	4.6	NaN	1.4	0.3	Iris-setosa
7	5.0	3.4	NaN	0.2	Iris-setosa
8	4.4	2.9	1.4	0.2	Iris-setosa
9	4.9	3.1	1.5	0.1	Iris-setosa

In [12]:
```
df.tail()
```

Out[12]:

	sepal_length	sepal_width	petal_length	petal_width	class
145	NaN	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 [14]:
```
df = df.fillna(df.mean()) ## That will be a problem - we have NaNs and the class co
```

```
---------------------------------------------------------------------------
ValueError                                Traceback (most recent call last)
File ~/anaconda3/envs/python3/lib/python3.10/site-packages/pandas/core/nanops.py:1
680, in _ensure_numeric(x)
   1679 try:
-> 1680     x = x.astype(np.complex128)
   1681 except (TypeError, ValueError):

ValueError: complex() arg is a malformed string

During handling of the above exception, another exception occurred:

ValueError                                Traceback (most recent call last)
File ~/anaconda3/envs/python3/lib/python3.10/site-packages/pandas/core/nanops.py:1
683, in _ensure_numeric(x)
   1682 try:
-> 1683     x = x.astype(np.float64)
   1684 except ValueError as err:
   1685     # GH#29941 we get here with object arrays containing strs

ValueError: could not convert string to float: 'Iris-setosaIris-setosaIris-setosaI
ris-setosaIris-setosaIris-setosaIris-setosaIris-setosaIris-setosaIris-setosaIris-s
etosaIris-setosaIris-setosaIris-setosaIris-setosaIris-setosaIris-setosaIris-setosa
Iris-setosaIris-setosaIris-setosaIris-setosaIris-setosaIris-setosaIris-setosaIris-
setosaIris-setosaIris-setosaIris-setosaIris-setosaIris-setosaIris-setosaIris-setos
aIris-setosaIris-setosaIris-setosaIris-setosaIris-setosaIris-setosaIris-setosaIris
-setosaIris-setosaIris-setosaIris-setosaIris-setosaIris-setosaIris-setosaIris-seto
saIris-setosaIris-setosaIris-versicolorIris-versicolorIris-versicolorIris-versicol
orIris-versicolorIris-versicolorIris-versicolorIris-versicolorIris-versicolorIris-
versicolorIris-versicolorIris-versicolorIris-versicolorIris-versicolorIris-versico
lorIris-versicolorIris-versicolorIris-versicolorIris-versicolorIris-versicolorIris
-versicolorIris-versicolorIris-versicolorIris-versicolorIris-versicolorIris-versic
olorIris-versicolorIris-versicolorIris-versicolorIris-versicolorIris-versicolorIri
s-versicolorIris-versicolorIris-versicolorIris-versicolorIris-versicolorIris-versi
colorIris-versicolorIris-versicolorIris-versicolorIris-versicolorIris-versicolorIr
is-versicolorIris-versicolorIris-versicolorIris-versicolorIris-versicolorIris-vers
icolorIris-versicolorIris-versicolorIris-virginicaIris-virginicaIris-virginicaIris
-virginicaIris-virginicaIris-virginicaIris-virginicaIris-virginicaIris-virginicaIr
is-virginicaIris-virginicaIris-virginicaIris-virginicaIris-virginicaIris-virginica
Iris-virginicaIris-virginicaIris-virginicaIris-virginicaIris-virginicaIris-virgini
caIris-virginicaIris-virginicaIris-virginicaIris-virginicaIris-virginicaIris-virgi
nicaIris-virginicaIris-virginicaIris-virginicaIris-virginicaIris-virginicaIris-vir
ginicaIris-virginicaIris-virginicaIris-virginicaIris-virginicaIris-virginicaIris-v
irginicaIris-virginicaIris-virginicaIris-virginicaIris-virginicaIris-virginicaIris
-virginicaIris-virginicaIris-virginicaIris-virginicaIris-virginicaIris-virginica'

The above exception was the direct cause of the following exception:

TypeError                                 Traceback (most recent call last)
Cell In[14], line 1
----> 1 df = df.fillna(df.mean()) ## That will be a problem - we have NaNs and the
class colunn is a string

File ~/anaconda3/envs/python3/lib/python3.10/site-packages/pandas/core/generic.py:
11556, in NDFrame._add_numeric_operations.<locals>.mean(self, axis, skipna, numeri
c_only, **kwargs)
```

```
  11539 @doc(
  11540     _num_doc,
  11541     desc="Return the mean of the values over the requested axis.",
   (...)
  11554     **kwargs,
  11555 ):
> 11556     return NDFrame.mean(self, axis, skipna, numeric_only, **kwargs)

File ~/anaconda3/envs/python3/lib/python3.10/site-packages/pandas/core/generic.py:
11201, in NDFrame.mean(self, axis, skipna, numeric_only, **kwargs)
  11194 def mean(
  11195     self,
  11196     axis: Axis | None = 0,
   (...)
  11199     **kwargs,
  11200 ) -> Series | float:
> 11201     return self._stat_function(
  11202         "mean", nanops.nanmean, axis, skipna, numeric_only, **kwargs
  11203     )

File ~/anaconda3/envs/python3/lib/python3.10/site-packages/pandas/core/generic.py:
11158, in NDFrame._stat_function(self, name, func, axis, skipna, numeric_only, **k
wargs)
  11154     nv.validate_stat_func((), kwargs, fname=name)
  11156 validate_bool_kwarg(skipna, "skipna", none_allowed=False)
> 11158 return self._reduce(
  11159     func, name=name, axis=axis, skipna=skipna, numeric_only=numeric_only
  11160 )

File ~/anaconda3/envs/python3/lib/python3.10/site-packages/pandas/core/frame.py:10
524, in DataFrame._reduce(self, op, name, axis, skipna, numeric_only, filter_type,
**kwds)
  10520     df = df.T
  10522 # After possibly _get_data and transposing, we are now in the
  10523 #  simple case where we can use BlockManager.reduce
> 10524 res = df._mgr.reduce(blk_func)
  10525 out = df._constructor(res).iloc[0]
  10526 if out_dtype is not None:

File ~/anaconda3/envs/python3/lib/python3.10/site-packages/pandas/core/internals/m
anagers.py:1534, in BlockManager.reduce(self, func)
   1532 res_blocks: list[Block] = []
   1533 for blk in self.blocks:
-> 1534     nbs = blk.reduce(func)
   1535     res_blocks.extend(nbs)
   1537 index = Index([None])  # placeholder

File ~/anaconda3/envs/python3/lib/python3.10/site-packages/pandas/core/internals/b
locks.py:339, in Block.reduce(self, func)
    333 @final
    334 def reduce(self, func) -> list[Block]:
    335     # We will apply the function and reshape the result into a single-row
    336     #  Block with the same mgr_locs; squeezing will be done at a higher le
vel
    337     assert self.ndim == 2
--> 339     result = func(self.values)
```

```
    341     if self.values.ndim == 1:
    342         # TODO(EA2D): special case not needed with 2D EAs
    343         res_values = np.array([[result]])

File ~/anaconda3/envs/python3/lib/python3.10/site-packages/pandas/core/frame.py:10
487, in DataFrame._reduce.<locals>.blk_func(values, axis)
  10485     return values._reduce(name, skipna=skipna, **kwds)
  10486 else:
> 10487     return op(values, axis=axis, skipna=skipna, **kwds)

File ~/anaconda3/envs/python3/lib/python3.10/site-packages/pandas/core/nanops.py:9
6, in disallow.__call__.<locals>._f(*args, **kwargs)
     94 try:
     95     with np.errstate(invalid="ignore"):
---> 96         return f(*args, **kwargs)
     97 except ValueError as e:
     98     # we want to transform an object array
     99     # ValueError message to the more typical TypeError
    100     # e.g. this is normally a disallowed function on
    101     # object arrays that contain strings
    102     if is_object_dtype(args[0]):

File ~/anaconda3/envs/python3/lib/python3.10/site-packages/pandas/core/nanops.py:1
58, in bottleneck_switch.__call__.<locals>.f(values, axis, skipna, **kwds)
    156         result = alt(values, axis=axis, skipna=skipna, **kwds)
    157 else:
--> 158     result = alt(values, axis=axis, skipna=skipna, **kwds)
    160 return result

File ~/anaconda3/envs/python3/lib/python3.10/site-packages/pandas/core/nanops.py:4
21, in _datetimelike_compat.<locals>.new_func(values, axis, skipna, mask, **kwarg
s)
    418 if datetimelike and mask is None:
    419     mask = isna(values)
--> 421 result = func(values, axis=axis, skipna=skipna, mask=mask, **kwargs)
    423 if datetimelike:
    424     result = _wrap_results(result, orig_values.dtype, fill_value=iNaT)

File ~/anaconda3/envs/python3/lib/python3.10/site-packages/pandas/core/nanops.py:7
27, in nanmean(values, axis, skipna, mask)
    724     dtype_count = dtype
    726 count = _get_counts(values.shape, mask, axis, dtype=dtype_count)
--> 727 the_sum = _ensure_numeric(values.sum(axis, dtype=dtype_sum))
    729 if axis is not None and getattr(the_sum, "ndim", False):
    730     count = cast(np.ndarray, count)

File ~/anaconda3/envs/python3/lib/python3.10/site-packages/pandas/core/nanops.py:1
686, in _ensure_numeric(x)
   1683         x = x.astype(np.float64)
   1684     except ValueError as err:
   1685         # GH#29941 we get here with object arrays containing strs
-> 1686         raise TypeError(f"Could not convert {x} to numeric") from err
   1687 else:
   1688     if not np.any(np.imag(x)):

TypeError: Could not convert ['Iris-setosaIris-setosaIris-setosaIris-setosaIris-se
```

```
tosaIris-setosaIris-setosaIris-setosaIris-setosaIris-setosaIris-setosaIris-setosaI
ris-setosaIris-setosaIris-setosaIris-setosaIris-setosaIris-setosaIris-setosaIris-s
etosaIris-setosaIris-setosaIris-setosaIris-setosaIris-setosaIris-setosaIris-setosa
Iris-setosaIris-setosaIris-setosaIris-setosaIris-setosaIris-setosaIris-setosaIris-
setosaIris-setosaIris-setosaIris-setosaIris-setosaIris-setosaIris-setosaIris-setos
aIris-setosaIris-setosaIris-setosaIris-setosaIris-setosaIris-setosaIris-setosaIris
-setosaIris-versicolorIris-versicolorIris-versicolorIris-versicolorIris-versicolor
Iris-versicolorIris-versicolorIris-versicolorIris-versicolorIris-versicolorIris-ve
rsicolorIris-versicolorIris-versicolorIris-versicolorIris-versicolorIris-versicolo
rIris-versicolorIris-versicolorIris-versicolorIris-versicolorIris-versicolorIris-v
ersicolorIris-versicolorIris-versicolorIris-versicolorIris-versicolorIris-versicol
orIris-versicolorIris-versicolorIris-versicolorIris-versicolorIris-versicolorIris-
versicolorIris-versicolorIris-versicolorIris-versicolorIris-versicolorIris-versico
lorIris-versicolorIris-versicolorIris-versicolorIris-versicolorIris-versicolorIris
-versicolorIris-versicolorIris-versicolorIris-versicolorIris-versicolorIris-versic
olorIris-versicolorIris-virginicaIris-virginicaIris-virginicaIris-virginicaIris-vi
rginicaIris-virginicaIris-virginicaIris-virginicaIris-virginicaIris-virginicaIris-
virginicaIris-virginicaIris-virginicaIris-virginicaIris-virginicaIris-virginicaIri
s-virginicaIris-virginicaIris-virginicaIris-virginicaIris-virginicaIris-virginicaI
ris-virginicaIris-virginicaIris-virginicaIris-virginicaIris-virginicaIris-virginic
aIris-virginicaIris-virginicaIris-virginicaIris-virginicaIris-virginicaIris-virgin
icaIris-virginicaIris-virginicaIris-virginicaIris-virginicaIris-virginicaIris-virg
inicaIris-virginicaIris-virginicaIris-virginicaIris-virginicaIris-virginicaIris-vi
rginicaIris-virginicaIris-virginicaIris-virginicaIris-virginica'] to numeric
```

In [45]:
```
# Learned something in a later lesson using the Shift+Tab stuff
df = df.fillna(df.mean(numeric_only=True))
```

In [46]:
```
df.head(10)
```

Out[46]:

	sepal_length	sepal_width	petal_length	petal_width	class
82	5.8	2.7	3.9	1.2	Iris-versicolor
134	6.1	2.6	5.6	1.4	Iris-virginica
114	5.8	2.8	5.1	2.4	Iris-virginica
42	4.4	3.2	1.3	0.2	Iris-setosa
109	7.2	3.6	6.1	2.5	Iris-virginica
57	4.9	2.4	3.3	1.0	Iris-versicolor
1	4.9	3.0	1.4	0.2	Iris-setosa
70	5.9	3.2	4.8	1.8	Iris-versicolor
25	5.0	3.0	1.6	0.2	Iris-setosa
84	5.4	3.0	4.5	1.5	Iris-versicolor

In [47]:
```
df.tail()
```

Out[47]:

	sepal_length	sepal_width	petal_length	petal_width	class
8	4.4	2.9	1.4	0.2	Iris-setosa
73	6.1	2.8	4.7	1.2	Iris-versicolor
144	6.7	3.3	5.7	2.5	Iris-virginica
118	7.7	2.6	6.9	2.3	Iris-virginica
99	5.7	2.8	4.1	1.3	Iris-versicolor

In [48]:
```
df.shape
```

Out[48]:
```
(150, 5)
```

In [49]:
```
# Print first 5 index values and last 5 index values
index_list = list(df.index)
print(index_list[:5])
print(index_list[-5:])
```

```
[82, 134, 114, 42, 109]
[8, 73, 144, 118, 99]
```

In [50]:
```
# Randomize a list
np.random.seed(5)
np.random.shuffle(index_list)
```

In [51]:
```
print('Shuffled list')
print(index_list[:5])
print(index_list[-5:])
```

```
Shuffled list
[59, 121, 83, 17, 108]
[25, 87, 112, 2, 120]
```

In [52]:
```
# Dataframe is now shuffled (well, after we run this command)
df = df.iloc[index_list]
```

In [53]:
```
df.head(10)
```

Out[53]:

	sepal_length	sepal_width	petal_length	petal_width	class
149	5.9	3.0	5.1	1.8	Iris-virginica
51	6.4	3.2	4.5	1.5	Iris-versicolor
45	4.8	3.0	1.4	0.3	Iris-setosa
130	7.4	2.8	6.1	1.9	Iris-virginica
137	6.4	3.1	5.5	1.8	Iris-virginica
24	4.8	3.4	1.9	0.2	Iris-setosa
121	5.6	2.8	4.9	2.0	Iris-virginica
86	6.7	3.1	4.7	1.5	Iris-versicolor
118	7.7	2.6	6.9	2.3	Iris-virginica
50	7.0	3.2	4.7	1.4	Iris-versicolor

In [54]:
```
df.tail()
```

Out[54]:

	sepal_length	sepal_width	petal_length	petal_width	class
148	6.2	3.4	5.40000	2.3	Iris-virginica
47	4.6	3.2	1.40000	0.2	Iris-setosa
18	5.7	3.8	1.70000	0.3	Iris-setosa
114	5.8	2.8	5.10000	2.4	Iris-virginica
140	6.7	3.1	3.74863	2.4	Iris-virginica

In [55]:
```
# Split data into training and test sets
# training = 70%
# test = 30%
size = df.shape[0]
train = round(size *.7)
test = size - train
```

In [56]:
```
size, train, test
```

Out[56]:
```
(150, 105, 45)
```

In [57]:
```
#DWB# df[:train] takes the first val(train) values (105) and outputs them as a CSV
df[:train].to_csv('iris_data_train.csv',index=True,index_label='Row',
columns=['sepal_length','sepal_width','petal_length','petal_width','class'])
```

In [58]:
```
#DWB# df[train:] takes all values after the first val(train), the last 45, in our c
df[train:].to_csv('iris_data_test.csv',index=True,index_label='Row',
          columns=['sepal_length','sepal_width','petal_length','petal_width'])
```

In [59]:
```
df['class'].value_counts()
```

Out[59]:
```
class
Iris-virginica     50
Iris-versicolor    50
Iris-setosa        50
Name: count, dtype: int64
```

In [60]:
```
df.describe()
```

Out[60]:

	sepal_length	sepal_width	petal_length	petal_width
count	150.000000	150.000000	150.000000	150.000000
mean	5.836486	3.056463	3.748630	1.205405
std	0.824959	0.429307	1.741643	0.757759
min	4.300000	2.000000	1.000000	0.100000
25%	5.100000	2.800000	1.600000	0.300000
50%	5.800000	3.000000	4.250000	1.300000
75%	6.400000	3.300000	5.100000	1.800000
max	7.900000	4.400000	6.900000	2.500000

In [61]:
```
df.head()
```

Out[61]:

	sepal_length	sepal_width	petal_length	petal_width	class
149	5.9	3.0	5.1	1.8	Iris-virginica
51	6.4	3.2	4.5	1.5	Iris-versicolor
45	4.8	3.0	1.4	0.3	Iris-setosa
130	7.4	2.8	6.1	1.9	Iris-virginica
137	6.4	3.1	5.5	1.8	Iris-virginica

In [62]:
```
# Find all rows that match the condition.  Returns a Series that contains index val
setosa = df['class'] == 'Iris-setosa'
```

In [63]:
```
setosa.head()
```

Out[63]:
```
149    False
51     False
45      True
130    False
137    False
Name: class, dtype: bool
```

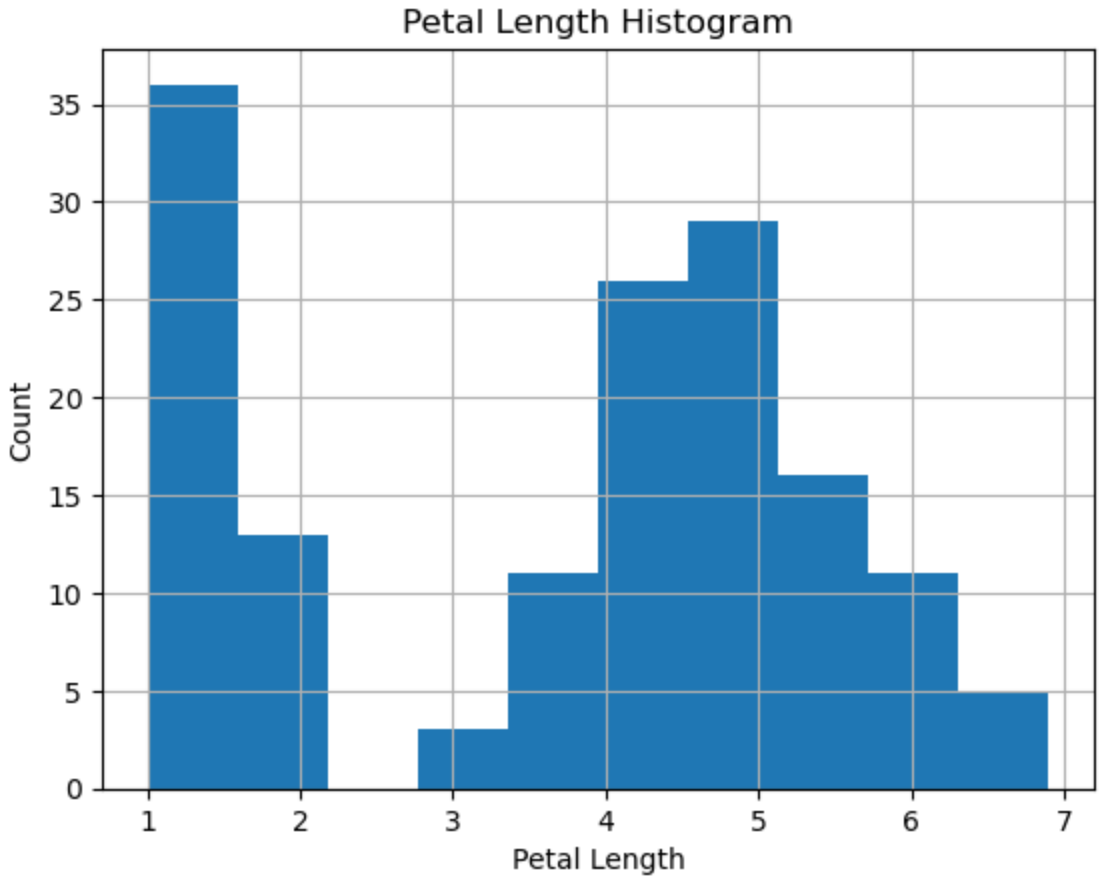
In [64]:
```
# Pull only setosa's
df[setosa].head()
```

Out[64]:

	sepal_length	sepal_width	petal_length	petal_width	class
45	4.8	3.0	1.4	0.3	Iris-setosa
24	4.8	3.4	1.9	0.2	Iris-setosa
11	4.8	3.4	1.6	0.2	Iris-setosa
42	4.4	3.2	1.3	0.2	Iris-setosa
43	5.0	3.5	1.6	0.6	Iris-setosa

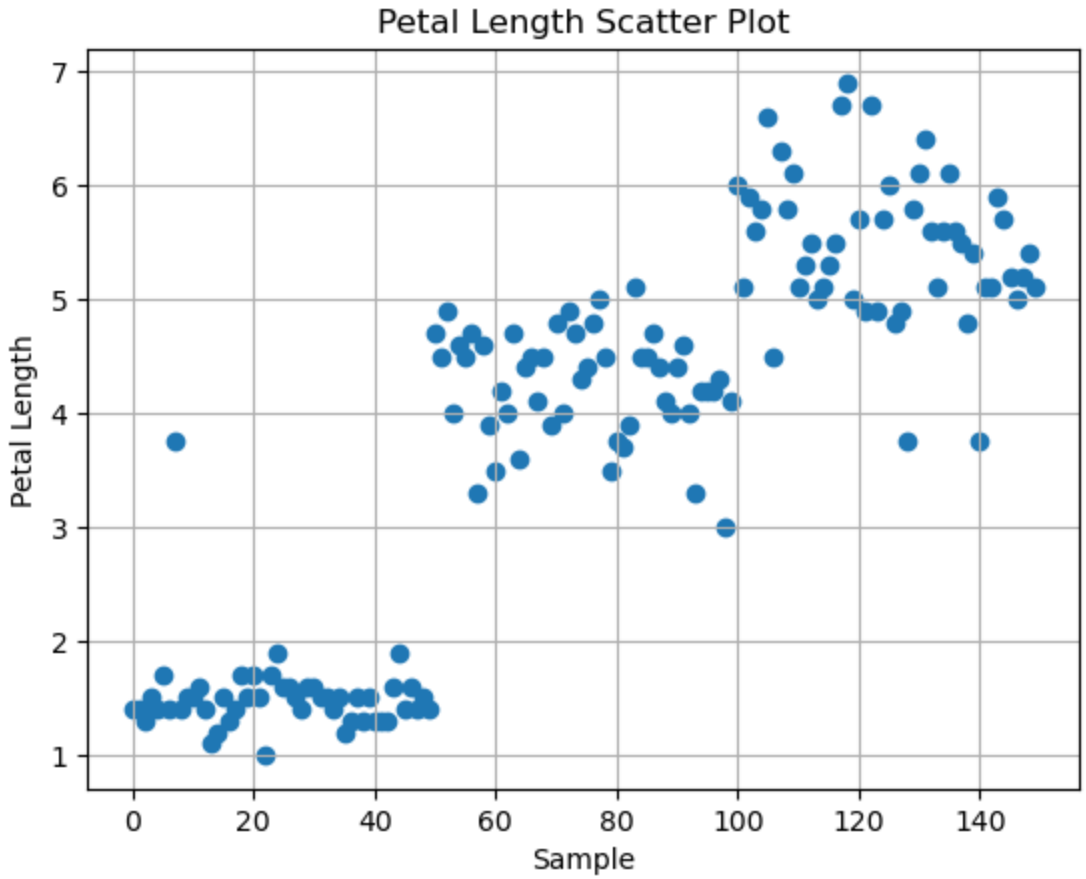
In [65]:
```
plt.hist(df.petal_length)
plt.grid(True)
plt.xlabel('Petal Length')
plt.ylabel('Count')
plt.title('Petal Length Histogram')
```

Out[65]:
```
Text(0.5, 1.0, 'Petal Length Histogram')
```



In [66]:
```
plt.scatter(df.index, df.petal_length)
plt.grid(True)
plt.xlabel('Sample')
plt.ylabel('Petal Length')
plt.title('Petal Length Scatter Plot')
```

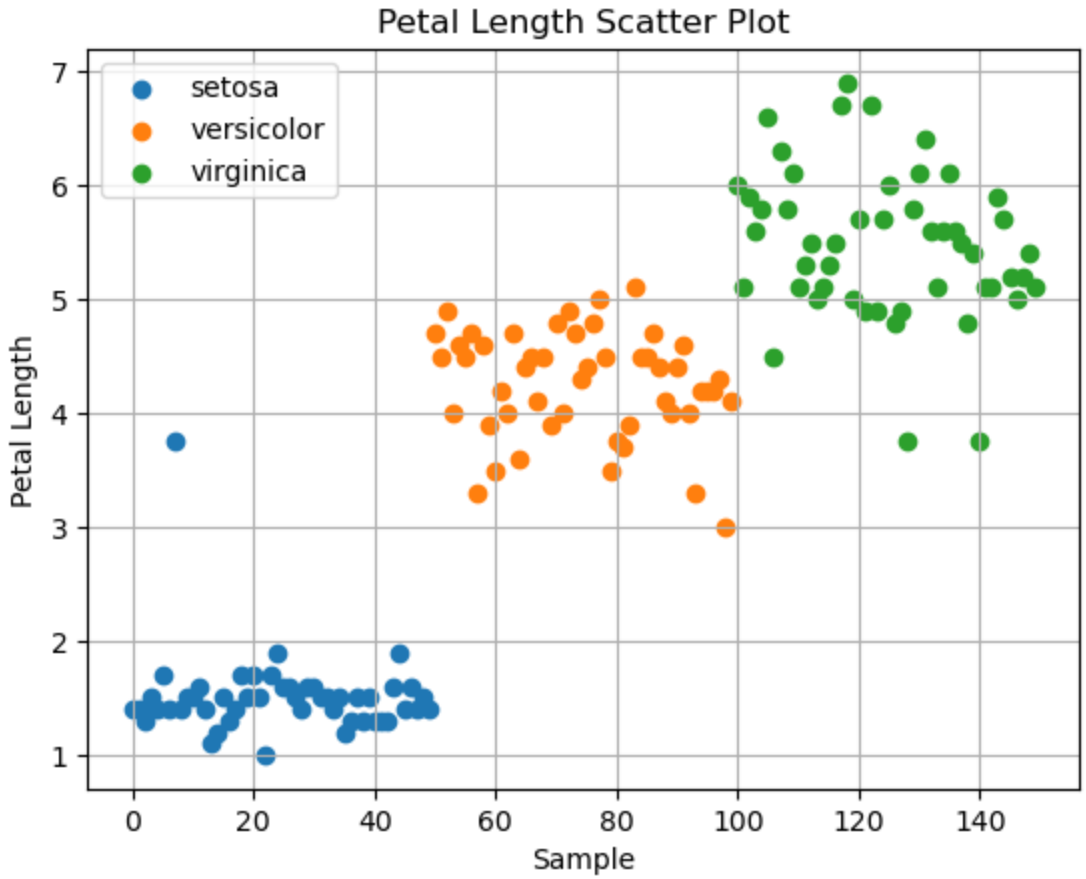
Out[66]:
```
Text(0.5, 1.0, 'Petal Length Scatter Plot')
```



In [67]:
```
versicolor = df['class'] == "Iris-versicolor"
virginica = df['class'] == "Iris-virginica"
```

In [68]:
```
plt.scatter(df[setosa].index, df[setosa].petal_length, label ='setosa')
plt.scatter(df[versicolor].index, df[versicolor].petal_length, label = 'versicolor'
plt.scatter(df[virginica].index, df[virginica].petal_length, label = 'virginica')
plt.grid(True)
plt.xlabel('Sample')
plt.ylabel('Petal Length')
plt.title('Petal Length Scatter Plot')
plt.legend()
```

Out[68]:
```
<matplotlib.legend.Legend at 0x7f4b21a43400>
```


Petal Length Scatter Plot
setosa
versicolor
virginica
Petal Length
Sample
1
2
3
4
5
6
7
0
20
40
60
80
100
120
140

In [ ]: