

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

150 rows × 2 columns

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

Out[11]:

	sepal_length	sepal_width	petal_length	petal_width	class
<b>0</b>	5.1	3.5	1.4	0.2	Iris-setosa
<b>1</b>	4.9	3.0	1.4	0.2	Iris-setosa
<b>2</b>	4.7	3.2	1.3	0.2	Iris-setosa
<b>3</b>	4.6	3.1	1.5	0.2	Iris-setosa
<b>4</b>	5.0	3.6	1.4	0.2	Iris-setosa
<b>5</b>	5.4	3.9	1.7	0.4	Iris-setosa
<b>6</b>	4.6	NaN	1.4	0.3	Iris-setosa
<b>7</b>	5.0	3.4	NaN	0.2	Iris-setosa
<b>8</b>	4.4	2.9	1.4	0.2	Iris-setosa
<b>9</b>	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
<b>145</b>	NaN	3.0	5.2	2.3	Iris-virginica
<b>146</b>	6.3	2.5	5.0	1.9	Iris-virginica
<b>147</b>	6.5	3.0	5.2	2.0	Iris-virginica
<b>148</b>	6.2	3.4	5.4	2.3	Iris-virginica
<b>149</b>	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*



```

11539 @doc(
11540     _num_doc,
11541     desc="Return the mean of the values over the requested axis.",
11542     (...)
11543     **kwargs,
11544 ):
> 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,
11197     (...)
11198     **kwargs,
11199 ) -> 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, **kwargs)
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 level
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:10487, 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:96, 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:158, 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:421, in \_datetimelike\_compat.<locals>.new\_func(values, axis, skipna, mask, \*\*kwargs)

```

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:727, 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:1686, 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



Out[16]:

	sepal_length	sepal_width	petal_length	petal_width	class
<b>145</b>	NaN	3.0	5.2	2.3	Iris-virginica
<b>146</b>	6.3	2.5	5.0	1.9	Iris-virginica
<b>147</b>	6.5	3.0	5.2	2.0	Iris-virginica
<b>148</b>	6.2	3.4	5.4	2.3	Iris-virginica
<b>149</b>	5.9	3.0	5.1	1.8	Iris-virginica

In [17]: `df.shape`

Out[17]: (150, 5)

In [24]: *# Print first 5 index values and last 5 index values*  
`index_list = list(df.index)`  
`print(index_list[:5])`  
`print(index_list[-5:])`

[0, 1, 2, 3, 4]  
 [145, 146, 147, 148, 149]

In [25]: *# Randomize a List*  
`np.random.seed(5)`  
`np.random.shuffle(index_list)`

In [26]: `print('Shuffled list')`  
`print(index_list[:5])`  
`print(index_list[-5:])`

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

In [28]: *# Dataframe is now shuffled (well, after we run this command)*  
`df = df.iloc[index_list]`

In [29]: `df.head(10)`



Out[29]:

	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 [30]: `df.tail()`

Out[30]:

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

In [32]: `size, train, test`

Out[32]: (150, 105, 45)

```
In [33]: #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 [34]: #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 [35]: `df['class'].value_counts()`

```
Out[35]: class
Iris-versicolor    50
Iris-virginica     50
Iris-setosa        50
Name: count, dtype: int64
```

```
In [36]: df.describe()
```

```
Out[36]:
```

	sepal_length	sepal_width	petal_length	petal_width
<b>count</b>	148.000000	147.000000	146.000000	148.000000
<b>mean</b>	5.836486	3.056463	3.748630	1.205405
<b>std</b>	0.830552	0.433695	1.765502	0.762897
<b>min</b>	4.300000	2.000000	1.000000	0.100000
<b>25%</b>	5.100000	2.800000	1.600000	0.300000
<b>50%</b>	5.800000	3.000000	4.350000	1.300000
<b>75%</b>	6.400000	3.300000	5.100000	1.800000
<b>max</b>	7.900000	4.400000	6.900000	2.500000

```
In [37]: df.head()
```

```
Out[37]:
```

	sepal_length	sepal_width	petal_length	petal_width	class
<b>82</b>	5.8	2.7	3.9	1.2	Iris-versicolor
<b>134</b>	6.1	2.6	5.6	1.4	Iris-virginica
<b>114</b>	5.8	2.8	5.1	2.4	Iris-virginica
<b>42</b>	4.4	3.2	1.3	0.2	Iris-setosa
<b>109</b>	7.2	3.6	6.1	2.5	Iris-virginica

```
In [38]: # Find all rows that match the condition. Returns a Series that contains index val
setosa = df['class'] == 'Iris-setosa'
```

```
In [39]: setosa.head()
```

```
Out[39]: 82    False
134    False
114    False
42     True
109    False
Name: class, dtype: bool
```

```
In [40]: # Pull only setosa's
df[setosa].head()
```

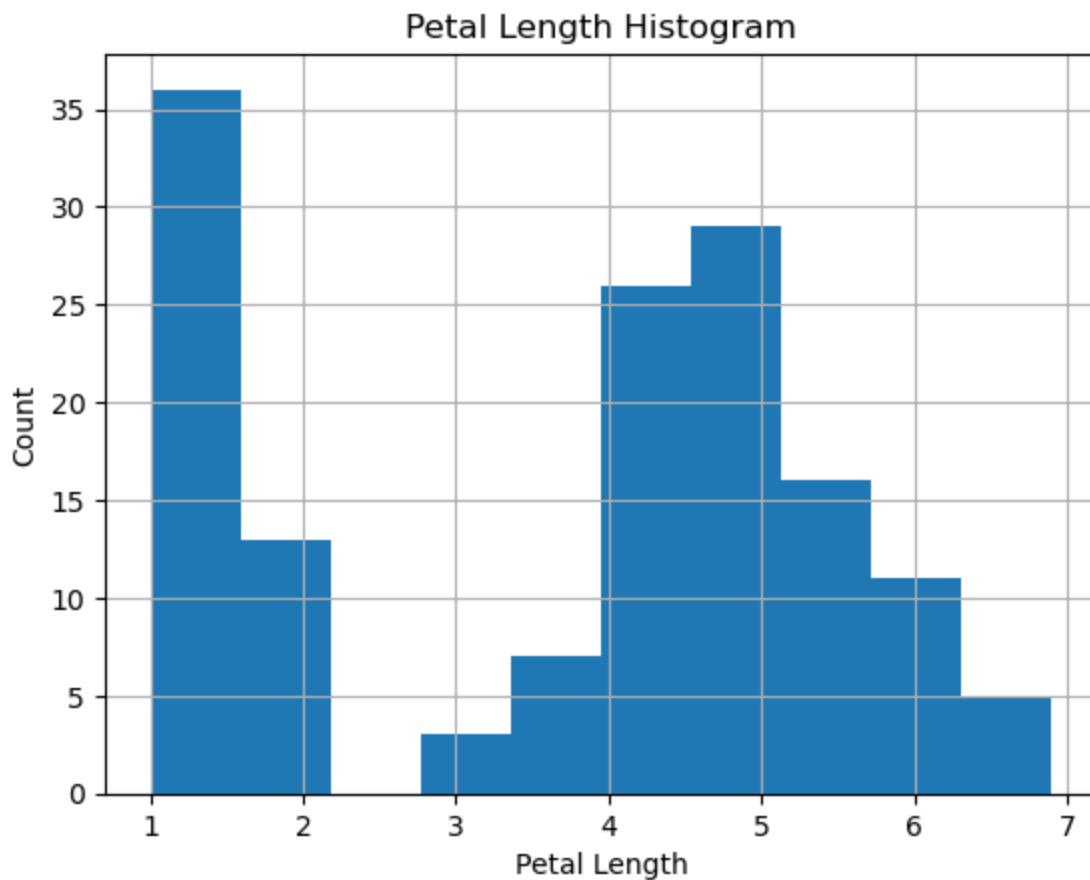
Out[40]:

	sepal_length	sepal_width	petal_length	petal_width	class
42	4.4	3.2	1.3	0.2	Iris-setosa
1	4.9	3.0	1.4	0.2	Iris-setosa
25	5.0	3.0	1.6	0.2	Iris-setosa
26	5.0	3.4	1.6	0.4	Iris-setosa
23	5.1	3.3	1.7	0.5	Iris-setosa

```
In [41]: plt.hist(df.petal_length)
plt.grid(True)
plt.xlabel('Petal Length')
plt.ylabel('Count')
plt.title('Petal Length Histogram')
```

```
/home/ec2-user/anaconda3/envs/python3/lib/python3.10/site-packages/numpy/lib/histograms.py:837: RuntimeWarning: invalid value encountered in greater_equal
  keep = (tmp_a >= first_edge)
/home/ec2-user/anaconda3/envs/python3/lib/python3.10/site-packages/numpy/lib/histograms.py:838: RuntimeWarning: invalid value encountered in less_equal
  keep &= (tmp_a <= last_edge)
```

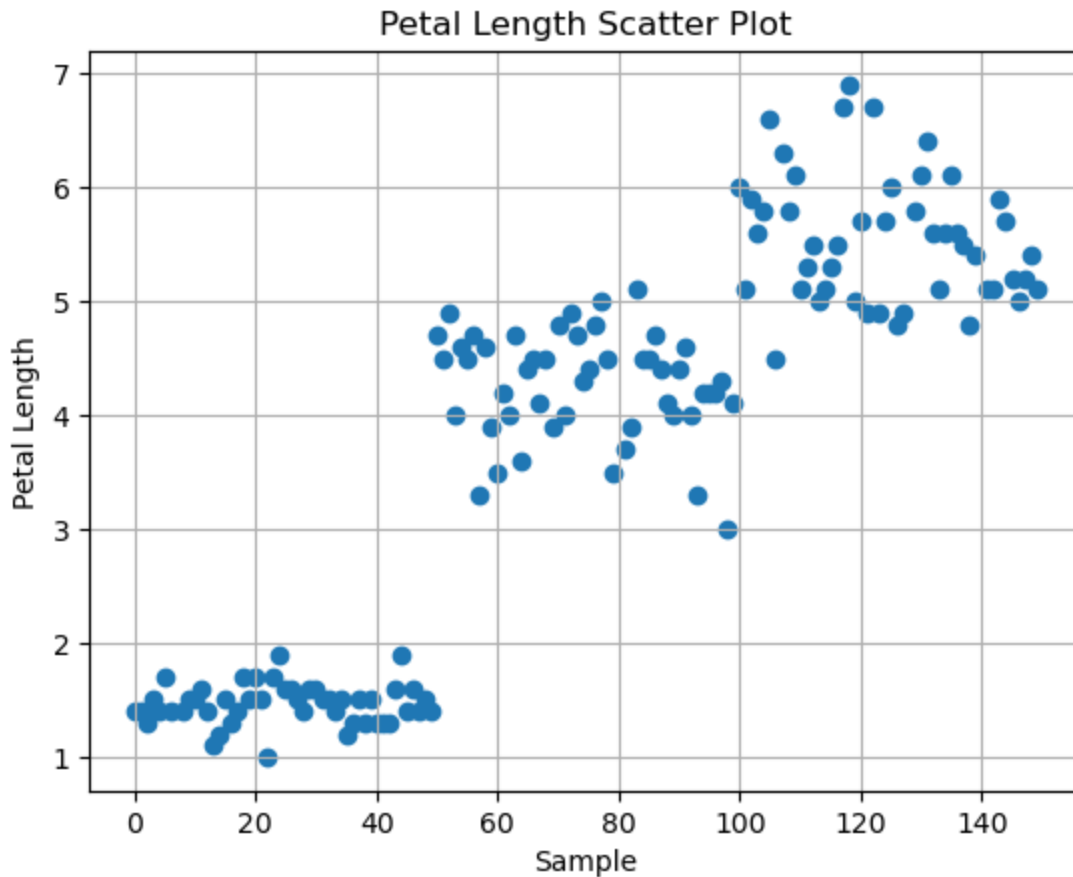
Out[41]: Text(0.5, 1.0, 'Petal Length Histogram')



```
In [42]: plt.scatter(df.index, df.petal_length)
plt.grid(True)
plt.xlabel('Sample')
```

```
plt.ylabel('Petal Length')  
plt.title('Petal Length Scatter Plot')
```

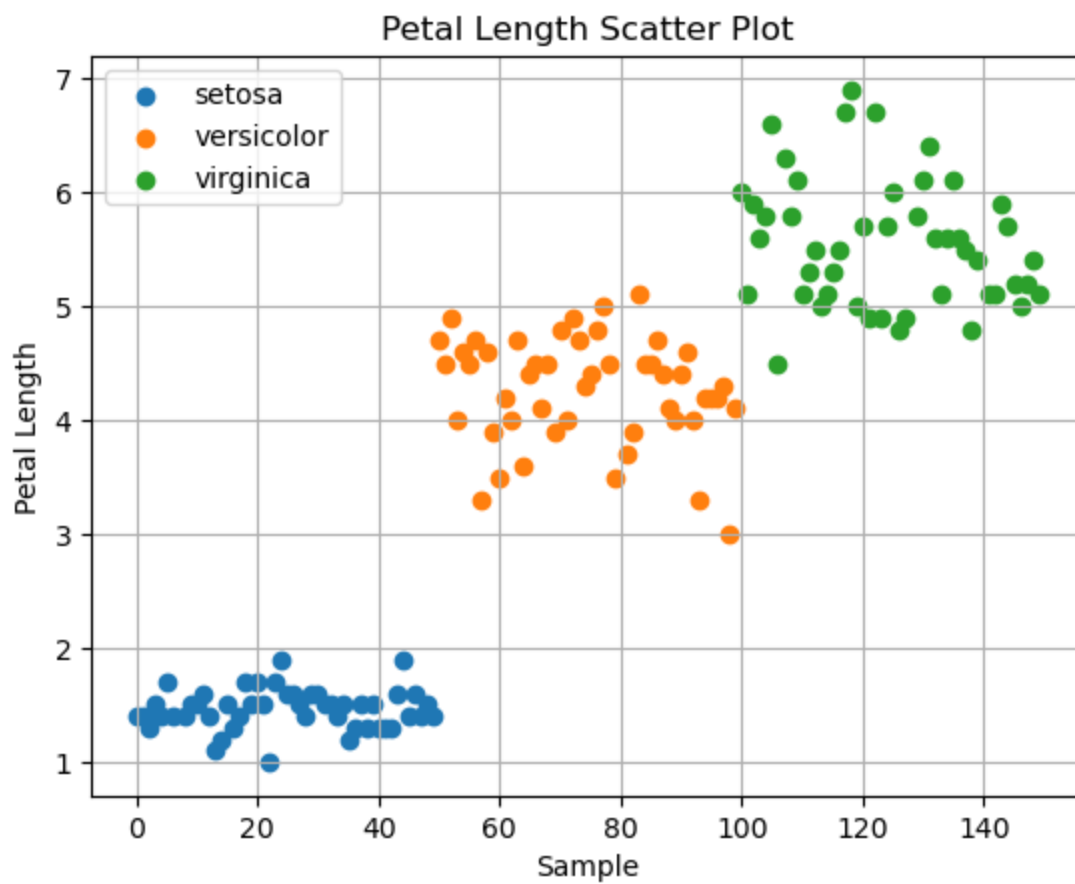
Out[42]: Text(0.5, 1.0, 'Petal Length Scatter Plot')



```
In [43]: versicolor = df['class'] == "Iris-versicolor"  
         virginica = df['class'] == "Iris-virginica"
```

```
In [44]: 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[44]: <matplotlib.legend.Legend at 0x7f4b21a87430>



In [ ]: