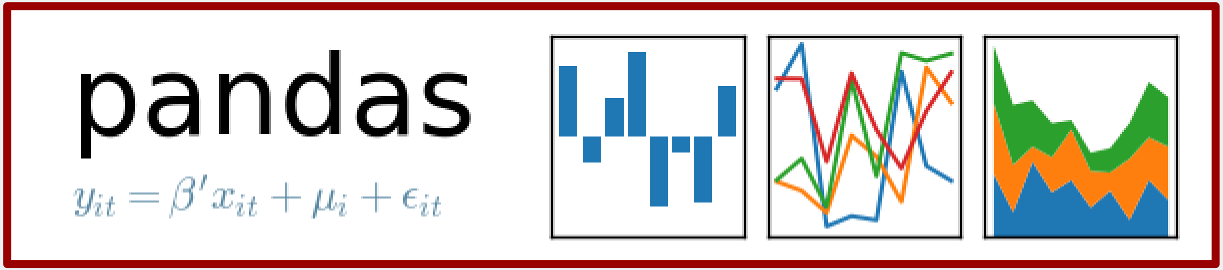
Data Analysis using Pandas

Foundations of AI Academy

Pandas

A Python library for processing data structures and data analysis



```
import pandas as pd
```

The Series Data Type

Similar to NumPy, Pandas uses its own similar data structure to help process data

```
import pandas as pd
s = pd.Series([1, 2, 3, 4, 5],
    index=['a', 'b', 'c', 'd', 'e'])
```

The Series Data Type

Similar to NumPy, Pandas uses its own similar data structure to help process data

Data is created via list, dictionary, or numpy array

```
import pandas as pd
s = pd.Series([1, 2, 3, 4, 5],
    index=['a', 'b', 'c', 'd', 'e'])
```

The Series Data Type

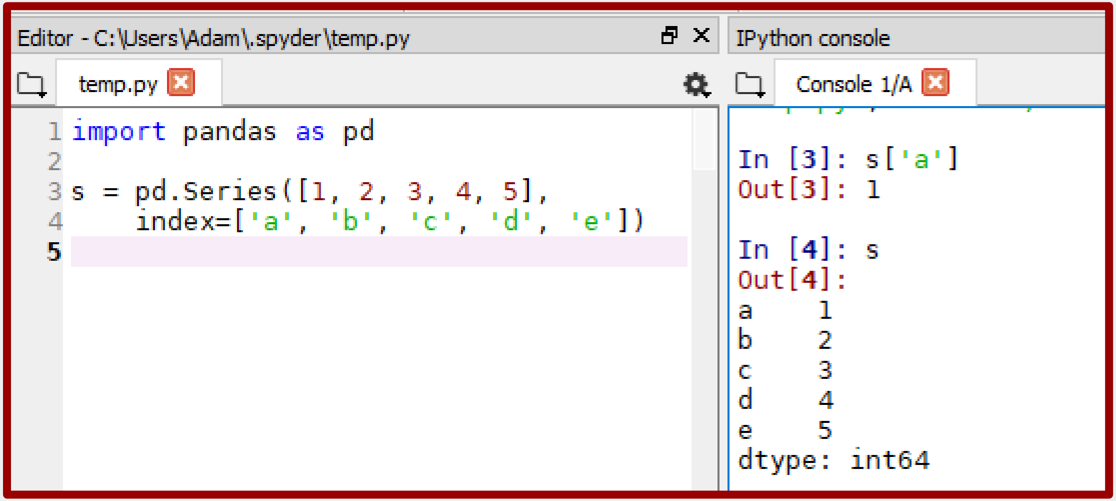
Similar to NumPy, Pandas uses its own similar data structure to help process data

```
import pandas as pd
s = pd.Series([1, 2, 3, 4, 5],
    index=['a', 'b', 'c', 'd', 'e'])
```

The index serves as the "header", similar to a dictionary

Interacting with REPL

Another variation to working with Python is to use the script to load the data and then use the Console to explore the data



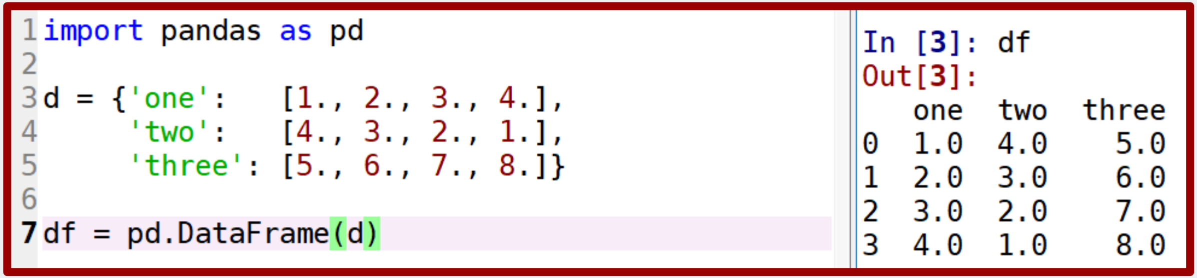
```python
import pandas as pd

s = pd.Series([1, 2, 3, 4, 5],
    index=['a', 'b', 'c', 'd', 'e'])

```

```
In [3]: s['a']
Out[3]: 1

In [4]: s
Out[4]:
a    1
b    2
c    3
d    4
e    5
dtype: int64
```

The DataFrame Data Type

DataFrame is a 2-dimensional labeled data structure with columns of potentially different types

- Similar to the CSV files we’ve been working with



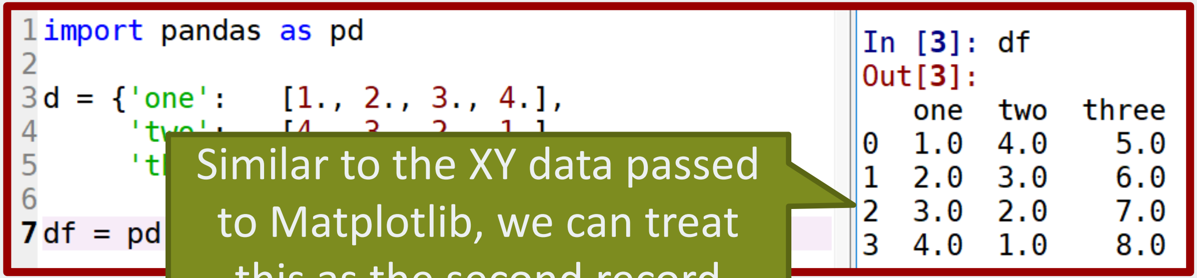
```python
import pandas as pd

d = {'one':   [1., 2., 3., 4.],
     'two':   [4., 3., 2., 1.],
     'three': [5., 6., 7., 8.]}

df = pd.DataFrame(d)
```

```
In [3]: df
Out[3]:
   one  two  three
0  1.0  4.0    5.0
1  2.0  3.0    6.0
2  3.0  2.0    7.0
3  4.0  1.0    8.0
```

The DataFrame Data Type

DataFrame is a 2-dimensional labeled data structure with columns of potentially different types

- Similar to the CSV files we’ve been working with



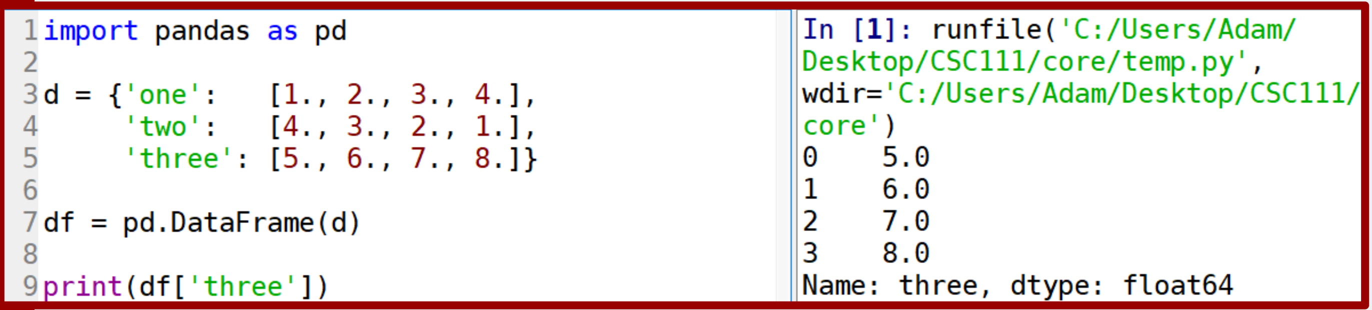
```
import pandas as pd

d = {'one':   [1., 2., 3., 4.],
     'two':   [4., 3., 2., 1.],
     'th

df = pd
```

```
In [3]: df
Out[3]:
   one  two  three
0  1.0  4.0    5.0
1  2.0  3.0    6.0
2  3.0  2.0    7.0
3  4.0  1.0    8.0
```

Similar to the XY data passed to Matplotlib, we can treat this as the second record

The DataFrame Data Type

Similar to the Series data type, we can use the labels / headers to isolate a single column of data

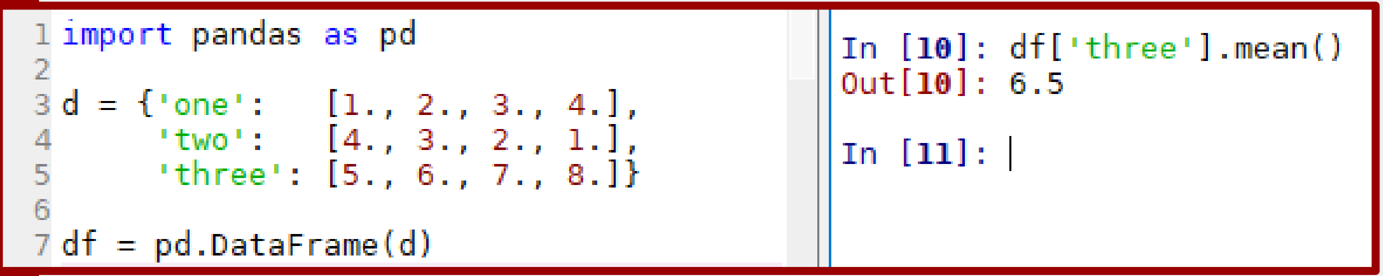


```python
import pandas as pd

d = {'one':   [1., 2., 3., 4.],
     'two':   [4., 3., 2., 1.],
     'three': [5., 6., 7., 8.]}

df = pd.DataFrame(d)

print(df['three'])
```

```
In [1]: runfile('C:/Users/Adam/
Desktop/CSC111/core/temp.py',
wdir='C:/Users/Adam/Desktop/CSC111/
core')
0    5.0
1    6.0
2    7.0
3    8.0
Name: three, dtype: float64
```

The DataFrame Data Type

This can be useful because then we can do exploratory analysis on descriptive statistics



```python
import pandas as pd

d = {'one':   [1., 2., 3., 4.],
     'two':   [4., 3., 2., 1.],
     'three': [5., 6., 7., 8.]}

df = pd.DataFrame(d)
```

```
In [10]: df['three'].mean()
Out[10]: 6.5

In [11]: 
```

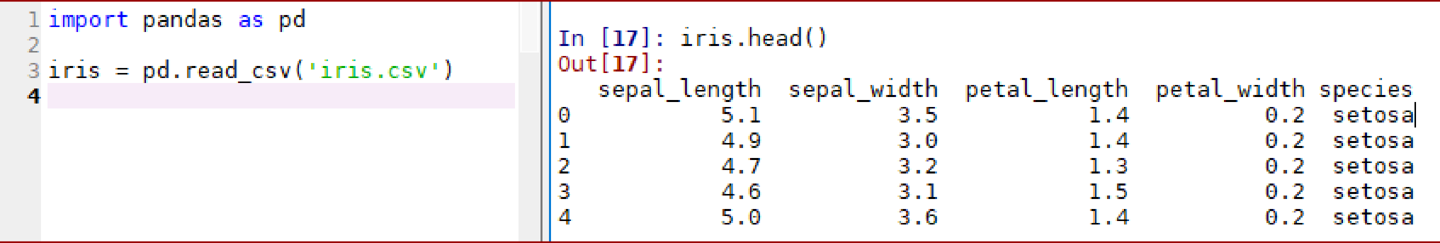
Transforming a CSV to a Data Frame

Pandas has a command similar to NumPy for reading and transforming a CSV file

```
# Pandas
df = pd.read_csv('iris.csv')

# NumPy
array = np.loadtxt('iris.csv',delimiter=',')
```

NumPy vs. Pandas

NumPy will only process CSV files with numeric values



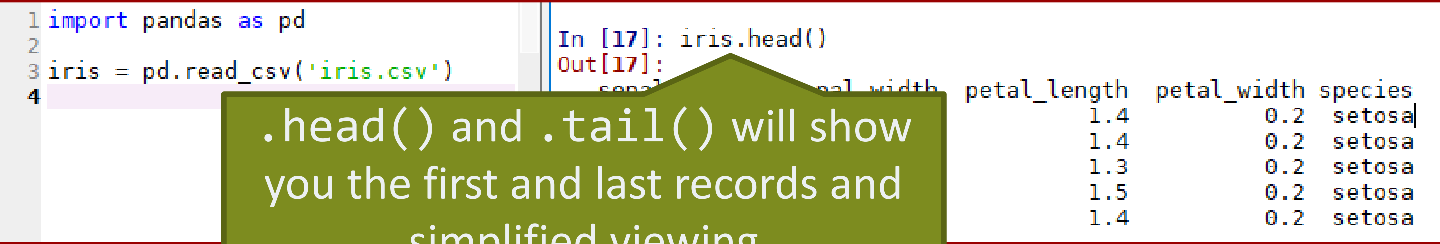
```
import pandas as pd

iris = pd.read_csv('iris.csv')

```

```
In [17]: iris.head()
Out[17]:
   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
```

Pandas will process CSV files with numeric values and characters

NumPy vs. Pandas

NumPy will only process CSV files with numeric values



```
import pandas as pd

iris = pd.read_csv('iris.csv')

```

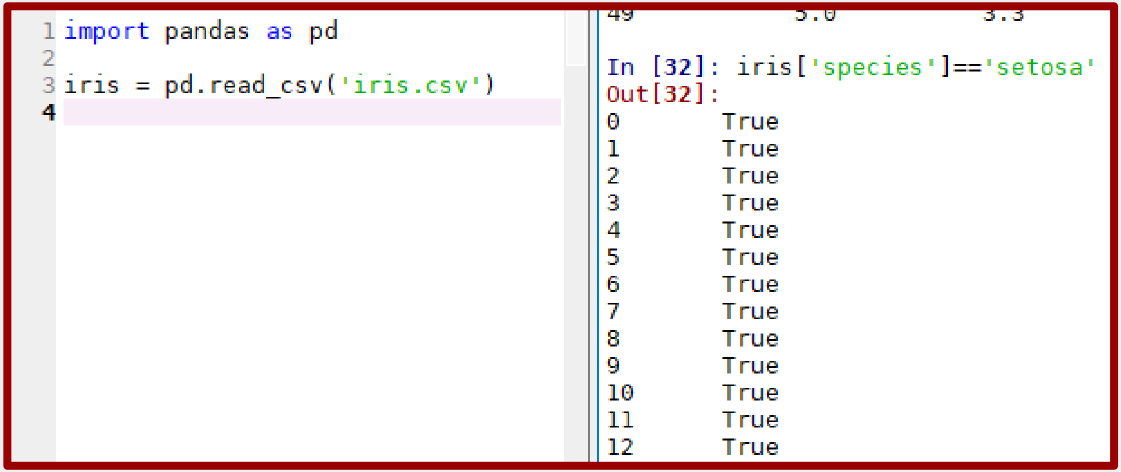
```
In [17]: iris.head()
Out[17]:
   sepal                    pal_width  petal_length  petal_width species
                                                1.4          0.2  setosa
                                                1.4          0.2  setosa
                                                1.3          0.2  setosa
                                                1.5          0.2  setosa
                                                1.4          0.2  setosa
```

.head() and .tail() will show you the first and last records and simplified viewing

Pandas will process CSV files with numeric values and characters

Evaluating DataFrame Rows

We can begin to filter our data incase we wish to do a specific analysis by applying conditional statements across a specific header

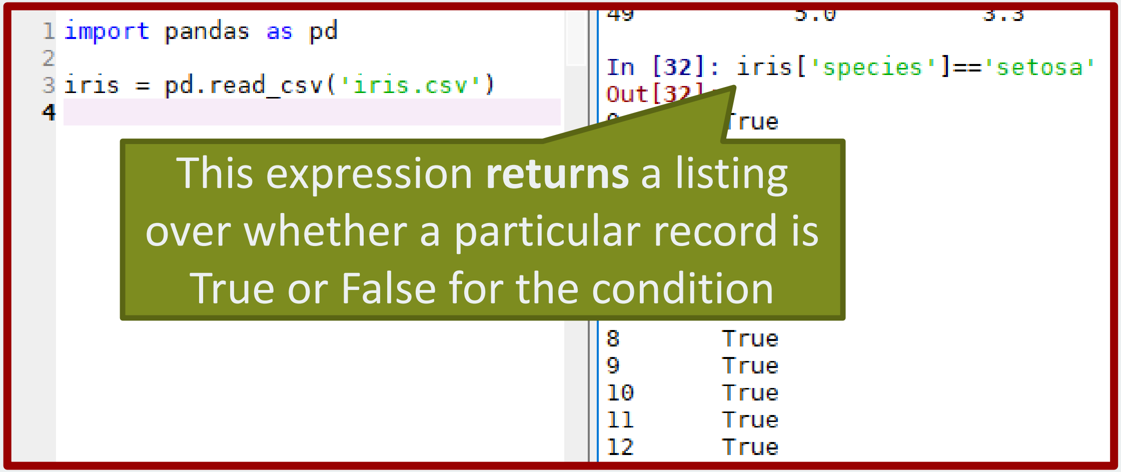


```
import pandas as pd

iris = pd.read_csv('iris.csv')

```

```
49              5.0             3.3

In [32]: iris['species']=='setosa'
Out[32]:
0        True
1        True
2        True
3        True
4        True
5        True
6        True
7        True
8        True
9        True
10       True
11       True
12       True
```

Evaluating DataFrame Rows

We can begin to filter our data incase we wish to do a specific analysis by applying conditional statements across a specific header



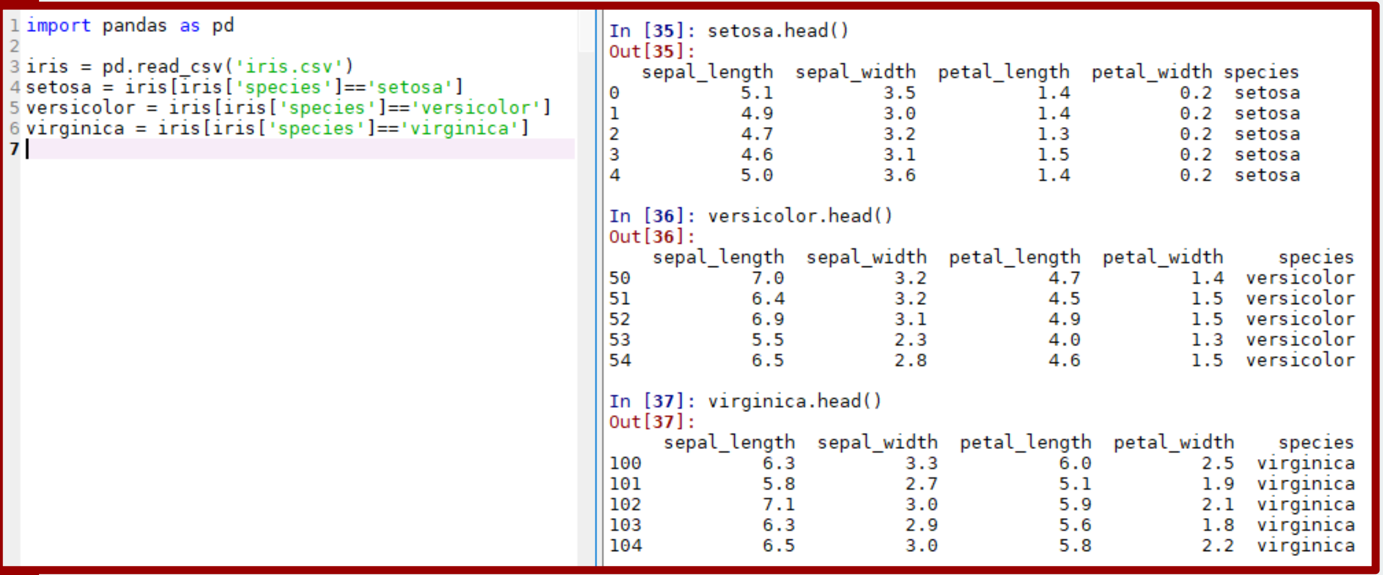
```
import pandas as pd

iris = pd.read_csv('iris.csv')

```

```
49           5.0          3.3

In [32]: iris['species']=='setosa'
Out[32]:
0         True
8         True
9         True
10        True
11        True
12        True
```

This expression **returns** a listing over whether a particular record is True or False for the condition

Filtering DataFrame Rows

With this we can now filter records by referencing them inside of square brackets

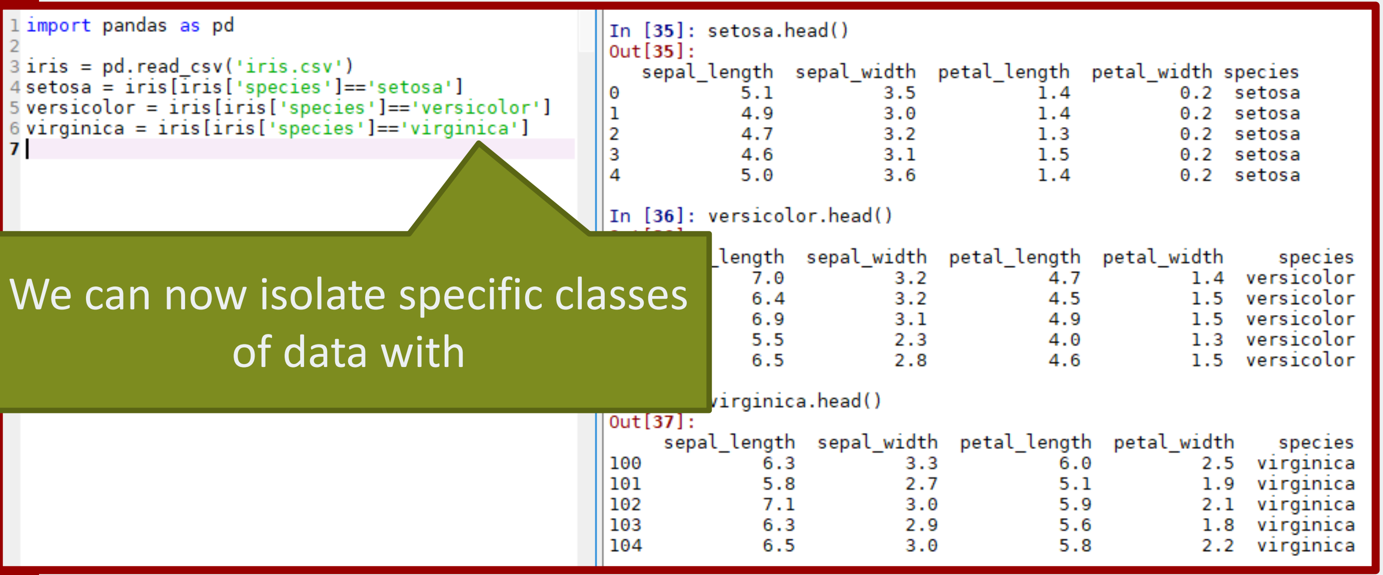


```python
import pandas as pd

iris = pd.read_csv('iris.csv')
setosa = iris[iris['species']=='setosa']
versicolor = iris[iris['species']=='versicolor']
virginica = iris[iris['species']=='virginica']

```

```
In [35]: setosa.head()
Out[35]:
   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

In [36]: versicolor.head()
Out[36]:
    sepal_length  sepal_width  petal_length  petal_width     species
50           7.0          3.2           4.7          1.4  versicolor
51           6.4          3.2           4.5          1.5  versicolor
52           6.9          3.1           4.9          1.5  versicolor
53           5.5          2.3           4.0          1.3  versicolor
54           6.5          2.8           4.6          1.5  versicolor

In [37]: virginica.head()
Out[37]:
     sepal_length  sepal_width  petal_length  petal_width    species
100           6.3          3.3           6.0          2.5  virginica
101           5.8          2.7           5.1          1.9  virginica
102           7.1          3.0           5.9          2.1  virginica
103           6.3          2.9           5.6          1.8  virginica
104           6.5          3.0           5.8          2.2  virginica
```

Filtering DataFrame Rows

With this we can now filter records by referencing them inside of square brackets



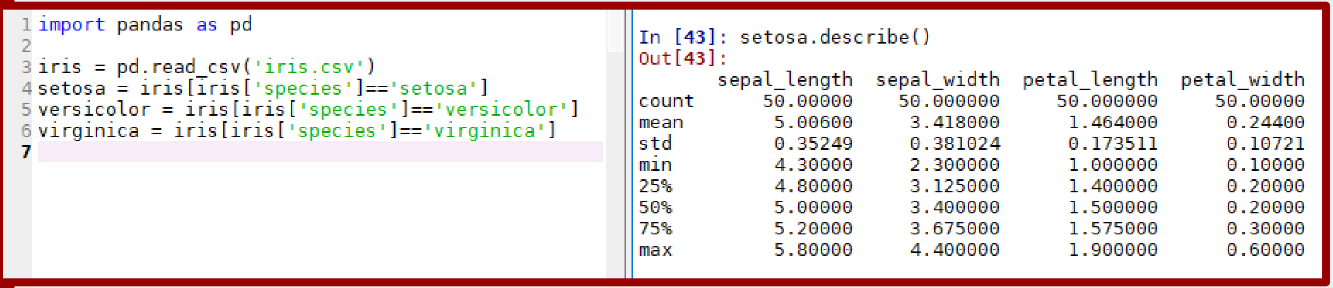
```python
import pandas as pd

iris = pd.read_csv('iris.csv')
setosa = iris[iris['species']=='setosa']
versicolor = iris[iris['species']=='versicolor']
virginica = iris[iris['species']=='virginica']
```

```
In [35]: setosa.head()
Out[35]:
   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

In [36]: versicolor.head()

        _length  sepal_width  petal_length  petal_width     species
            7.0          3.2           4.7          1.4  versicolor
            6.4          3.2           4.5          1.5  versicolor
            6.9          3.1           4.9          1.5  versicolor
            5.5          2.3           4.0          1.3  versicolor
            6.5          2.8           4.6          1.5  versicolor

         virginica.head()
Out[37]:
     sepal_length  sepal_width  petal_length  petal_width    species
100           6.3          3.3           6.0          2.5  virginica
101           5.8          2.7           5.1          1.9  virginica
102           7.1          3.0           5.9          2.1  virginica
103           6.3          2.9           5.6          1.8  virginica
104           6.5          3.0           5.8          2.2  virginica
```

We can now isolate specific classes of data with

Filtering DataFrame Rows

Using the DataFrama structure, we can also `describe()`

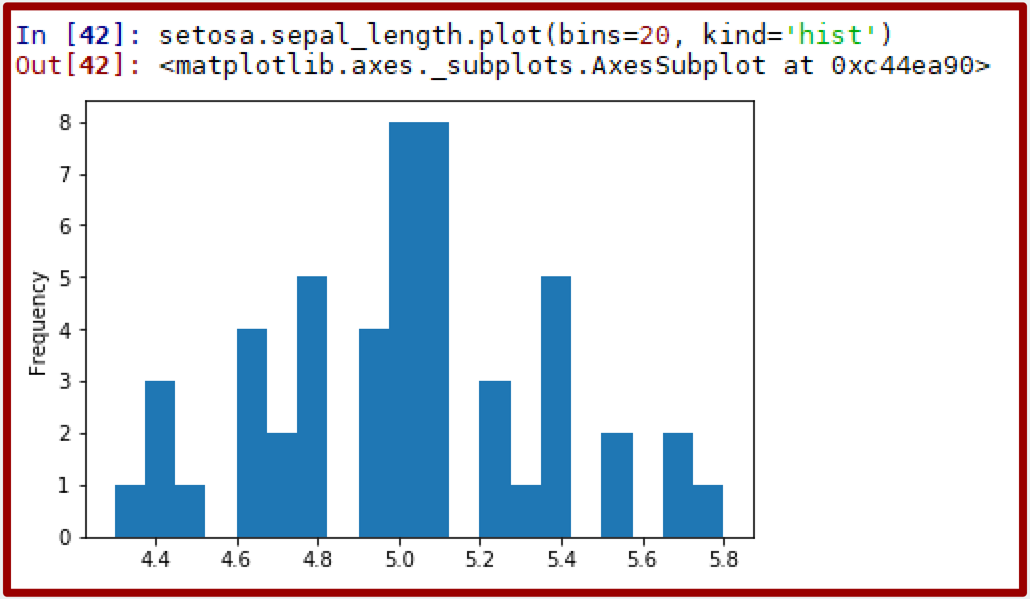


```python
import pandas as pd

iris = pd.read_csv('iris.csv')
setosa = iris[iris['species']=='setosa']
versicolor = iris[iris['species']=='versicolor']
virginica = iris[iris['species']=='virginica']
```

```
In [43]: setosa.describe()
Out[43]:
       sepal_length  sepal_width  petal_length  petal_width
count      50.00000    50.000000     50.000000     50.00000
mean        5.00600     3.418000      1.464000      0.24400
std         0.35249     0.381024      0.173511      0.10721
min         4.30000     2.300000      1.000000      0.10000
25%         4.80000     3.125000      1.400000      0.20000
50%         5.00000     3.400000      1.500000      0.20000
75%         5.20000     3.675000      1.575000      0.30000
max         5.80000     4.400000      1.900000      0.60000
```

Filtering DataFrame Rows

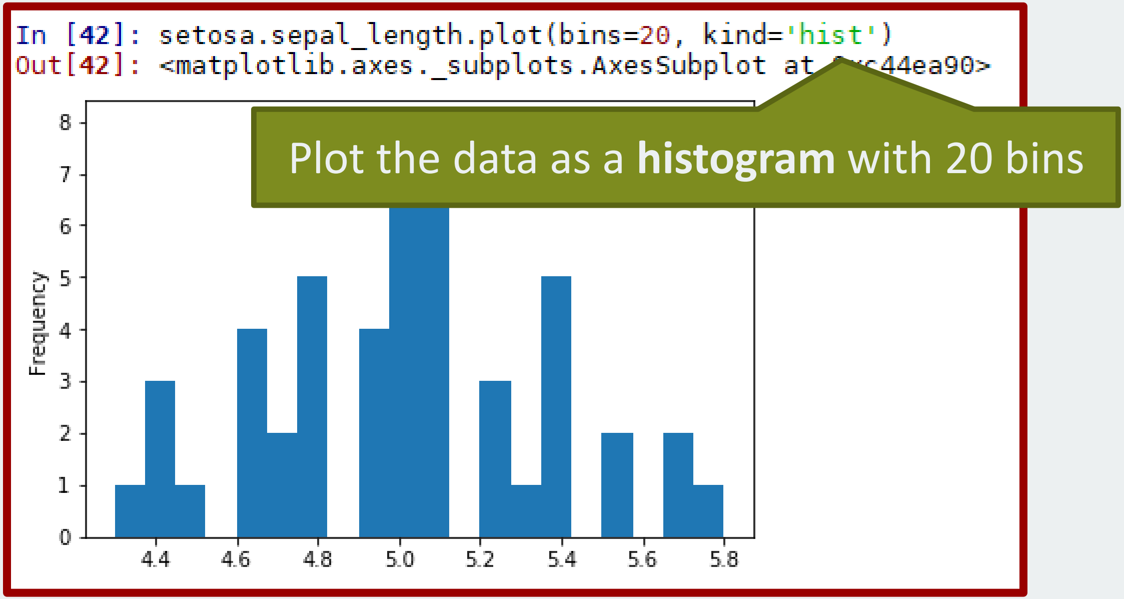
Using the DataFrama structure, we can also `describe()` and `plot()` our data



```
In [42]: setosa.sepal_length.plot(bins=20, kind='hist')
Out[42]: <matplotlib.axes._subplots.AxesSubplot at 0xc44ea90>
```

Filtering DataFrame Rows

Using the DataFrama structure, we can also `describe()` and `plot()` our data

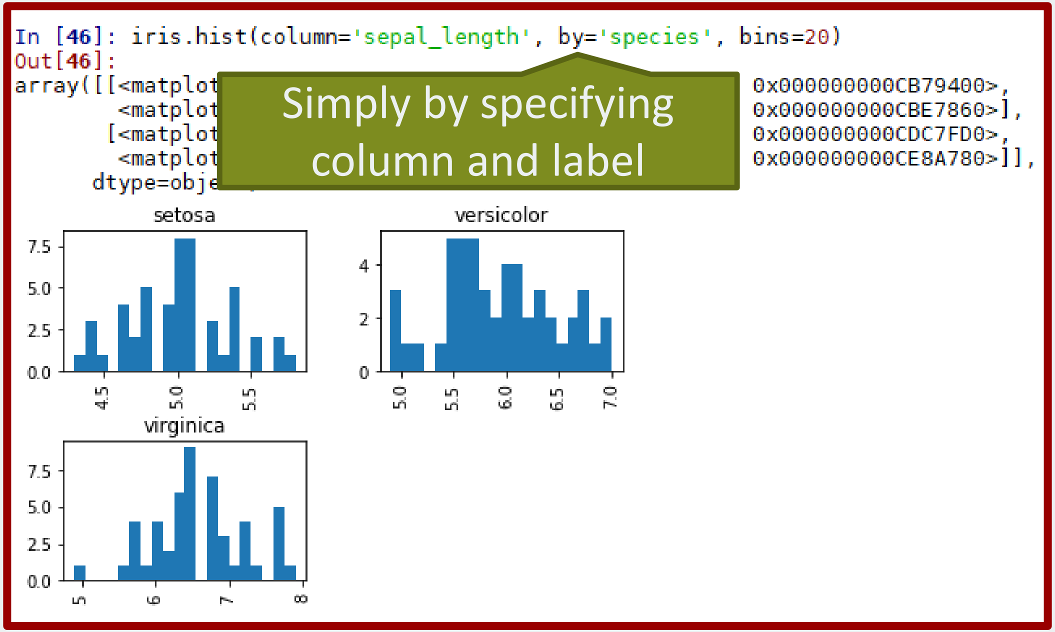


```
In [42]: setosa.sepal_length.plot(bins=20, kind='hist')
Out[42]: <matplotlib.axes._subplots.AxesSubplot at 0xc44ea90>
```

Plot the data as a **histogram** with 20 bins

Filtering DataFrame Rows

We can also do the same thing across the whole DataFrame with `.hist()`



```
In [46]: iris.hist(column='sepal_length', by='species', bins=20)
Out[46]:
array([[<matplot                                                  0x000000000CB79400>,
        <matplot                                                  0x000000000CBE7860>],
       [<matplot                                                  0x000000000CDC7FD0>,
        <matplot                                                  0x000000000CE8A780>]],
      dtype=obje
```

Simply by specifying column and label