Using Python for Data Science

Hongchang Gao

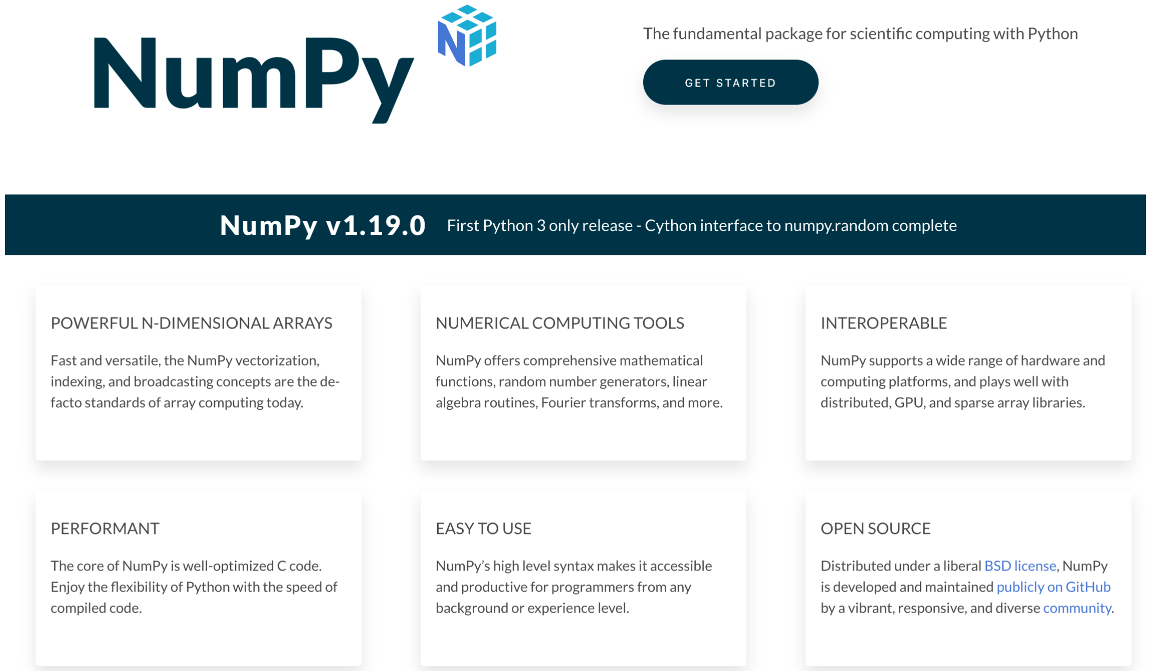
Spring 2024

Python for Data Science

- Packages:
  - Numpy
  - Scikit-learn (sklearn)
  - Pytorch
  - Tensorflow
  - Pandas
  - Matplotlib

Python for Data Science: NumPy

- NumPy: fundamental package for scientific computing
  - Large, multi-dimensional arrays and matrices
  - High-level mathematical functions



NumPy

The fundamental package for scientific computing with Python

GET STARTED

NumPy v1.19.0 First Python 3 only release - Cython interface to numpy.random complete

POWERFUL N-DIMENSIONAL ARRAYS

Fast and versatile, the NumPy vectorization, indexing, and broadcasting concepts are the de-facto standards of array computing today.

NUMERICAL COMPUTING TOOLS

NumPy offers comprehensive mathematical functions, random number generators, linear algebra routines, Fourier transforms, and more.

INTEROPERABLE

NumPy supports a wide range of hardware and computing platforms, and plays well with distributed, GPU, and sparse array libraries.

PERFORMANT

The core of NumPy is well-optimized C code. Enjoy the flexibility of Python with the speed of compiled code.

EASY TO USE

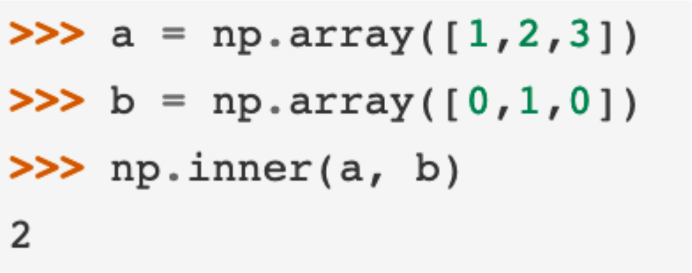
NumPy's high level syntax makes it accessible and productive for programmers from any background or experience level.

OPEN SOURCE

Distributed under a liberal BSD license, NumPy is developed and maintained publicly on GitHub by a vibrant, responsive, and diverse community.

Python for Data Science: NumPy

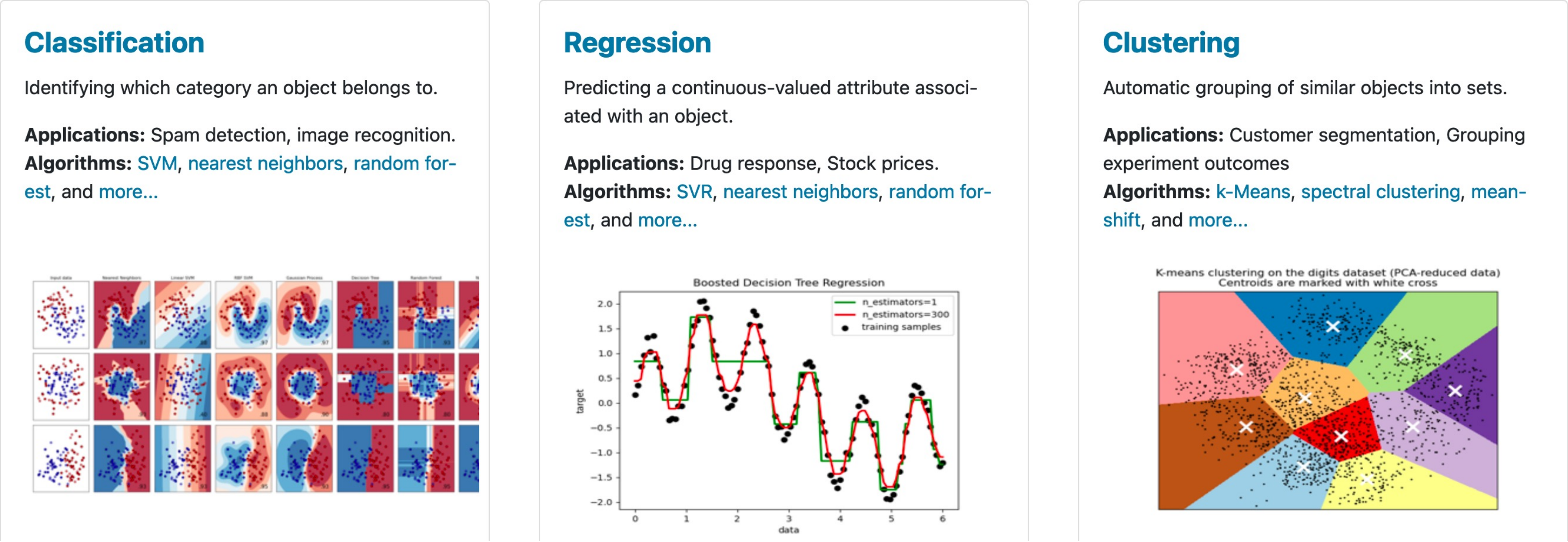
- NumPy: fundamental package for scientific computing
  - 1-D array: `arr = np.array([1, 2, 3, 4, 5])`
  - 2-D array `arr = np.array([[1, 2, 3], [4, 5, 6]])`
  - Array indexing: `arr[2] + arr[3]`
  - Mathematic operation



```
>>> a = np.array([1,2,3])
>>> b = np.array([0,1,0])
>>> np.inner(a, b)
2
```

Python for Data Science: scikit-learn (sklearn)

- Machine learning package in python



Classification

Identifying which category an object belongs to.

Applications: Spam detection, image recognition.
Algorithms: SVM, nearest neighbors, random forest, and more...

Regression

Predicting a continuous-valued attribute associated with an object.

Applications: Drug response, Stock prices.
Algorithms: SVR, nearest neighbors, random forest, and more...

Clustering

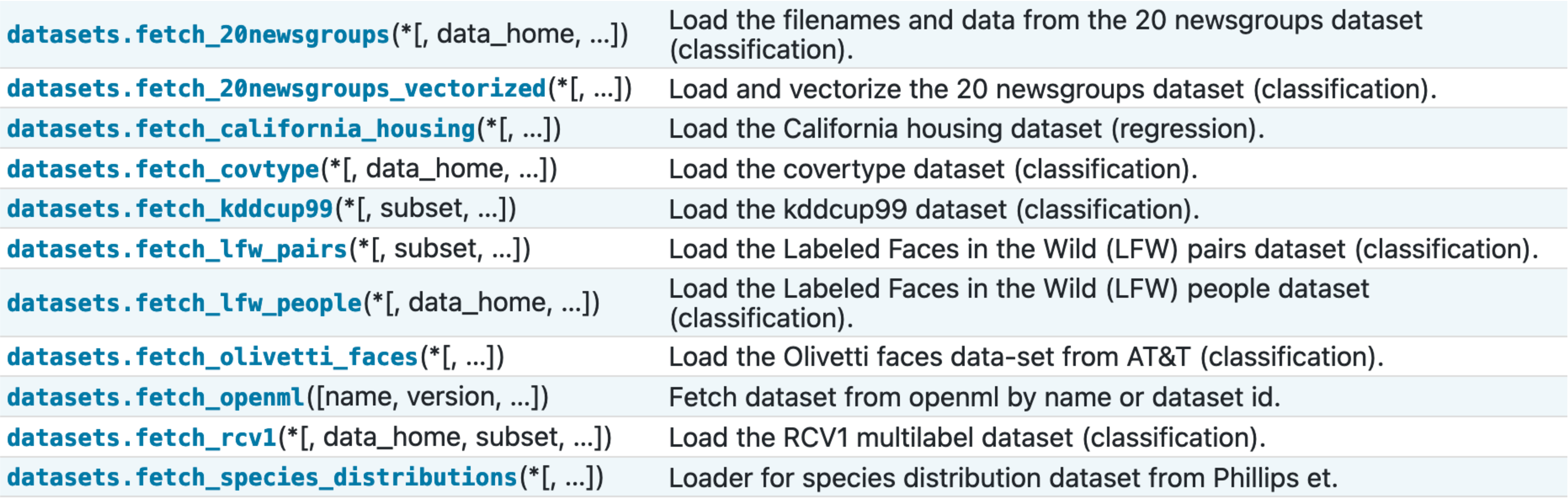
Automatic grouping of similar objects into sets.

Applications: Customer segmentation, Grouping experiment outcomes
Algorithms: k-Means, spectral clustering, mean-shift, and more...

Boosted Decision Tree Regression
n_estimators=1
n_estimators=300
training samples
target
data
2.0
1.5
1.0
0.5
0.0
−0.5
−1.0
−1.5
−2.0
0
1
2
3
4
5
6
K-means clustering on the digits dataset (PCA-reduced data)
Centroids are marked with white cross

Python for Data Science: scikit-learn

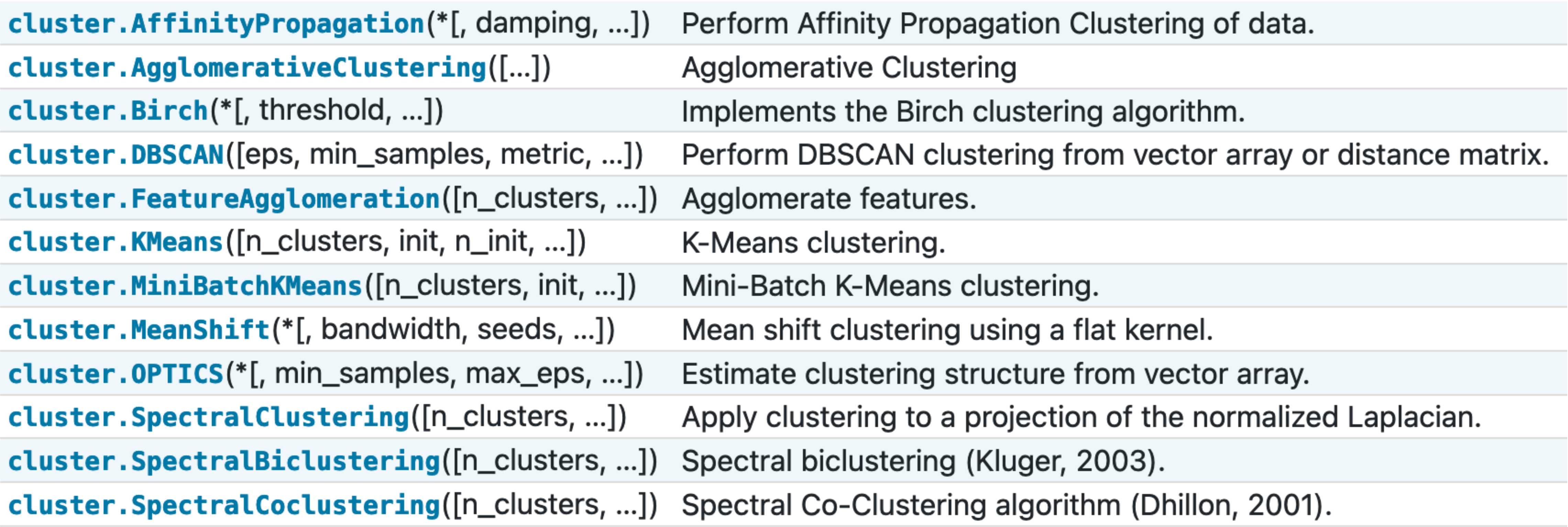
- It includes various datasets for machine learning tasks



`datasets.fetch_20newsgroups`(*[, data_home, ...])	Load the filenames and data from the 20 newsgroups dataset (classification).
`datasets.fetch_20newsgroups_vectorized`(*[, ...])	Load and vectorize the 20 newsgroups dataset (classification).
`datasets.fetch_california_housing`(*[, ...])	Load the California housing dataset (regression).
`datasets.fetch_covtype`(*[, data_home, ...])	Load the covertype dataset (classification).
`datasets.fetch_kddcup99`(*[, subset, ...])	Load the kddcup99 dataset (classification).
`datasets.fetch_lfw_pairs`(*[, subset, ...])	Load the Labeled Faces in the Wild (LFW) pairs dataset (classification).
`datasets.fetch_lfw_people`(*[, data_home, ...])	Load the Labeled Faces in the Wild (LFW) people dataset (classification).
`datasets.fetch_olivetti_faces`(*[, ...])	Load the Olivetti faces data-set from AT&T (classification).
`datasets.fetch_openml`([name, version, ...])	Fetch dataset from openml by name or dataset id.
`datasets.fetch_rcv1`(*[, data_home, subset, ...])	Load the RCV1 multilabel dataset (classification).
`datasets.fetch_species_distributions`(*[, ...])	Loader for species distribution dataset from Phillips et.

Python for Data Science: scikit-learn

- It includes various state-of-the-art machine learning models

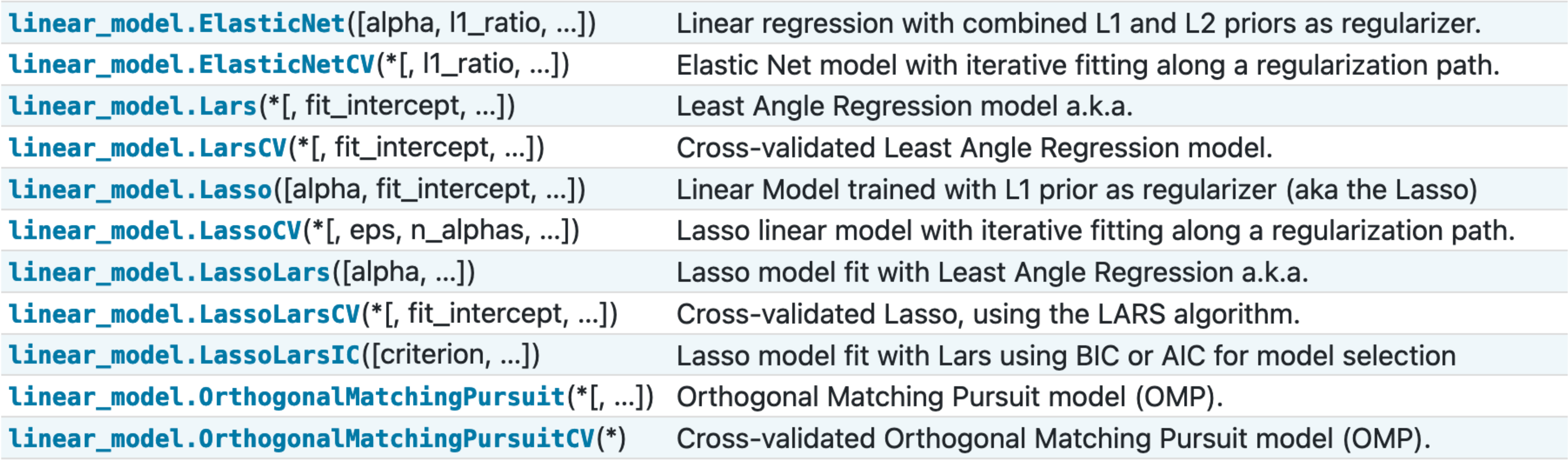


cluster.AffinityPropagation(*[, damping, ...])	Perform Affinity Propagation Clustering of data.
cluster.AgglomerativeClustering([...])	Agglomerative Clustering
cluster.Birch(*[, threshold, ...])	Implements the Birch clustering algorithm.
cluster.DBSCAN([eps, min_samples, metric, ...])	Perform DBSCAN clustering from vector array or distance matrix.
cluster.FeatureAgglomeration([n_clusters, ...])	Agglomerate features.
cluster.KMeans([n_clusters, init, n_init, ...])	K-Means clustering.
cluster.MiniBatchKMeans([n_clusters, init, ...])	Mini-Batch K-Means clustering.
cluster.MeanShift(*[, bandwidth, seeds, ...])	Mean shift clustering using a flat kernel.
cluster.OPTICS(*[, min_samples, max_eps, ...])	Estimate clustering structure from vector array.
cluster.SpectralClustering([n_clusters, ...])	Apply clustering to a projection of the normalized Laplacian.
cluster.SpectralBiclustering([n_clusters, ...])	Spectral biclustering (Kluger, 2003).
cluster.SpectralCoclustering([n_clusters, ...])	Spectral Co-Clustering algorithm (Dhillon, 2001).

Unsupervised

Python for Data Science: scikit-learn

- It includes various state-of-the-art machine learning models

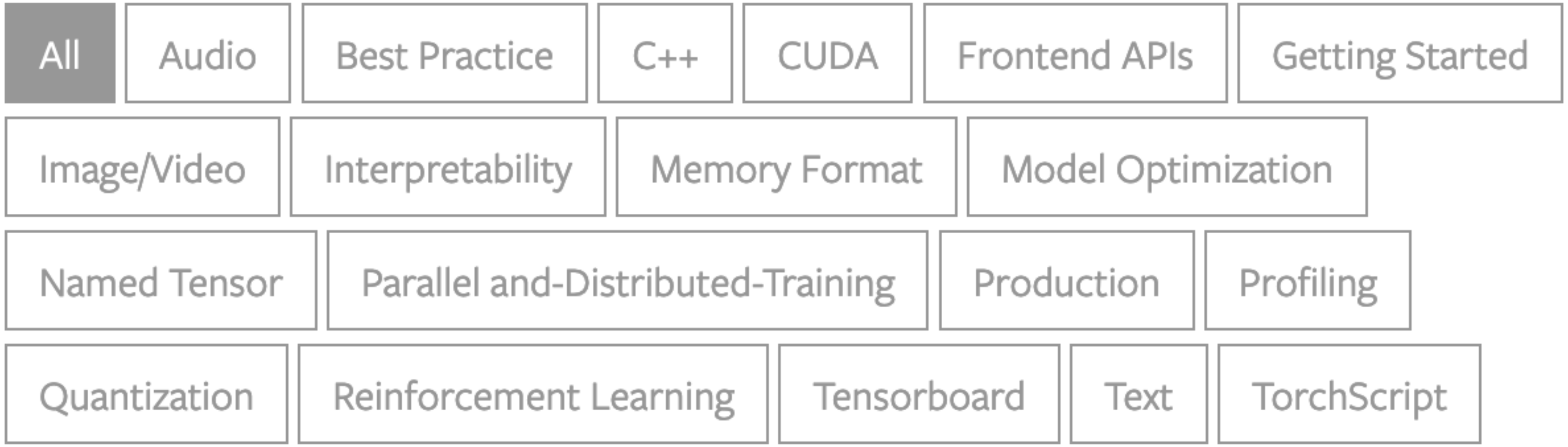


`linear_model.ElasticNet`([alpha, l1_ratio, ...])	Linear regression with combined L1 and L2 priors as regularizer.
`linear_model.ElasticNetCV`(*[, l1_ratio, ...])	Elastic Net model with iterative fitting along a regularization path.
`linear_model.Lars`(*[, fit_intercept, ...])	Least Angle Regression model a.k.a.
`linear_model.LarsCV`(*[, fit_intercept, ...])	Cross-validated Least Angle Regression model.
`linear_model.Lasso`([alpha, fit_intercept, ...])	Linear Model trained with L1 prior as regularizer (aka the Lasso)
`linear_model.LassoCV`(*[, eps, n_alphas, ...])	Lasso linear model with iterative fitting along a regularization path.
`linear_model.LassoLars`([alpha, ...])	Lasso model fit with Least Angle Regression a.k.a.
`linear_model.LassoLarsCV`(*[, fit_intercept, ...])	Cross-validated Lasso, using the LARS algorithm.
`linear_model.LassoLarsIC`([criterion, ...])	Lasso model fit with Lars using BIC or AIC for model selection
`linear_model.OrthogonalMatchingPursuit`(*[, ...])	Orthogonal Matching Pursuit model (OMP).
`linear_model.OrthogonalMatchingPursuitCV`(*)	Cross-validated Orthogonal Matching Pursuit model (OMP).

Supervised

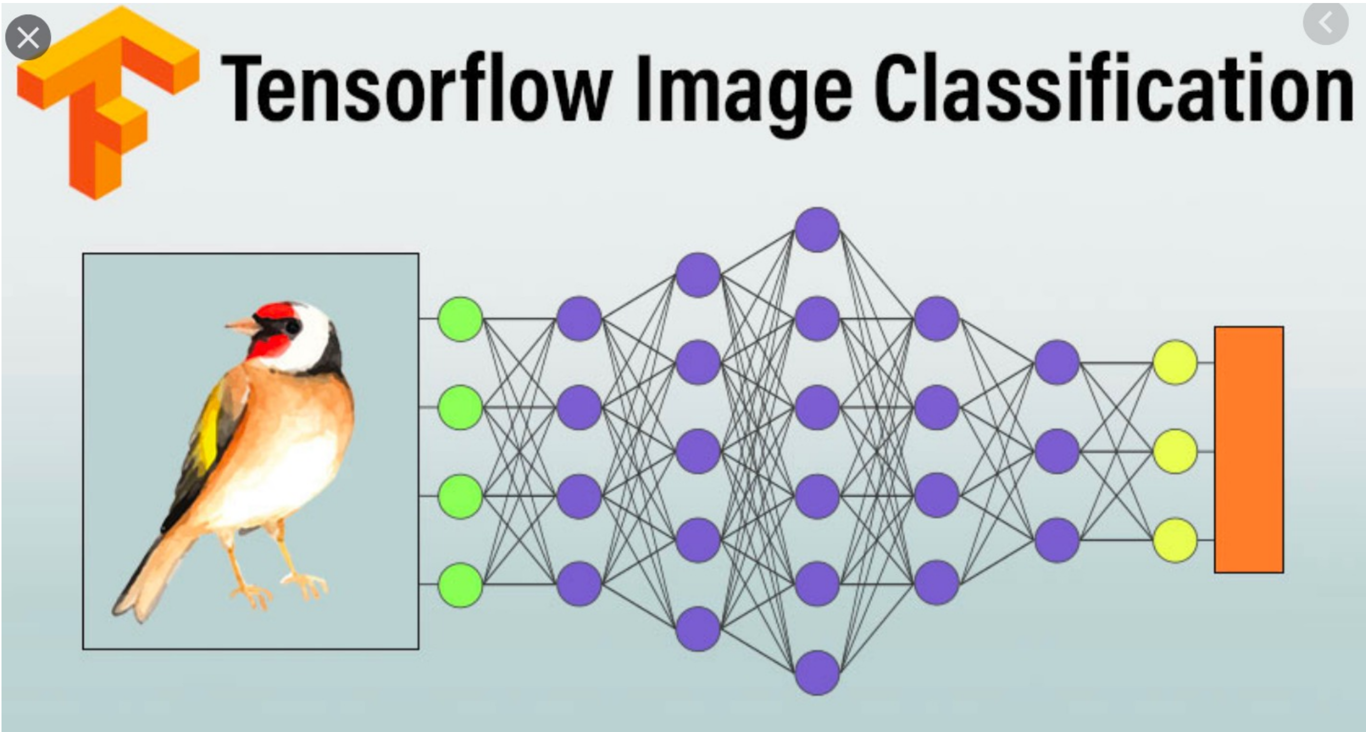
Python for Data Science: PyTorch

- Deep Learning tools in Python, developed by Facebook



Python for Data Science: Tensorflow

- Deep Learning tools in Python, developed by Google

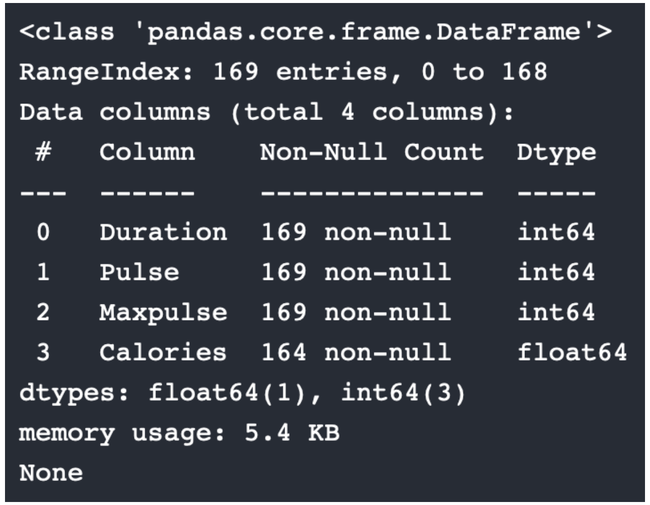


Python for Data Science: Pandas

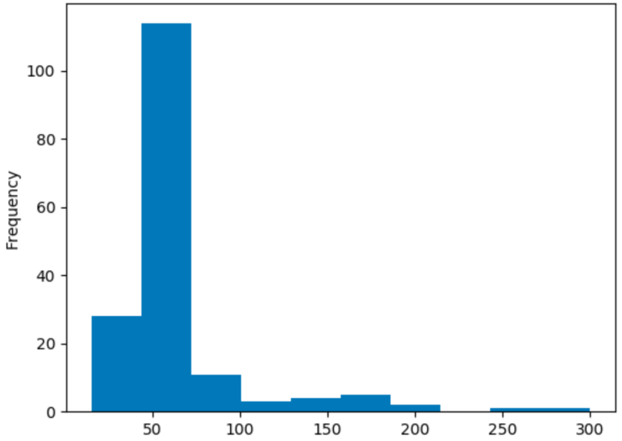
- A Python package for working with datasets
- Has functions for
  - Analyze data
  - Clean data
  - Explore data
  - Manipulate data

```
import pandas as pd

df = pd.read_csv('data.csv')
```

```
import pandas as pd

df = pd.read_json('data.json')
```

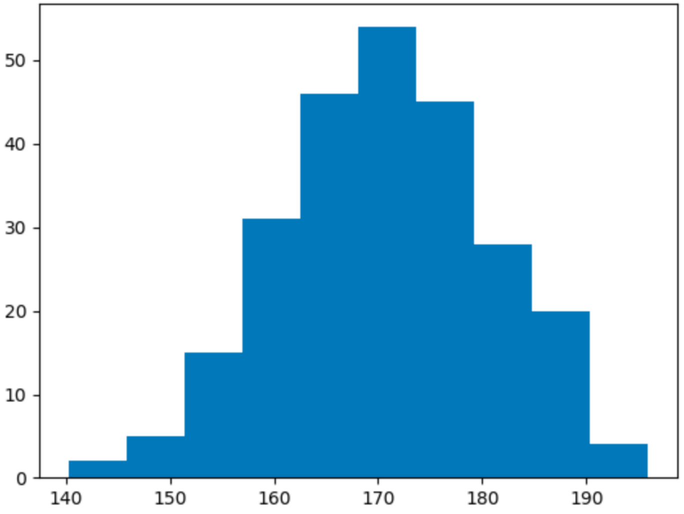


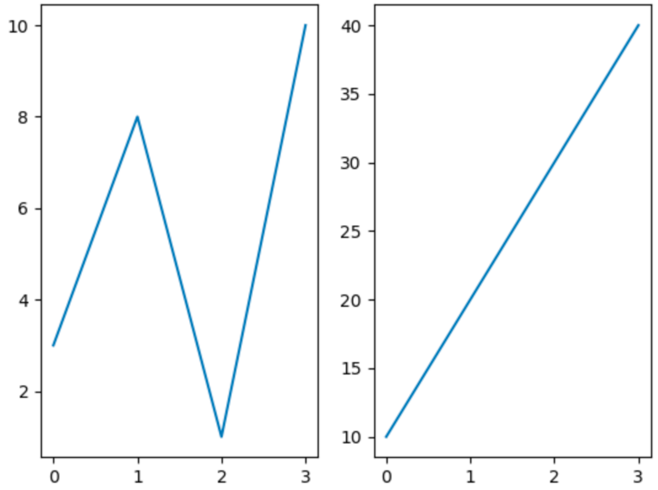
```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 169 entries, 0 to 168
Data columns (total 4 columns):
 #   Column    Non-Null Count  Dtype
---  ------    --------------  -----
 0   Duration  169 non-null    int64
 1   Pulse     169 non-null    int64
 2   Maxpulse  169 non-null    int64
 3   Calories  164 non-null    float64
dtypes: float64(1), int64(3)
memory usage: 5.4 KB
None
```

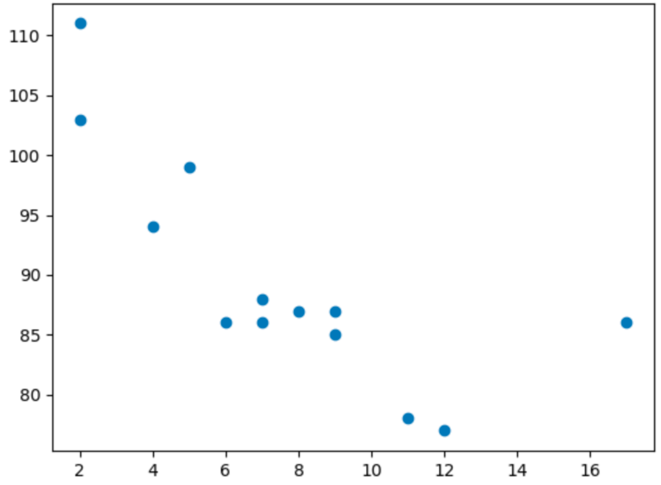


Python for Data Science: Matplotlib

- A Python package for visualizing data





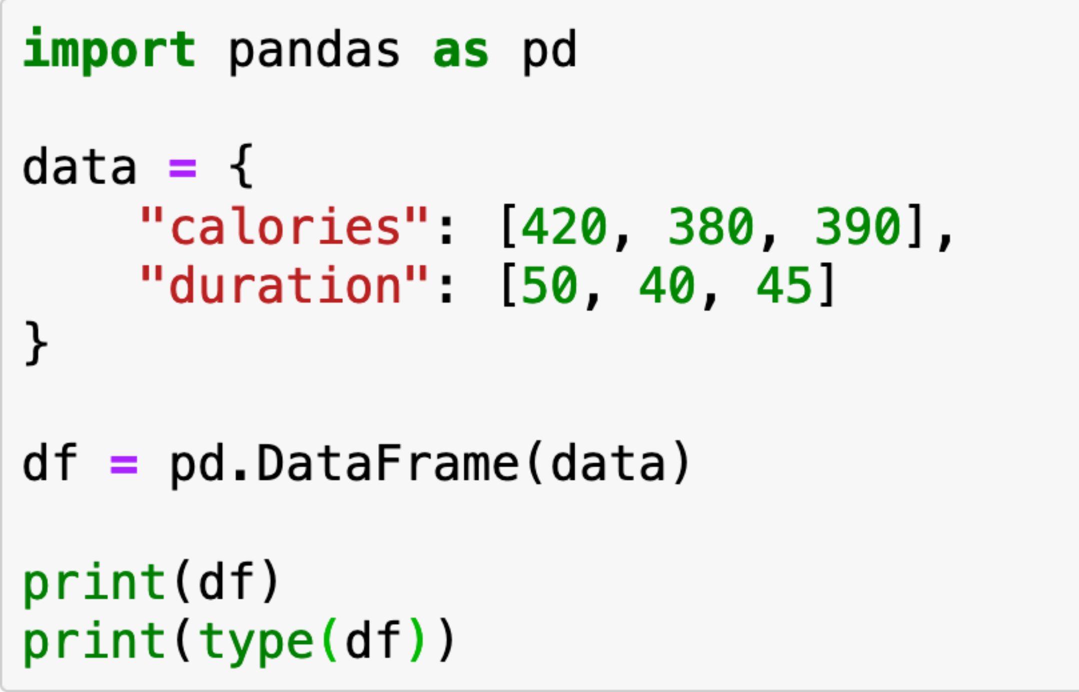


Pandas

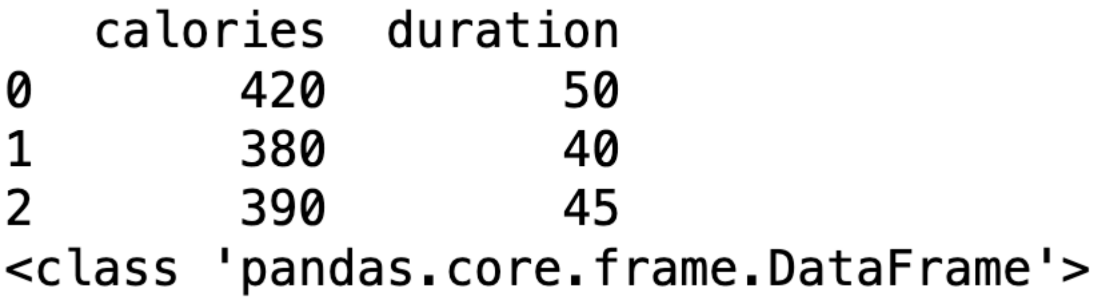
- A python library for working with data sets
- Has functions for
  - Analyzing data
  - Cleaning data
  - Exploring data
  - Manipulating data

Pandas

- DataFrame
  - A 2-dimensional data structure, like a 2-dimensional array, or a table with rows and columns.



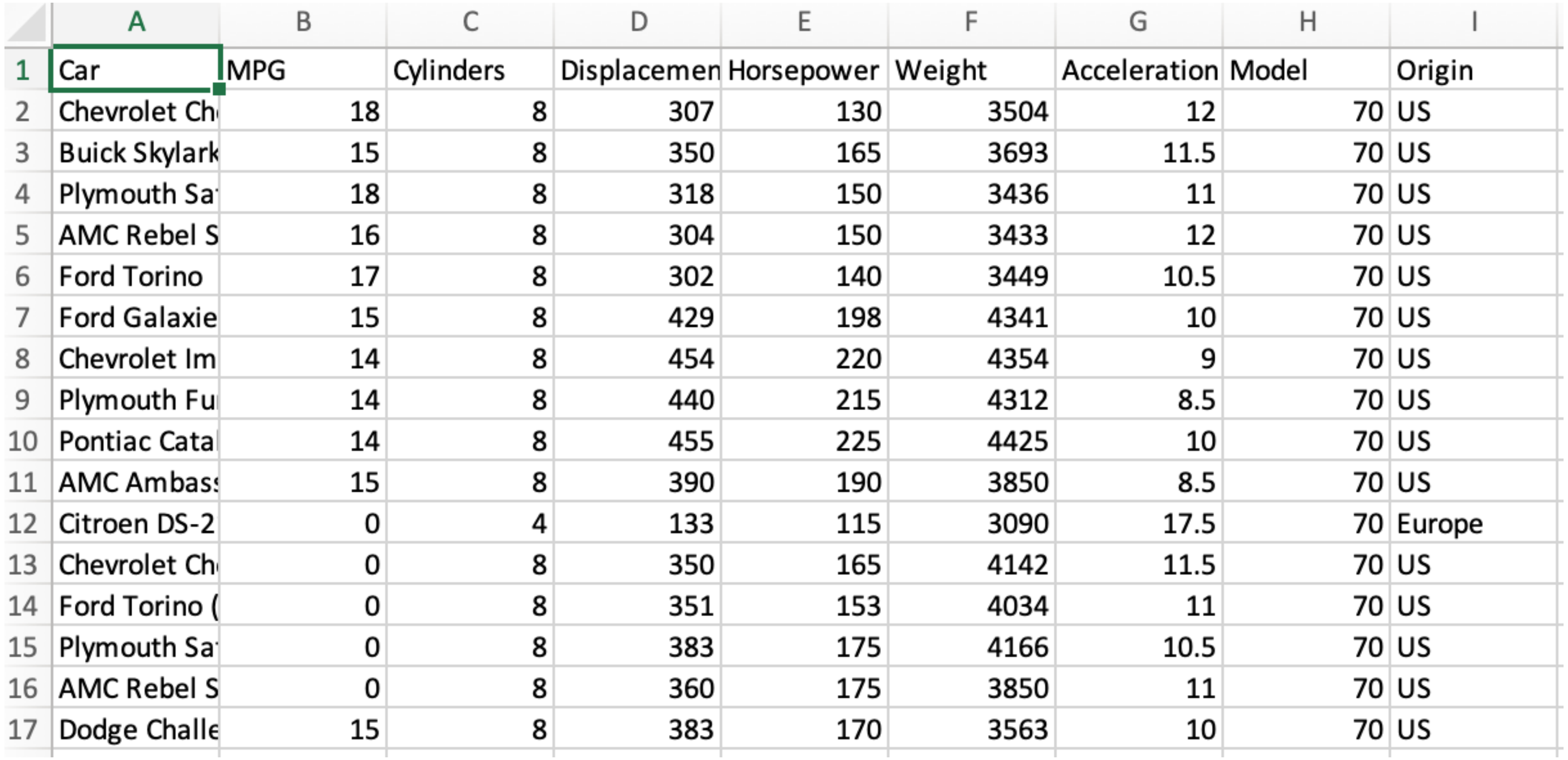
```python
import pandas as pd

data = {
    "calories": [420, 380, 390],
    "duration": [50, 40, 45]
}

df = pd.DataFrame(data)

print(df)
print(type(df))
```



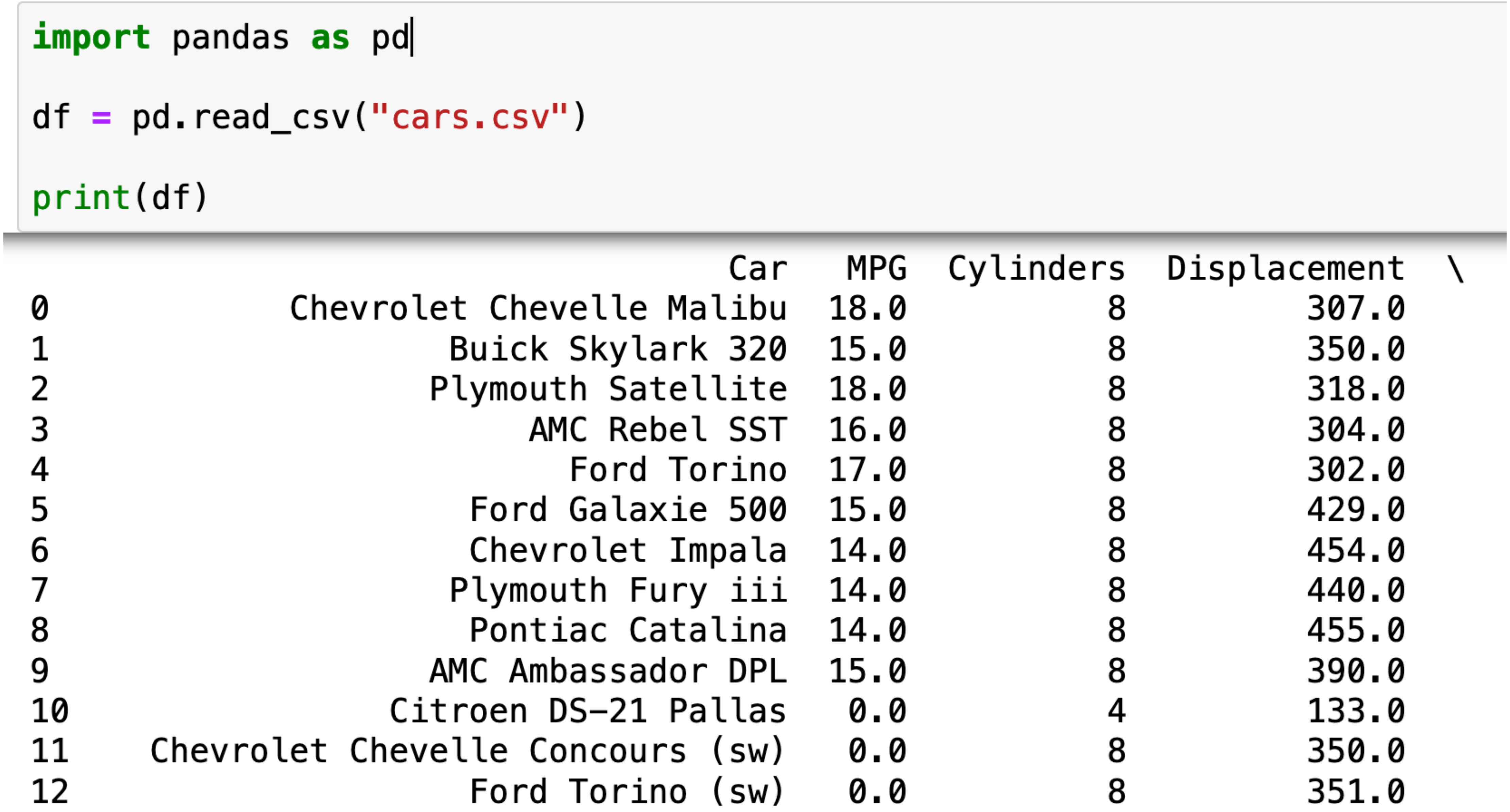
```
   calories  duration
0       420        50
1       380        40
2       390        45
<class 'pandas.core.frame.DataFrame'>
```

Pandas

- Load CSV file



	A	B	C	D	E	F	G	H	I
1	Car	MPG	Cylinders	Displacemen	Horsepower	Weight	Acceleration	Model	Origin
2	Chevrolet Ch	18	8	307	130	3504	12	70	US
3	Buick Skylark	15	8	350	165	3693	11.5	70	US
4	Plymouth Sa	18	8	318	150	3436	11	70	US
5	AMC Rebel S	16	8	304	150	3433	12	70	US
6	Ford Torino	17	8	302	140	3449	10.5	70	US
7	Ford Galaxie	15	8	429	198	4341	10	70	US
8	Chevrolet Im	14	8	454	220	4354	9	70	US
9	Plymouth Fu	14	8	440	215	4312	8.5	70	US
10	Pontiac Catal	14	8	455	225	4425	10	70	US
11	AMC Ambass	15	8	390	190	3850	8.5	70	US
12	Citroen DS-2	0	4	133	115	3090	17.5	70	Europe
13	Chevrolet Ch	0	8	350	165	4142	11.5	70	US
14	Ford Torino (	0	8	351	153	4034	11	70	US
15	Plymouth Sa	0	8	383	175	4166	10.5	70	US
16	AMC Rebel S	0	8	360	175	3850	11	70	US
17	Dodge Challe	15	8	383	170	3563	10	70	US

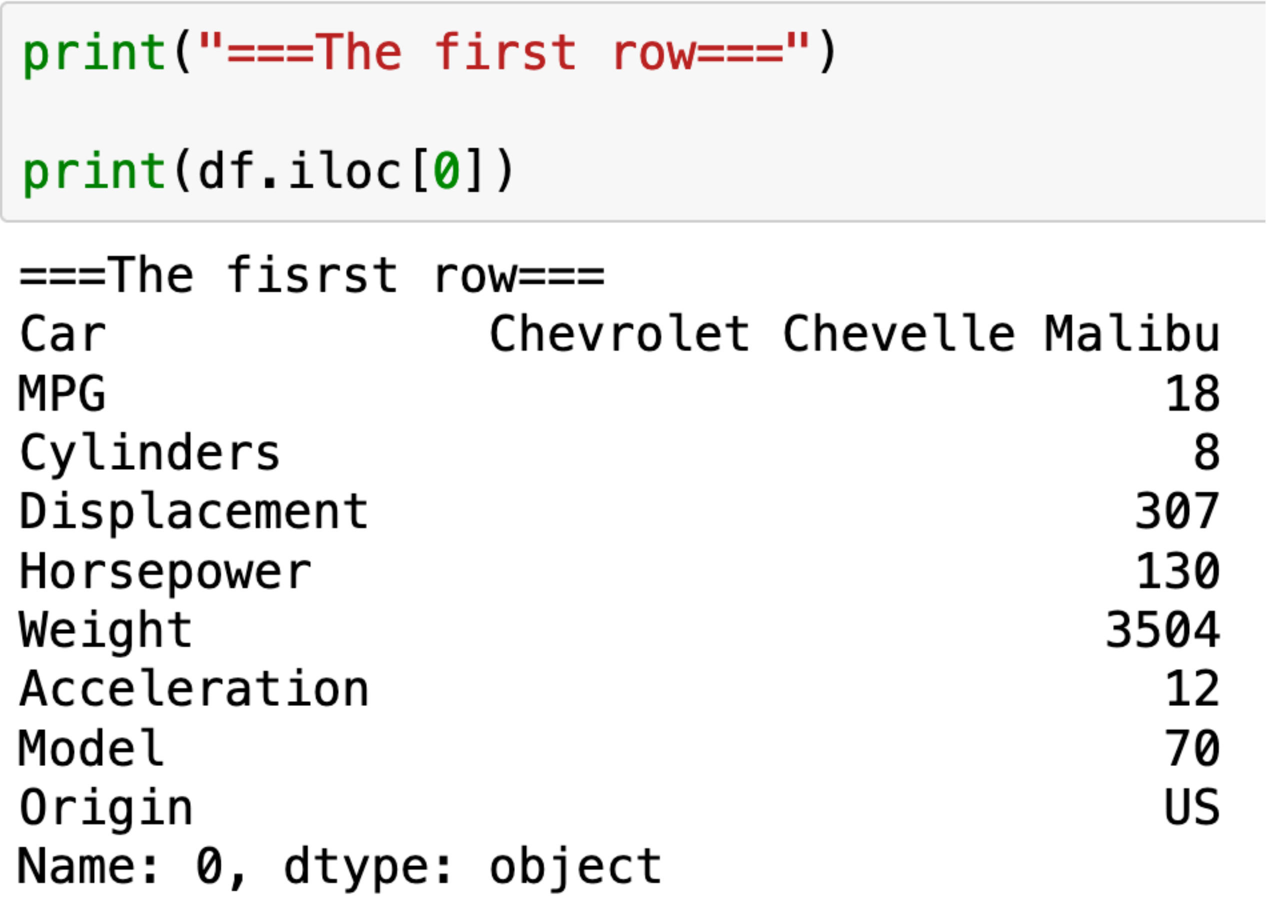


```python
import pandas as pd

df = pd.read_csv("cars.csv")

print(df)
```

```
                                   Car   MPG  Cylinders  Displacement  \
0            Chevrolet Chevelle Malibu  18.0          8         307.0   
1                    Buick Skylark 320  15.0          8         350.0   
2                   Plymouth Satellite  18.0          8         318.0   
3                        AMC Rebel SST  16.0          8         304.0   
4                          Ford Torino  17.0          8         302.0   
5                     Ford Galaxie 500  15.0          8         429.0   
6                     Chevrolet Impala  14.0          8         454.0   
7                    Plymouth Fury iii  14.0          8         440.0   
8                     Pontiac Catalina  14.0          8         455.0   
9                   AMC Ambassador DPL  15.0          8         390.0   
10                Citroen DS-21 Pallas   0.0          4         133.0   
11    Chevrolet Chevelle Concours (sw)   0.0          8         350.0   
12                    Ford Torino (sw)   0.0          8         351.0   
```

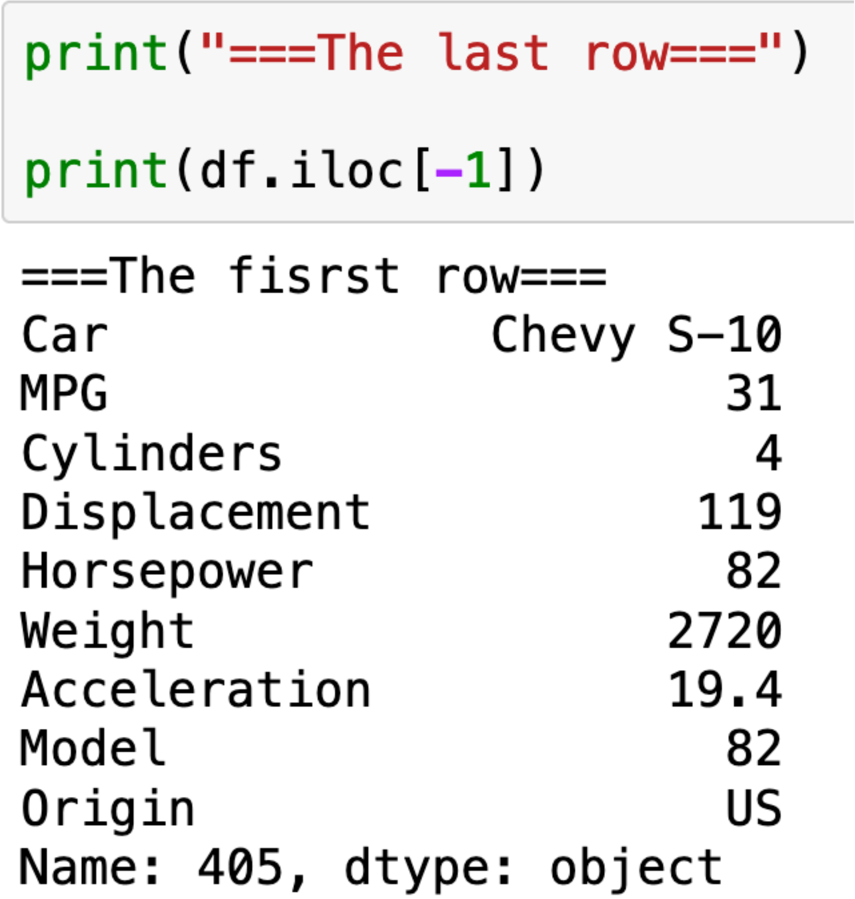
Pandas

- Get rows



```
print("===The first row===")

print(df.iloc[0])
```

```
===The fisrst row===
Car             Chevrolet Chevelle Malibu
MPG                                    18
Cylinders                               8
Displacement                          307
Horsepower                            130
Weight                               3504
Acceleration                           12
Model                                  70
Origin                                 US
Name: 0, dtype: object
```

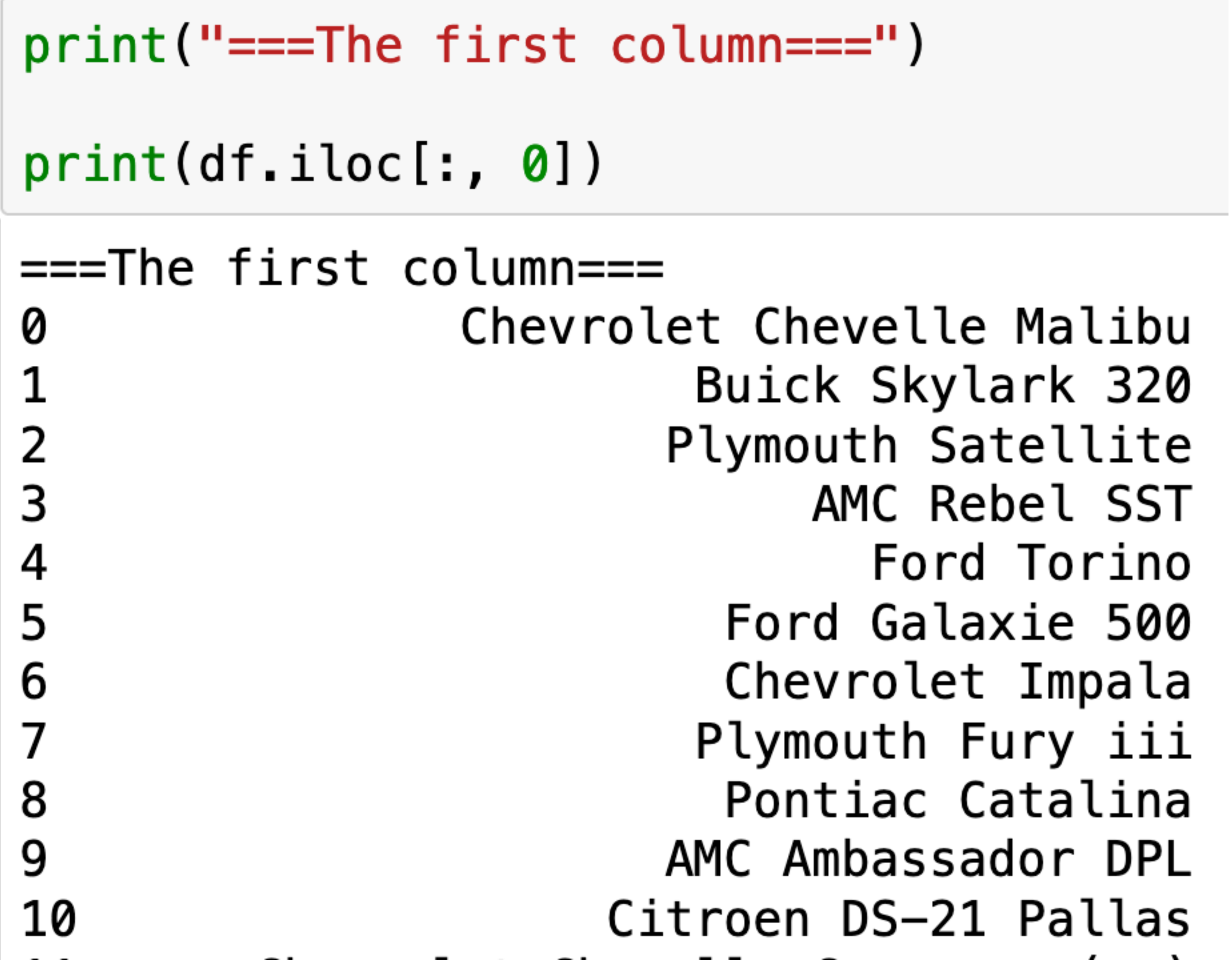


```
print("===The last row===")

print(df.iloc[-1])
```

```
===The fisrst row===
Car             Chevy S-10
MPG                     31
Cylinders                4
Displacement           119
Horsepower              82
Weight                2720
Acceleration          19.4
Model                   82
Origin                  US
Name: 405, dtype: object
```

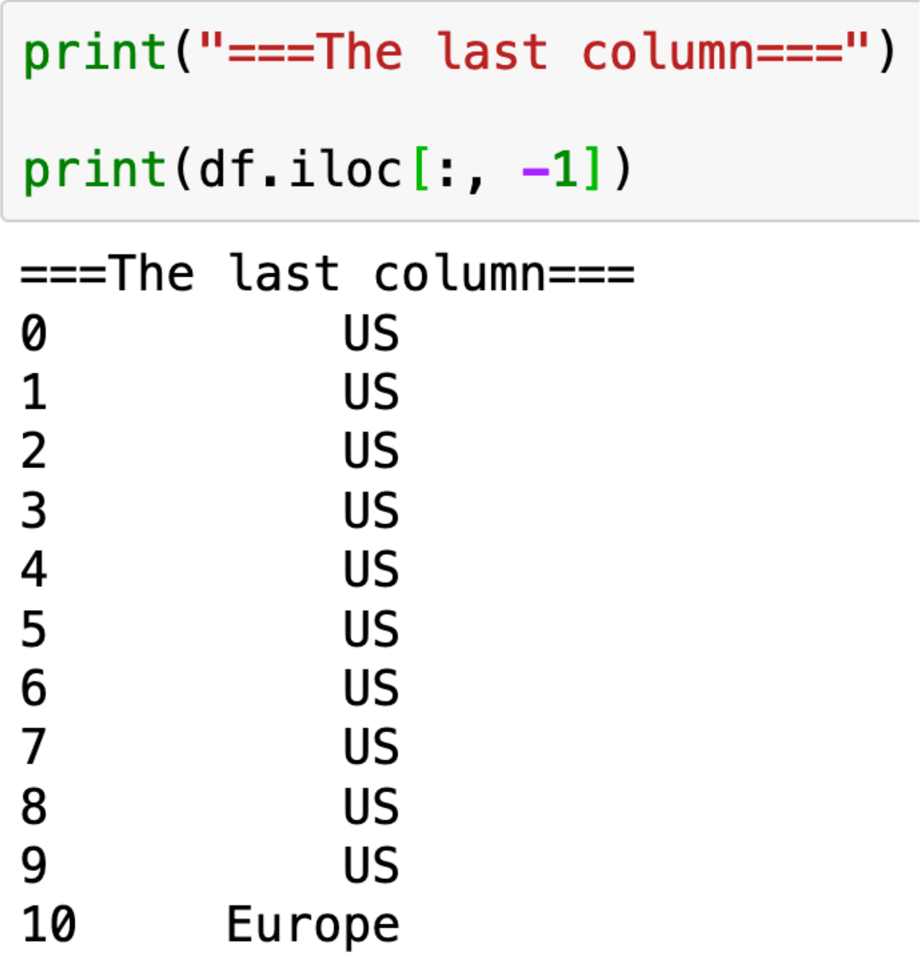
Pandas

- Get columns



```python
print("===The first column===")

print(df.iloc[:, 0])
```

```
===The first column===
0              Chevrolet Chevelle Malibu
1                      Buick Skylark 320
2                     Plymouth Satellite
3                          AMC Rebel SST
4                            Ford Torino
5                       Ford Galaxie 500
6                       Chevrolet Impala
7                      Plymouth Fury iii
8                       Pontiac Catalina
9                     AMC Ambassador DPL
10                  Citroen DS-21 Pallas
```

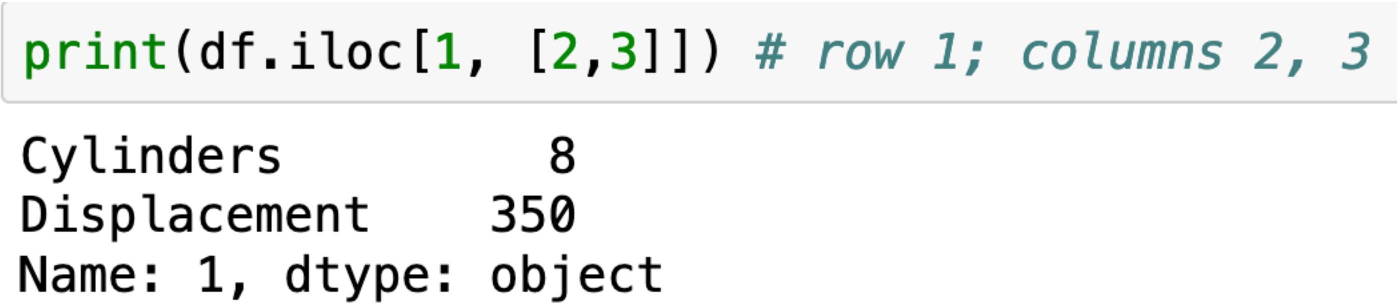


```python
print("===The last column===")

print(df.iloc[:, -1])
```

```
===The last column===
0          US
1          US
2          US
3          US
4          US
5          US
6          US
7          US
8          US
9          US
10     Europe
```

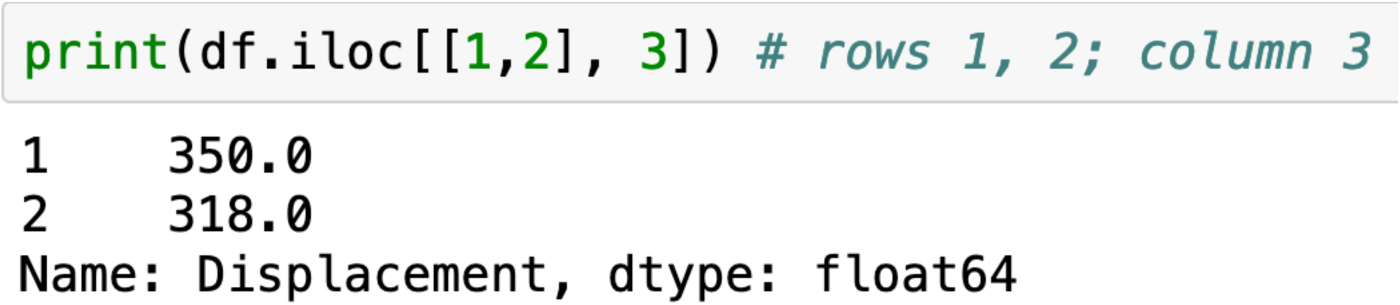
Pandas

- Get multiple rows or columns



```
print(df.iloc[1, [2,3]]) # row 1; columns 2, 3
```

```
Cylinders         8
Displacement    350
Name: 1, dtype: object
```

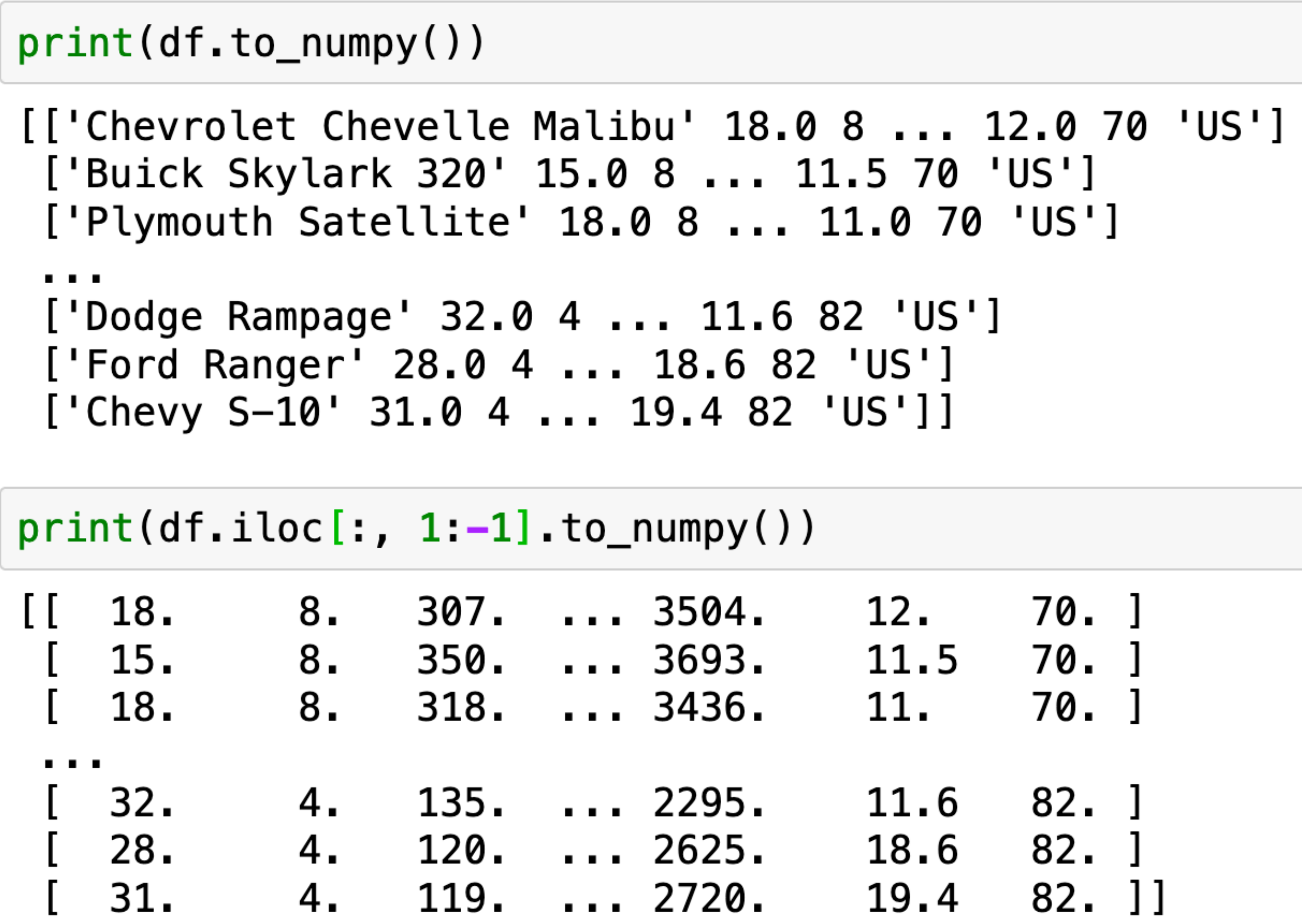


```
print(df.iloc[[1,2], 3]) # rows 1, 2; column 3
```

```
1    350.0
2    318.0
Name: Displacement, dtype: float64
```

Pandas

- Convert DataFrame to numpy Array



```
print(df.to_numpy())
```

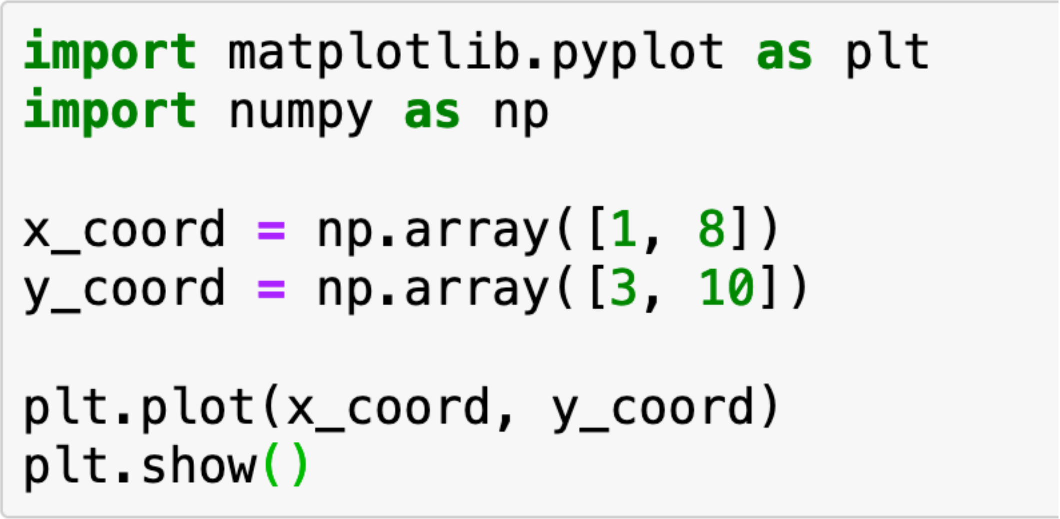
```
[['Chevrolet Chevelle Malibu' 18.0 8 ... 12.0 70 'US']
 ['Buick Skylark 320' 15.0 8 ... 11.5 70 'US']
 ['Plymouth Satellite' 18.0 8 ... 11.0 70 'US']
 ...
 ['Dodge Rampage' 32.0 4 ... 11.6 82 'US']
 ['Ford Ranger' 28.0 4 ... 18.6 82 'US']
 ['Chevy S-10' 31.0 4 ... 19.4 82 'US']]
```

```
print(df.iloc[:, 1:-1].to_numpy())
```

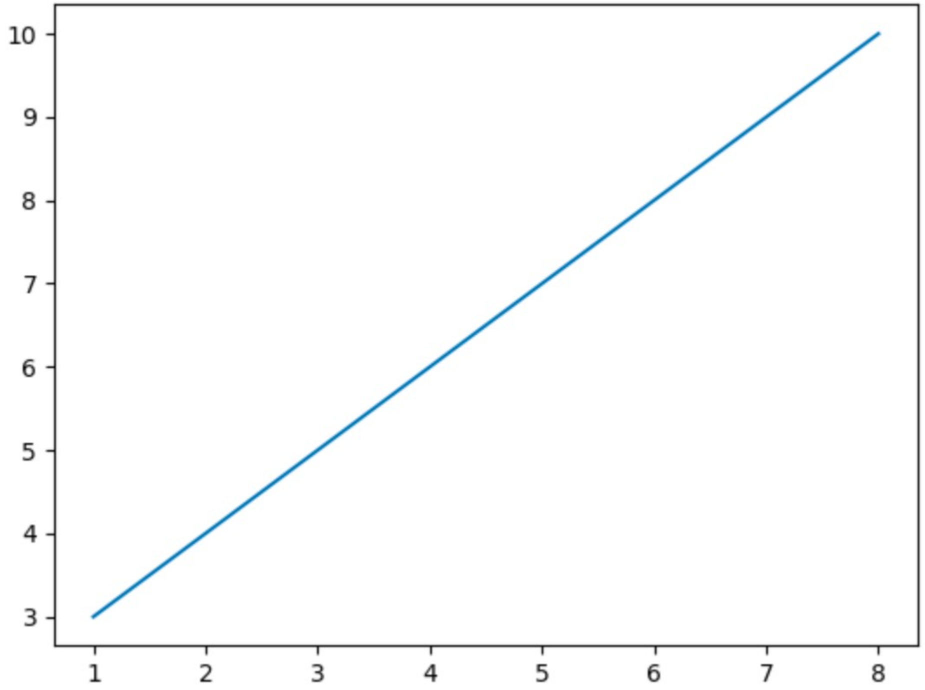
```
[[  18.     8.   307.  ... 3504.    12.     70. ]
 [  15.     8.   350.  ... 3693.    11.5    70. ]
 [  18.     8.   318.  ... 3436.    11.     70. ]
 ...
 [  32.     4.   135.  ... 2295.    11.6    82. ]
 [  28.     4.   120.  ... 2625.    18.6    82. ]
 [  31.     4.   119.  ... 2720.    19.4    82. ]]
```

Matplotlib

- A Python package for visualization
  - Plot: draws a line from point to point

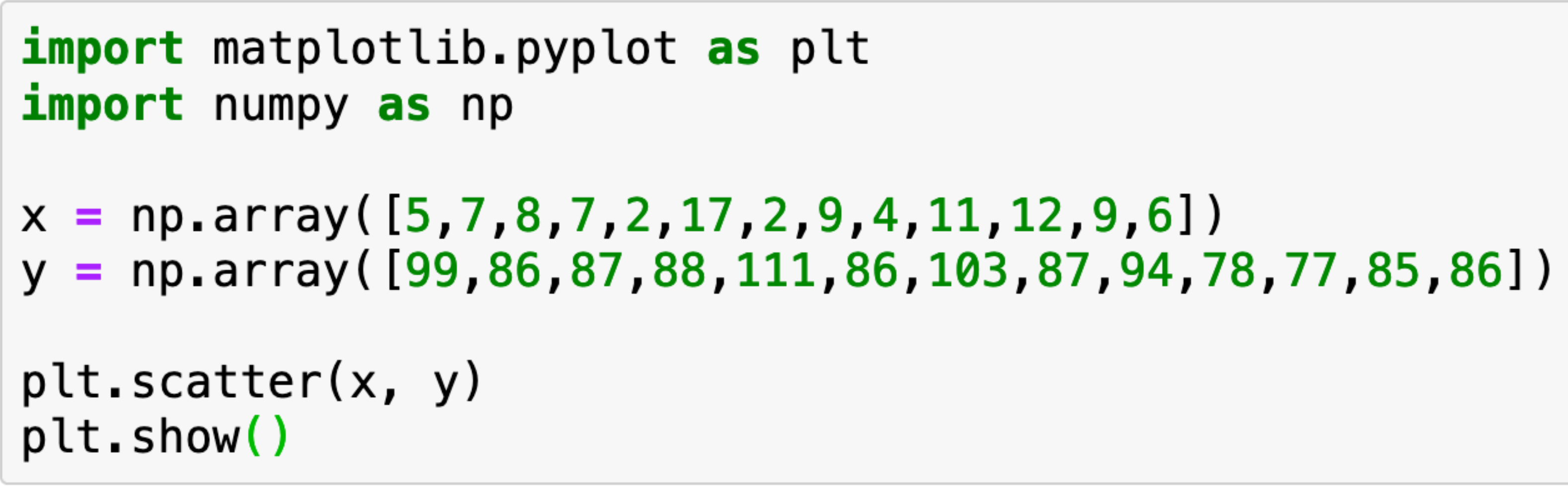


```
import matplotlib.pyplot as plt
import numpy as np

x_coord = np.array([1, 8])
y_coord = np.array([3, 10])

plt.plot(x_coord, y_coord)
plt.show()
```

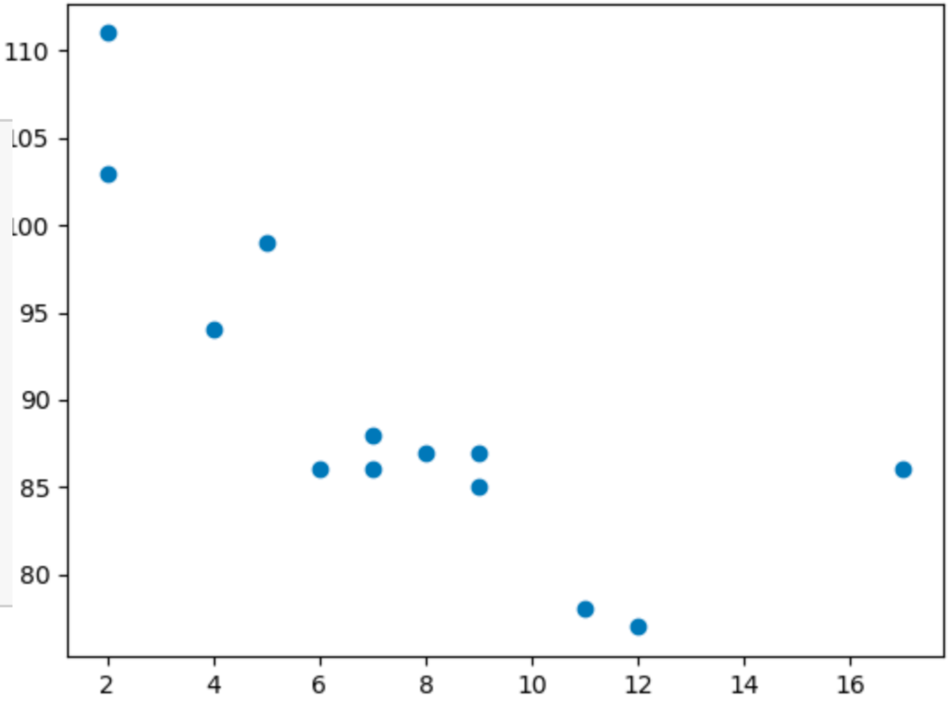


Matplotlib

- Scatter
  - plot one dot for each point

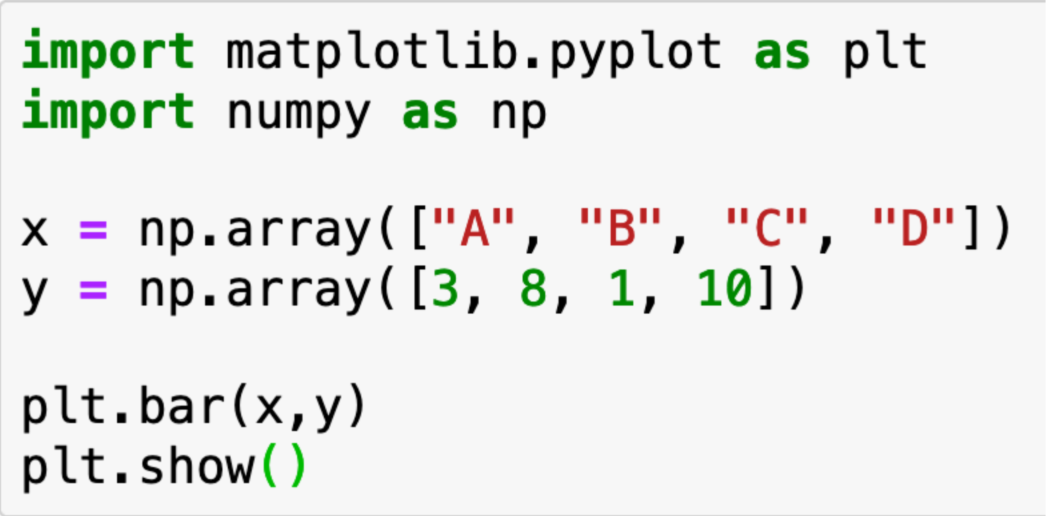


```
import matplotlib.pyplot as plt
import numpy as np

x = np.array([5,7,8,7,2,17,2,9,4,11,12,9,6])
y = np.array([99,86,87,88,111,86,103,87,94,78,77,85,86])

plt.scatter(x, y)
plt.show()
```

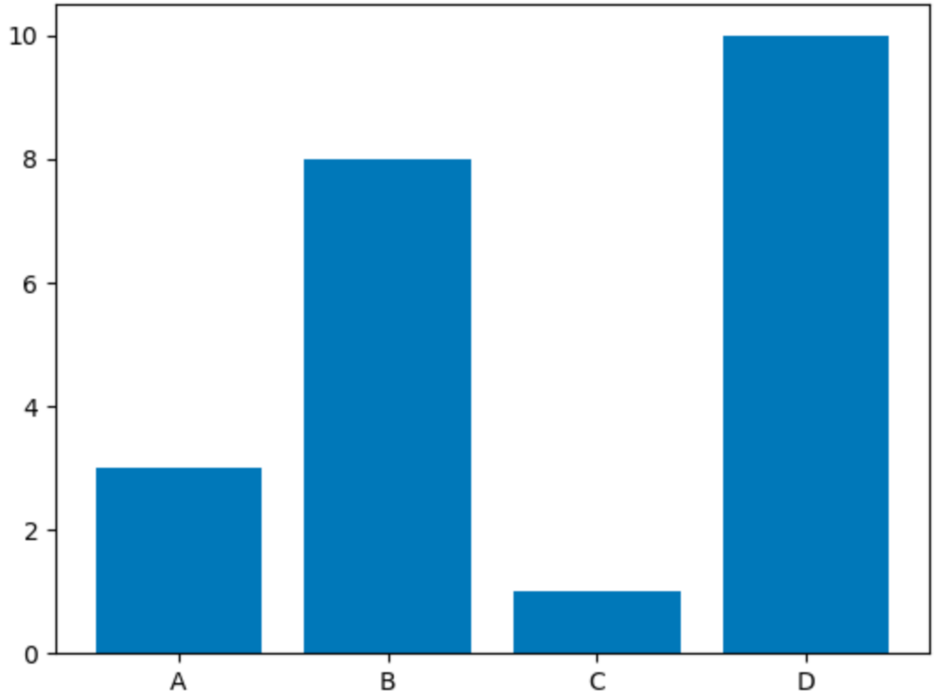


Matplotlib

- Bar plot
  - draw bar graph

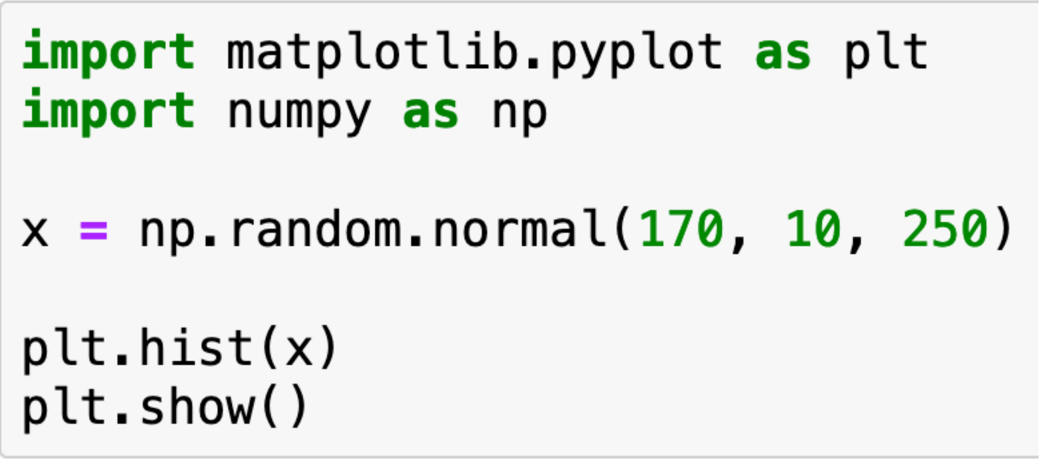


```
import matplotlib.pyplot as plt
import numpy as np

x = np.array(["A", "B", "C", "D"])
y = np.array([3, 8, 1, 10])

plt.bar(x,y)
plt.show()
```

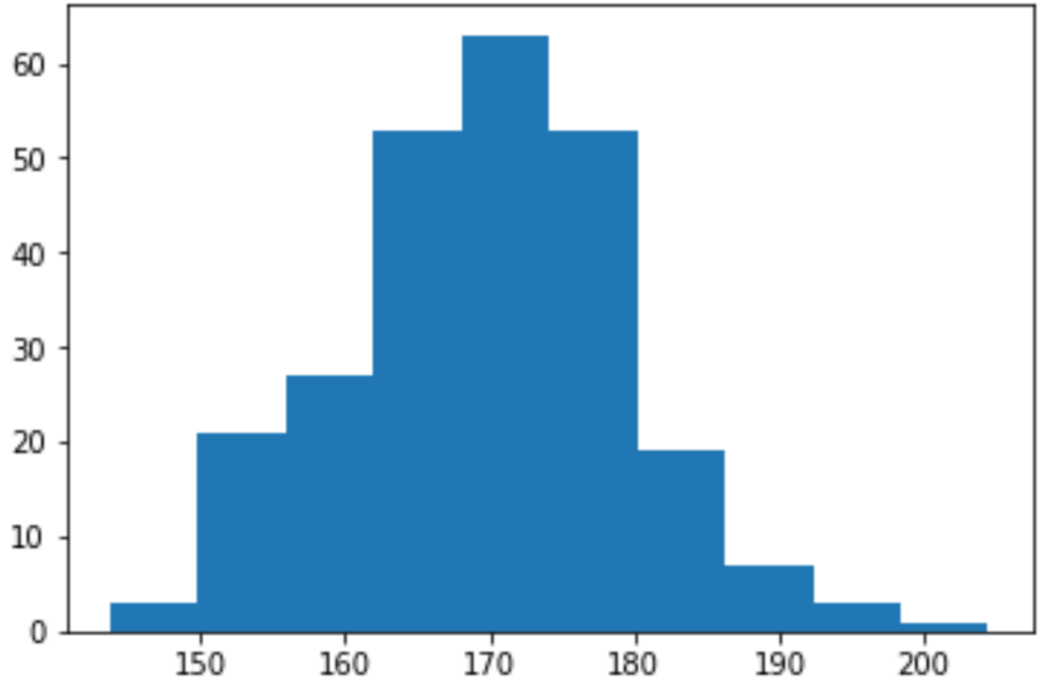


Matplotlib

- Histogram plot
  - the number of observations within each given interval

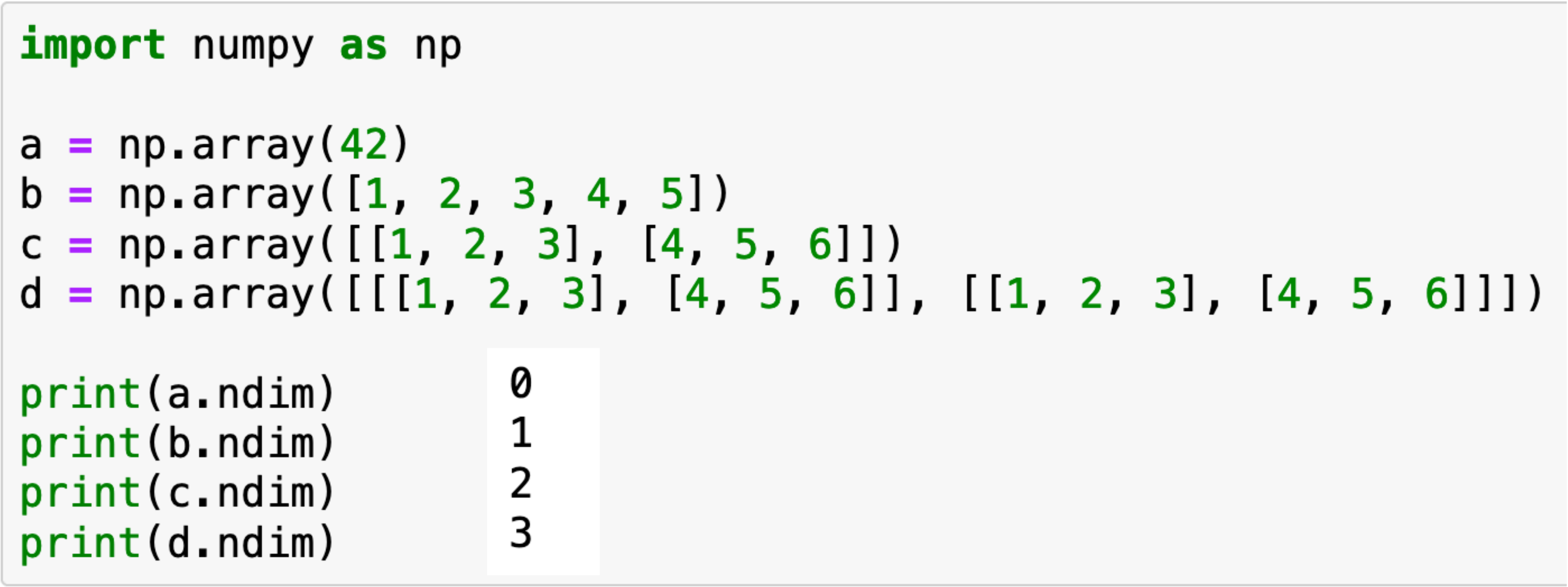


```
import matplotlib.pyplot as plt
import numpy as np

x = np.random.normal(170, 10, 250)

plt.hist(x)
plt.show()
```



NumPy

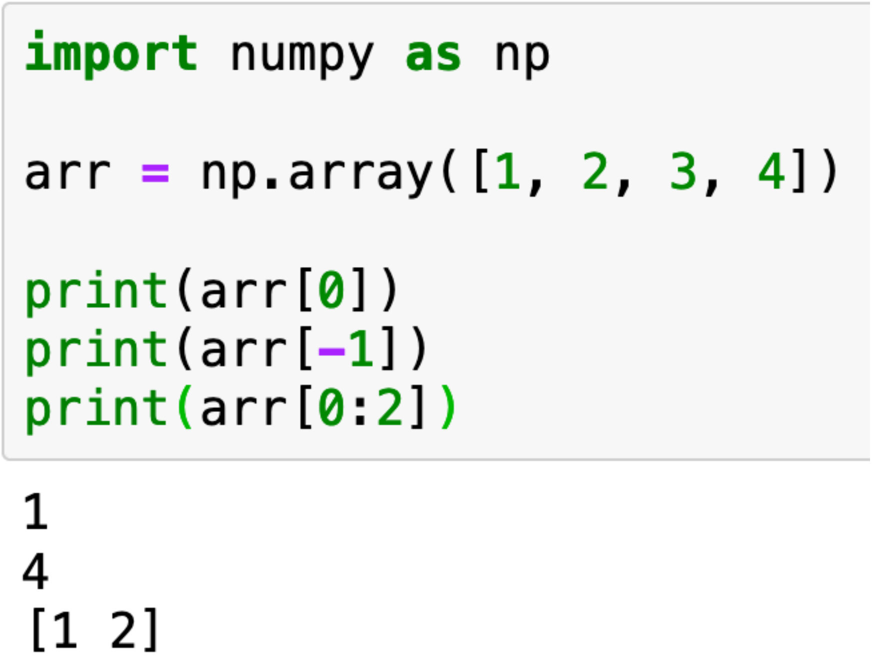
- Create a NumPy nd-array Object
  - pass a list, tuple or any array-like object into the array() method



```
import numpy as np

a = np.array(42)
b = np.array([1, 2, 3, 4, 5])
c = np.array([[1, 2, 3], [4, 5, 6]])
d = np.array([[[1, 2, 3], [4, 5, 6]], [[1, 2, 3], [4, 5, 6]]])

print(a.ndim)       0
print(b.ndim)       1
print(c.ndim)       2
print(d.ndim)       3
```

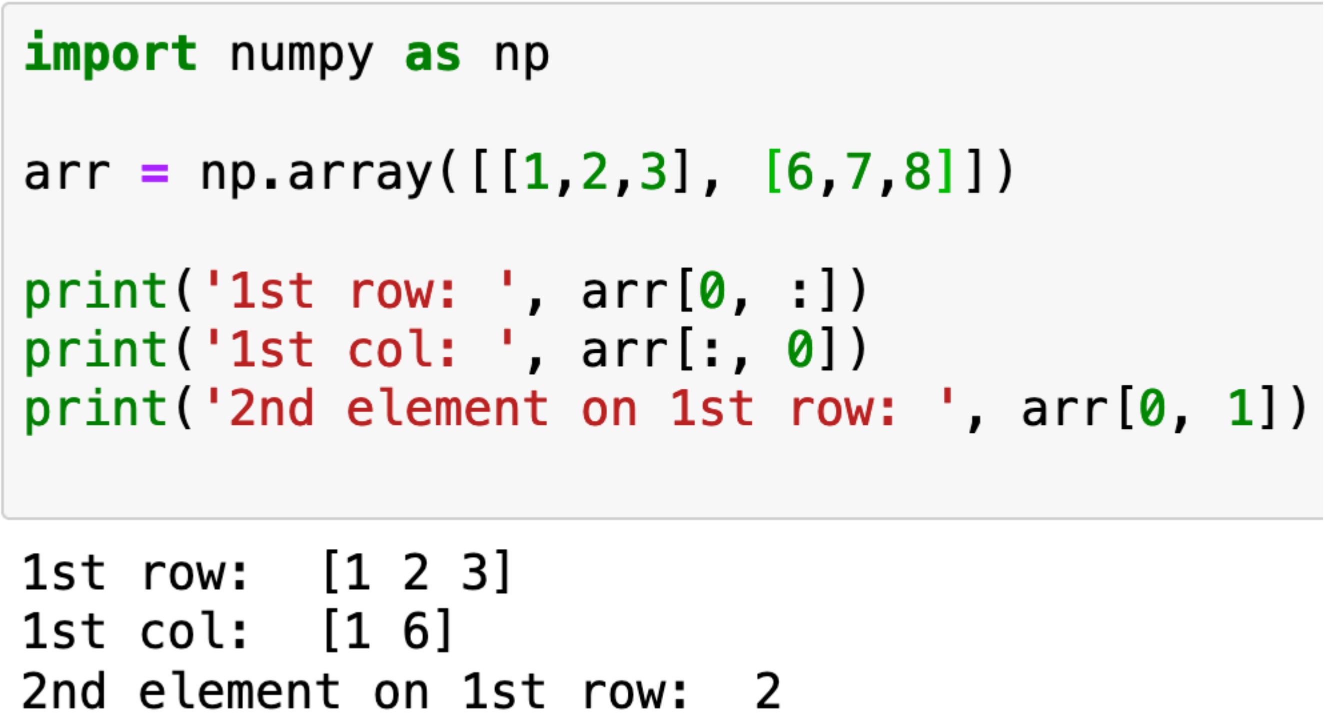
NumPy

- Access array elements



```python
import numpy as np

arr = np.array([1, 2, 3, 4])

print(arr[0])
print(arr[-1])
print(arr[0:2])
```

```
1
4
[1 2]
```

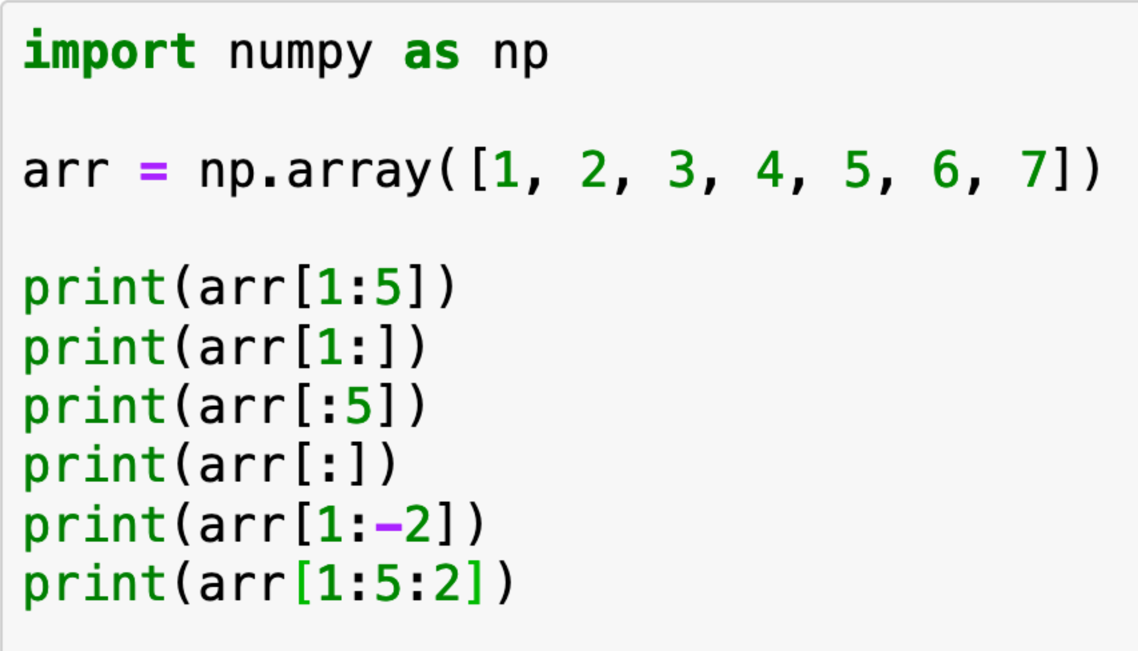


```python
import numpy as np

arr = np.array([[1,2,3], [6,7,8]])

print('1st row: ', arr[0, :])
print('1st col: ', arr[:, 0])
print('2nd element on 1st row: ', arr[0, 1])
```

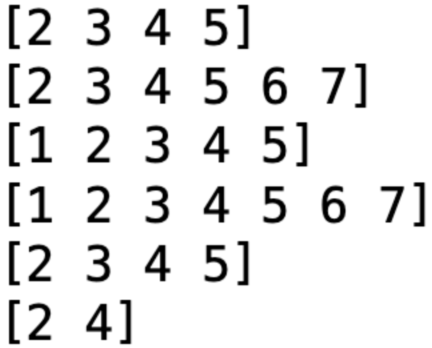
```
1st row:  [1 2 3]
1st col:  [1 6]
2nd element on 1st row:  2
```

NumPy

- Array slicing:
  - taking elements from one given index to another given index



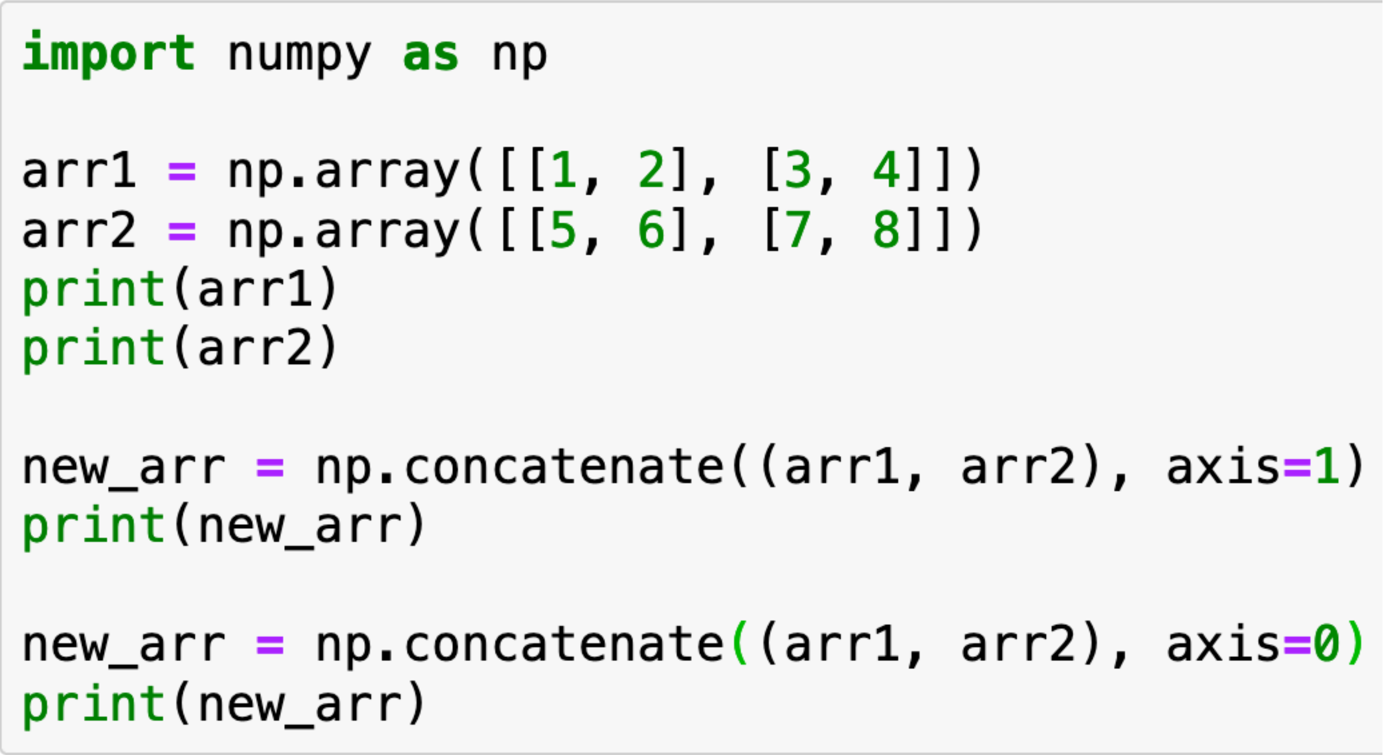
```python
import numpy as np

arr = np.array([1, 2, 3, 4, 5, 6, 7])

print(arr[1:5])
print(arr[1:])
print(arr[:5])
print(arr[:])
print(arr[1:-2])
print(arr[1:5:2])
```



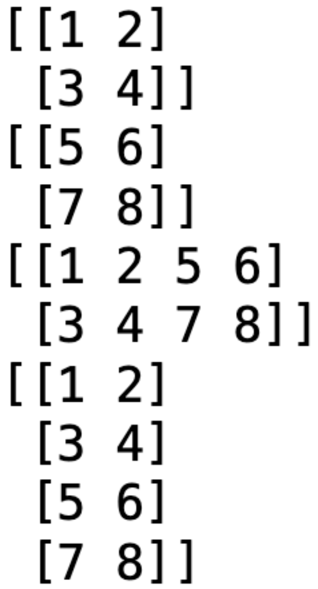
```
[2 3 4 5]
[2 3 4 5 6 7]
[1 2 3 4 5]
[1 2 3 4 5 6 7]
[2 3 4 5]
[2 4]
```

NumPy

- Joining NumPy Arrays
  - putting contents of two or more arrays in a single array



```python
import numpy as np

arr1 = np.array([[1, 2], [3, 4]])
arr2 = np.array([[5, 6], [7, 8]])
print(arr1)
print(arr2)

new_arr = np.concatenate((arr1, arr2), axis=1)
print(new_arr)

new_arr = np.concatenate((arr1, arr2), axis=0)
print(new_arr)
```



```
[[1 2]
 [3 4]]
[[5 6]
 [7 8]]
[[1 2 5 6]
 [3 4 7 8]]
[[1 2]
 [3 4]
 [5 6]
 [7 8]]
```