Introduction

The file preamble.py imports all the necessary libraries that we need for our programs

```
In [1]: from preamble import *
        %matplotlib inline
```

Why Python?

Python has become the lingua franca for many data science applications. It combines the power of general-purpose programming languages with the ease of use of domain-specific scripting languages like MATLAB or R. Python has libraries for data loading, visualization, statistics, natural language processing, image processing, and more. One of the main advantages of using Python is the ability to interact directly with the code, using a terminal or other tools like the Jupyter Notebook.

scikit-learn

Scikit-learn contains a number of state-of-the-art machine learning algorithms, as well as comprehensive documentation about each algorithm. `scikit-learn` is a very popular tool, and the most prominent Python library for machine learning. It is widely used in industry and academia, and a wealth of tutorials and code snippets are available online.

Installing scikit-learn

`scikit-learn` depends on two other Python packages, `numpy` and `scipy`. For plotting and interactive development, you should also install matplotlib, IPython, and the Jupyter Notebook.

Anaconda

A Python distribution made for large-scale data processing, predictive analytics, and scientific computing. Anaconda comes with NumPy, SciPy, matplotlib, pandas, IPython, Jupyter Notebook, and scikit-learn. Available on Mac OS, Windows, and Linux, it is a very convenient solution and is the one we suggest for people without an existing installation of the scientific Python packages.

After installing Anaconda run

`conda install numpy scipy scikit-learn matplotlib pandas pillow graphviz python-graphviz`

to install the necessary libraries in your system.

Essential Libraries and Tools

Jupyter Notebook

The Jupyter Notebook is an interactive environment for running code in the browser. It is a great tool for exploratory data analysis and is widely used by data scientists.

NumPy

NumPy is one of the fundamental packages for scientific computing in Python. It contains functionality for multidimensional arrays, high-level mathematical functions such as linear algebra operations and the Fourier transform, and pseudorandom number generators.

In [2]:
```
import numpy as np

x = np.array([[1, 2, 3], [4, 5, 6]])
print("x:\n{}".format(x))
```

```
x:
[[1 2 3]
 [4 5 6]]
```

SciPy

SciPy is a collection of functions for scientific computing in Python. It provides, among other functionality, advanced linear algebra routines, mathematical function optimization, signal processing, special mathematical functions, and statistical distributions. scikit-learn draws from SciPy's collection of functions for implementing its algorithms. The most important part of SciPy for us is scipy.sparse: this provides sparse matrices, which are another representation that is used for data in scikit-learn. Sparse matrices are used whenever we want to store a 2D array that contains mostly zeros:

In [4]:
```
from scipy import sparse

# Create a 2D NumPy array with a diagonal of ones, and zeros everywhere else
eye = np.eye(4)
print("NumPy array:\n", eye)
```

```
NumPy array:
 [[1. 0. 0. 0.]
 [0. 1. 0. 0.]
 [0. 0. 1. 0.]
 [0. 0. 0. 1.]]
```

In [5]:
```
# Convert the NumPy array to a SciPy sparse matrix in CSR format
# Only the nonzero entries are stored
sparse_matrix = sparse.csr_matrix(eye)
print("\nSciPy sparse CSR matrix:\n", sparse_matrix)
```

```
SciPy sparse CSR matrix:
 <Compressed Sparse Row sparse matrix of dtype 'float64'
	with 4 stored elements and shape (4, 4)>
  Coords	Values
  (0, 0)	1.0
  (1, 1)	1.0
  (2, 2)	1.0
  (3, 3)	1.0
```

Usually it is not possible to create dense representations of sparse data (as they would not fit into memory), so we need to create sparse representations directly. Here is a way to create the same sparse matrix as before, using the COO format:

In [6]:
```
data = np.ones(4)
row_indices = np.arange(4)
col_indices = np.arange(4)
eye_coo = sparse.coo_matrix((data, (row_indices, col_indices)))
print("COO representation:\n", eye_coo)
```

```
COO representation:
 <COOrdinate sparse matrix of dtype 'float64'
	with 4 stored elements and shape (4, 4)>
  Coords	Values
  (0, 0)	1.0
  (1, 1)	1.0
  (2, 2)	1.0
  (3, 3)	1.0
```

matplotlib

matplotlib is the primary scientific plotting library in Python. It provides functions for making publication-quality visualizations such as line charts, histograms, scatter plots, and so on.

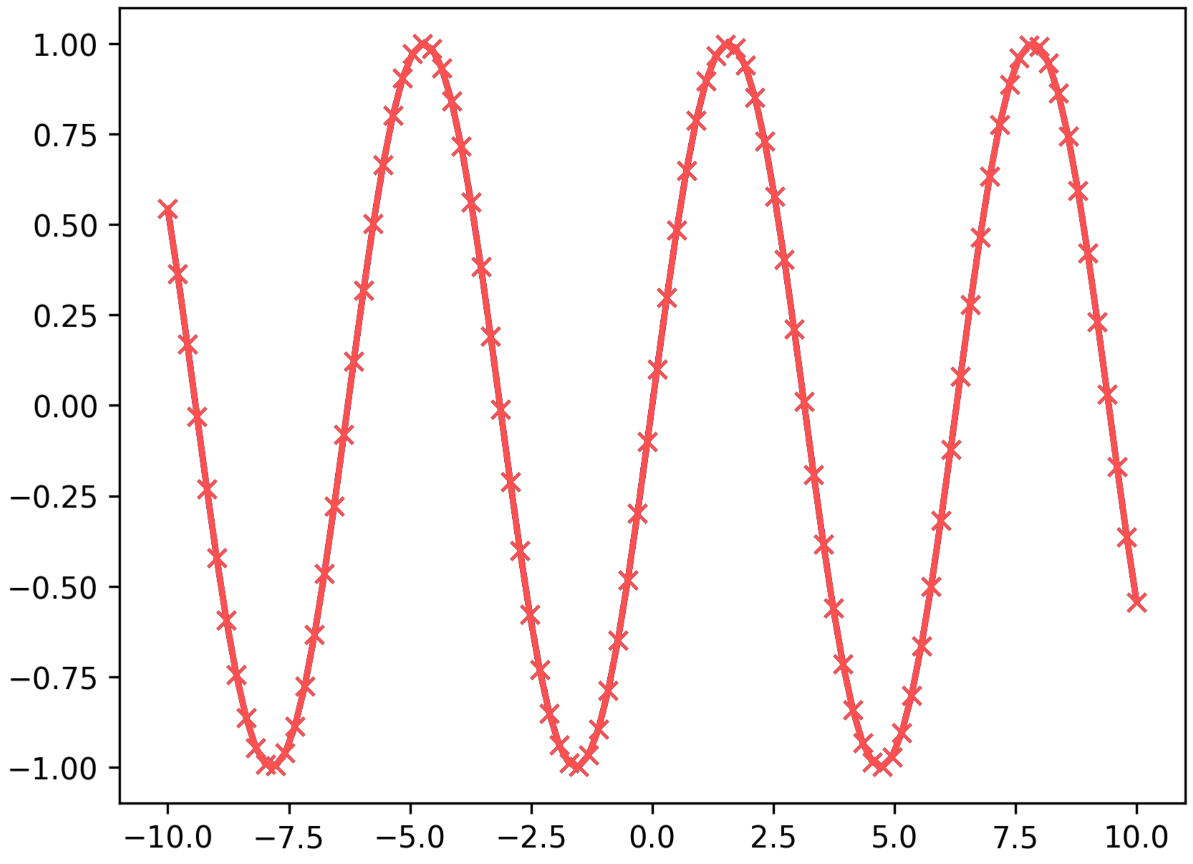
When working inside the Jupyter Notebook, you can show figures directly in the browser by using the %matplotlib inline command.

In [16]:
```
%matplotlib inline
import matplotlib.pyplot as plt

# Generate a sequence of numbers from -10 to 10 with 100 steps in between
x = np.linspace(-10, 10, 100)
print(x)
# Create a second array using sine
y = np.sin(x)
print(y)
# The plot function makes a line chart of one array against another
```

```
plt.plot(x, y, marker="x")
plt.show()
```

```
[-10.     -9.798  -9.596  -9.394  -9.192  -8.99   -8.788  -8.586  -8.384
  -8.182  -7.98   -7.778  -7.576  -7.374  -7.172  -6.97   -6.768  -6.566
  -6.364  -6.162  -5.96   -5.758  -5.556  -5.354  -5.152  -4.949  -4.747
  -4.545  -4.343  -4.141  -3.939  -3.737  -3.535  -3.333  -3.131  -2.929
  -2.727  -2.525  -2.323  -2.121  -1.919  -1.717  -1.515  -1.313  -1.111
  -0.909  -0.707  -0.505  -0.303  -0.101   0.101   0.303   0.505   0.707
   0.909   1.111   1.313   1.515   1.717   1.919   2.121   2.323   2.525
   2.727   2.929   3.131   3.333   3.535   3.737   3.939   4.141   4.343
   4.545   4.747   4.949   5.152   5.354   5.556   5.758   5.96    6.162
   6.364   6.566   6.768   6.97    7.172   7.374   7.576   7.778   7.98
   8.182   8.384   8.586   8.788   8.99    9.192   9.394   9.596   9.798
  10.   ]
[ 0.544  0.365  0.17  -0.031 -0.231 -0.421 -0.595 -0.744 -0.863 -0.947
 -0.992 -0.997 -0.962 -0.887 -0.776 -0.634 -0.466 -0.279 -0.08   0.121
  0.318  0.502  0.665  0.801  0.905  0.972  0.999  0.986  0.933  0.841
  0.716  0.561  0.384  0.191 -0.01  -0.211 -0.403 -0.578 -0.73  -0.852
 -0.94  -0.989 -0.998 -0.967 -0.896 -0.789 -0.65  -0.484 -0.298 -0.101
  0.101  0.298  0.484  0.65   0.789  0.896  0.967  0.998  0.989  0.94
  0.852  0.73   0.578  0.403  0.211  0.01  -0.191 -0.384 -0.561 -0.716
 -0.841 -0.933 -0.986 -0.999 -0.972 -0.905 -0.801 -0.665 -0.502 -0.318
 -0.121  0.08   0.279  0.466  0.634  0.776  0.887  0.962  0.997  0.992
  0.947  0.863  0.744  0.595  0.421  0.231  0.031 -0.17  -0.365 -0.544]
```



pandas

`pandas` is a Python library for data wrangling and analysis. It is built around a data structure called the `DataFrame` that is modeled after the R DataFrame. Simply put, a pandas DataFrame is a table, similar to an Excel spreadsheet. pandas provides a great range of methods to modify and operate on this table; in particular, it allows SQL-like queries and joins of tables.

In [17]:

```
import pandas as pd

# create a simple dataset of people
data = {'Name': ["John", "Anna", "Peter", "Linda"],
        'Location' : ["New York", "Paris", "Berlin", "London"],
        'Age' : [24, 13, 53, 33]
       }

data_pandas = pd.DataFrame(data)
# IPython.display allows "pretty printing" of dataframes
# in the Jupyter notebook
display(data_pandas)
```

	Name	Location	Age
0	John	New York	24
1	Anna	Paris	13
2	Peter	Berlin	53
3	Linda	London	33

In [18]:

```
# Select all rows that have an age column greater than 30
display(data_pandas[data_pandas.Age > 30])
```

	Name	Location	Age
2	Peter	Berlin	53
3	Linda	London	33

mglearn

This is a library of utility functions we wrote, so that we don't clutter up our code listings with details of plotting and data loading.

Python 2 versus Python 3

There are two major versions of Python that are widely used at the moment: Python 2 (more precisely, 2.7) and Python 3 (with the latest release being 3.9 at the time of writing). This sometimes leads to some confusion. Python 2 is no longer actively developed, but because Python 3 contains major changes, Python 2 code usually does not run on Python 3. If you are new to Python, or are starting a new project from scratch, we highly recommend using the latest version of Python 3.

Versions Used in this Book

In [19]:
```
import sys
print("Python version:", sys.version)

import pandas as pd
print("pandas version:", pd.__version__)

import matplotlib
print("matplotlib version:", matplotlib.__version__)

import numpy as np
print("NumPy version:", np.__version__)

import scipy as sp
print("SciPy version:", sp.__version__)

import IPython
print("IPython version:", IPython.__version__)

import sklearn
print("scikit-learn version:", sklearn.__version__)
```

```
Python version: 3.12.7 | packaged by Anaconda, Inc. | (main, Oct  4 2024, 0
8:22:19) [Clang 14.0.6 ]
pandas version: 2.2.3
matplotlib version: 3.10.0
NumPy version: 2.1.3
SciPy version: 1.15.0
IPython version: 8.27.0
scikit-learn version: 1.6.0
```

In [ ]: