**In[1]:**

```
from sklearn.base import BaseEstimator, TransformerMixin

class MyTransformer(BaseEstimator, TransformerMixin):
    def __init__(self, first_parameter=1, second_parameter=2):
        # All parameters must be specified in the __init__ function
        self.first_parameter = 1
        self.second_parameter = 2

    def fit(self, X, y=None):
        # fit should only take X and y as parameters
        # Even if your model is unsupervised, you need to accept a y argument!

        # Model fitting code goes here
        print("fitting the model right here")
        # fit returns self
        return self

    def transform(self, X):
        # transform takes as parameter only X

        # Apply some transformation to X
        X_transformed = X + 1
        return X_transformed
```

Implementing a classifier or regressor works similarly, only instead of `TransformerMixin` you need to inherit from `ClassifierMixin` or `RegressorMixin`. Also, instead of implementing `transform`, you would implement `predict`.

As you can see from the example given here, implementing your own estimator requires very little code, and most `scikit-learn` users build up a collection of custom models over time.

# Where to Go from Here

This book provides an introduction to machine learning and will make you an effective practitioner. However, if you want to further your machine learning skills, here are some suggestions of books and more specialized resources to investigate to dive deeper.

## Theory

In this book, we tried to provide an intuition of how the most common machine learning algorithms work, without requiring a strong foundation in mathematics or computer science. However, many of the models we discussed use principles from probability theory, linear algebra, and optimization. While it is not necessary to understand all the details of how these algorithms are implemented, we think that

knowing some of the theory behind the algorithms will make you a better data scientist. There have been many good books written about the theory of machine learning, and if we were able to excite you about the possibilities that machine learning opens up, we suggest you pick up at least one of them and dig deeper. We already mentioned Hastie, Tibshirani, and Friedman's book *The Elements of Statistical Learning* in the Preface, but it is worth repeating this recommendation here. Another quite accessible book, with accompanying Python code, is *Machine Learning: An Algorithmic Perspective* by Stephen Marsland (Chapman and Hall/CRC). Two other highly recommended classics are *Pattern Recognition and Machine Learning* by Christopher Bishop (Springer), a book that emphasizes a probabilistic framework, and *Machine Learning: A Probabilistic Perspective* by Kevin Murphy (MIT Press), a comprehensive (read: 1,000+ pages) dissertation on machine learning methods featuring in-depth discussions of state-of-the-art approaches, far beyond what we could cover in this book.

## Other Machine Learning Frameworks and Packages

While `scikit-learn` is our favorite package for machine learning[1] and Python is our favorite language for machine learning, there are many other options out there. Depending on your needs, Python and `scikit-learn` might not be the best fit for your particular situation. Often using Python is great for trying out and evaluating models, but larger web services and applications are more commonly written in Java or C++, and integrating into these systems might be necessary for your model to be deployed. Another reason you might want to look beyond `scikit-learn` is if you are more interested in statistical modeling and inference than prediction. In this case, you should consider the `statsmodel` package for Python, which implements several linear models with a more statistically minded interface. If you are not married to Python, you might also consider using R, another lingua franca of data scientists. R is a language designed specifically for statistical analysis and is famous for its excellent visualization capabilities and the availability of many (often highly specialized) statistical modeling packages.

Another popular machine learning package is `vowpal wabbit` (often called `vw` to avoid possible tongue twisting), a highly optimized machine learning package written in C++ with a command-line interface. `vw` is particularly useful for large datasets and for streaming data. For running machine learning algorithms distributed on a cluster, one of the most popular solutions at the time of writing is `mllib`, a Scala library built on top of the `spark` distributed computing environment.

1 Andreas might not be entirely objective in this matter.