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