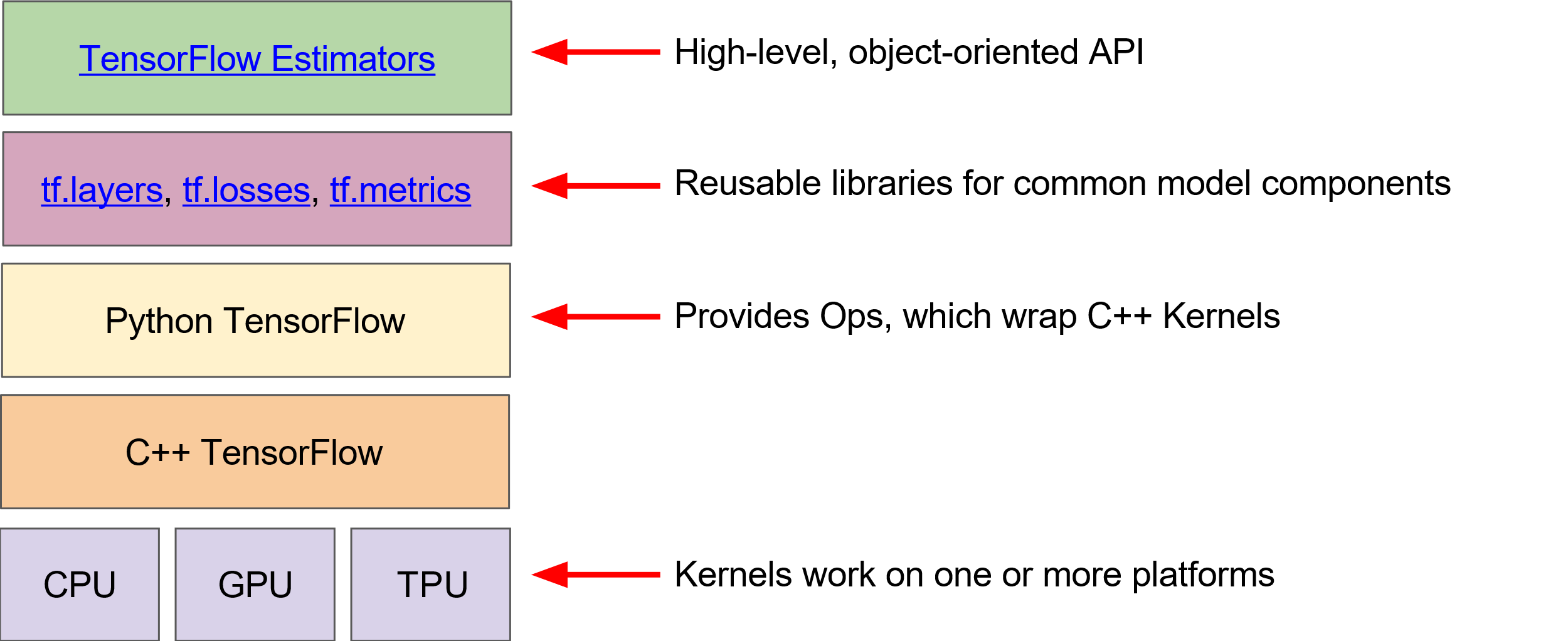
First Steps with TensorFlow: Toolkit

**Estimated Time:** 4 minutes

The following figure shows the current hierarchy of TensorFlow toolkits:



**Figure 1. TensorFlow toolkit hierarchy.**

The following table summarizes the purposes of the different layers:

| Toolkit(s) | Description |
| --- | --- |
| Estimator (tf.estimator) | High-level, OOP API. |
| tf.layers/tf.losses/tf.metrics | Libraries for common model components. |
| TensorFlow | Lower-level APIs |

TensorFlow consists of the following two components:

* a [graph protocol buffer](https://www.tensorflow.org/extend/tool_developers/#protocol_buffers)
* a runtime that executes the (distributed) graph

These two components are analogous to the Java compiler and the JVM. Just as the JVM is implemented on multiple hardware platforms, so is TensorFlow—CPUs and GPUs.

Which API(s) should you use? You should use the highest level of abstraction that solves the problem. The higher levels of abstraction are easier to use, but are also (by design) less flexible. We recommend you start with the highest-level API first and get everything working. If you need additional flexibility for some special modeling concerns, move one level lower. Note that each level is built using the APIs in lower levels, so dropping down the hierarchy should be reasonably straightforward.

tf.estimator API

We'll use tf.estimator for the majority of exercises in Machine Learning Crash Course. Everything you'll do in the exercises could have been done in lower-level (raw) TensorFlow, but using tf.estimator dramatically lowers the number of lines of code.

tf.estimator is compatible with the scikit-learn API. [Scikit-learn](http://scikit-learn.org/) is an extremely popular open-source ML library in Python, with over 100k users, including many at Google.

Very broadly speaking, here's the format of a linear regression program implemented in tf.estimator:

import tensorflow as tf  
  
# Set up a linear classifier.  
classifier = tf.estimator.LinearClassifier()  
  
# Train the model on some example data.  
classifier.train(input\_fn=train\_input\_fn, steps=2000)  
  
# Use it to predict.  
predictions = classifier.predict(input\_fn=predict\_input\_fn)

First Steps with TensorFlow: Programming Exercises

**Estimated Time:** 55 minutes

As you progress through Machine Learning Crash Course, you'll put the principles and techniques you learn into practice by coding models using tf.estimator, a high-level [TensorFlow](https://www.tensorflow.org/) API.

The programming exercises in Machine Learning Crash Course use a data-analysis platform that combines code, output, and descriptive text into one collaborative document.

Programming exercises run directly in your browser (no setup required!) using the [Colaboratory](https://colab.research.google.com/) platform. Colaboratory is supported on most major browsers, and is most thoroughly tested on desktop versions of Chrome and Firefox. If you'd prefer to download and run the exercises offline, see [these instructions](https://github.com/google/eng-edu/blob/master/ml/cc/README.md#with-docker) for setting up a local environment.

Run the following three exercises in the provided order:

1. [Quick Introduction to pandas](https://colab.research.google.com/notebooks/mlcc/intro_to_pandas.ipynb?hl=en). pandas is an important library for data analysis and modeling, and is widely used in TensorFlow coding. This tutorial provides all the pandas information you need for this course. If you already know pandas, you can skip this exercise.
2. [First Steps with TensorFlow](https://colab.research.google.com/notebooks/mlcc/first_steps_with_tensor_flow.ipynb?hl=en). This exercise explores linear regression.
3. [Synthetic Features and Outliers](https://colab.research.google.com/notebooks/mlcc/synthetic_features_and_outliers.ipynb?hl=en). This exercise explores synthetic features and the effect of input outliers.

Common hyperparameters in Machine Learning Crash Course exercises

Many of the coding exercises contain the following hyperparameters:

* **steps**, which is the total number of training iterations. One step calculates the loss from *one batch* and uses that value to modify the model's weights *once*.
* **batch size**, which is the number of examples (chosen at random) for a single step. For example, the batch size for SGD is 1.

The following formula applies:



A convenience variable in Machine Learning Crash Course exercises

The following convenience variable appears in several exercises:

* **periods**, which controls the granularity of reporting. For example, if periods is set to 7 and steps is set to 70, then the exercise will output the loss value every 10 steps (or 7 times). Unlike hyperparameters, we don't expect you to modify the value of periods. Note that modifying periods does not alter what your model learns.

The following formula applies:

