While Python is an excellent environment for building computationally-intensive scientific applications and building most kinds of general purpose systems, there are a number of uses for which Python may be less suitable. As Python is an interpreted programming language, in general most Python code will run substantially slower than code written in a compiled language like Java or C++. As programmer time is typically more valuable than CPU time, many are happy to make this tradeoff. However, in an application with very low latency requirements (for example, a high frequency trading system), the time spent programming in a lower-level, lower-productivity language like C++ to achieve the maximum possible performance might be time well spent. Python is not an ideal language for highly concurrent, multithreaded applications, particularly applications with many CPU-bound threads. The reason for this is that it has what is known as the global interpreter lock (GIL), a mechanism which prevents the interpreter from executing more than one Python bytecode instruction at a time. The technical reasons for why the GIL exists are beyond the scope of this book, but as of this writing it does not seem likely that the GIL will disappear anytime soon. While it is true that in many big data processing applications, a cluster of computers may be required to process a data set in a reasonable amount of time, there are still situations where a single-process, multithreaded system is desirable. This is not to say that Python cannot execute truly multithreaded, parallel code; that code just cannot be executed in a single Python process. As an example, the Cython