6.1.2 "Compiled" Python files

As an important speed-up of the start-up time for short programs that use a lot of standard modules, if a file called 'spam.pyc' exists in the directory where 'spam.py' is found, this is assumed to contain an already-"byte-compiled" version of the module 'spam'. The modification time of the version of 'spam.py' used to create 'spam.pyc' is recorded in 'spam.pyc', and the '.pyc' file is ignored if these don't match.

Normally, you don't need to do anything to create the'spam.pyc' file. Whenever 'spam.py' is successfully compiled, an attempt is made to write the compiled version to'spam.pyc'. It is not an error if this attempt fails; if for any reason the file is not written completely, the resulting'spam.pyc' file will be recognized as invalid and thus ignored later. The contents of the 'spam.pyc' file are platform independent, so a Python module directory can be shared by machines of different architectures.

Some tips for experts:

- When the Python interpreter is invoked with the -O flag, optimized code is generated and stored in '.pyo' files. The optimizer currently doesn't help much; it only removes assert statements. When -O is used, *all*bytecode is optimized; .pyc files are ignored and .py files are compiled to optimized bytecode.
- Passing two -O flags to the Python interpreter (-OO) will cause the bytecode compiler to perform optimizations that could in some rare cases result in malfunctioning programs. Currently only __doc__ strings are removed from the bytecode, resulting in more compact '.pyo'files. Since some programs may rely on having these available, you should only use this option if you know what you're doing.
- A program doesn't run any faster when it is read from a '.pyc' or '.pyo' file than when it is read from a '.py' file; the only thing that's faster about '.pyc' or '.pyo' files is the speed with which they are loaded.
- When a script is run by giving its name on the command line, the bytecode for the script is never written to a '.pyc' or '.pyo' file. Thus, the startup time of a script may be reduced by moving most of its code to a module and having a small bootstrap script that imports that module. It is also possible to name a '.pyc' or '.pyo' file directly on the command line.
- The module 'compileall'{} can create '.pyc' files (or'.pyo' files when -O is used) for all modules in a directory.