32.10. py_compile — Compile Python source files

Source code: Lib/py_compile.py

The py_compile module provides a function to generate a byte-code file from a source file, and another function used when the module source file is invoked as a script.

Though not often needed, this function can be useful when installing modules for shared use, especially if some of the users may not have permission to write the byte-code cache files in the directory containing the source code.

exception py_compile.**PyCompileError**

Exception raised when an error occurs while attempting to compile the file.

py_compile.**compile**(*file*, *cfile=None*, *dfile=None*, *doraise=False*, *optimize=-1*)

Compile a source file to byte-code and write out the byte-code cache file. The source code is loaded from the file named *file*. The byte-code is written to *cfile*, which defaults to the **PEP 3147**/**PEP 488** path, ending in .pyc. For example, if *file* is /foo/bar/baz.py *cfile* will default to /foo/bar/__pycache__/baz.cpython-32.pyc for Python 3.2. If *dfile* is specified, it is used as the name of the source file in error messages when instead of *file*. If *doraise* is true, a PyCompileError is raised when an error is encountered while compiling *file*. If *doraise* is false (the default), an error string is written to sys.stderr, but no exception is raised. This function returns the path to byte-compiled file, i.e. whatever *cfile* value was used.

If the path that *cfile* becomes (either explicitly specified or computed) is a symlink or non-regular file, FileExistsError will be raised. This is to act as a warning that import will turn those paths into regular files if it is allowed to write byte-compiled files to those paths. This is a side-effect of import using file renaming to place the final byte-compiled file into place to prevent concurrent file writing issues.

optimize controls the optimization level and is passed to the built-in compile() function. The default of -1 selects the optimization level of the current interpreter.

Changed in version 3.2: Changed default value of *cfile* to be **PEP 3147**-compliant. Previous default was *file* + 'c' ('o' if optimization was enabled). Also added the *optimize* parameter.

Changed in version 3.4: Changed code to use `importlib` for the byte-code cache file writing. This means file creation/writing semantics now match what `importlib` does, e.g. permissions, write-and-move semantics, etc. Also added the caveat that `FileExistsError` is raised if *cfile* is a symlink or non-regular file.

`py_compile.`**`main`**(*args=None*)

Compile several source files. The files named in *args* (or on the command line, if *args* is `None`) are compiled and the resulting byte-code is cached in the normal manner. This function does not search a directory structure to locate source files; it only compiles files named explicitly. If `'-'` is the only parameter in args, the list of files is taken from standard input.

Changed in version 3.2: Added support for `'-'`.

When this module is run as a script, the `main()` is used to compile all the files named on the command line. The exit status is nonzero if one of the files could not be compiled.

See also:

Module `compileall`

Utilities to compile all Python source files in a directory tree.