The following scripts are written to demonstrate file handling and management modules in Python. The library modules explored include;

LZMA: compression using the LZMA algorithm. This module provides classes and convenience functions for compressing and decompressing data using the LZMA compression algorithm. Also included is a file interface supporting the .xz and legacy .lzma file formats used by the xz utility, as well as raw compressed streams. The interface provided by this module is very similar to that of the bz2 module. However, note that LZMAFile is not thread-safe, unlike bz2.BZ2File, so if you need to use a single LZMAFile instance from multiple threads, it is necessary to protect it with a lock.

GZIP: Support for gzip files. This module provides a simple interface to compress and decompress files just like the GNU programs gzip and gunzip would. The data compression is provided by the zlib module. The gzip module provides the GzipFile class, as well as the open(), compress() and decompress() convenience functions. The GzipFile class reads and writes gzip-format files, automatically compressing or decompressing the data so that it looks like an ordinary file object.

Tarfile: read and write tar archive files. The tarfile module makes it possible to read and write tar archives, including those using gzip, bz2 and lzma compression. Use the zipfile module to read or write .zip files, or the higher-level functions in shutil.

Plistlib: generate and parse Mac OS X .plist files. This module provides an interface for reading and writing the “property list” files used mainly by Mac OS X and supports both binary and XML plist files. The property list (.plist) file format is a simple serialization supporting basic object types, like dictionaries, lists, numbers and strings. Usually the top level object is a dictionary. To write out and to parse a plist file, use the dump() and load() functions. To work with plist data in bytes objects, use dumps() and loads(). Values can be strings, integers, floats, booleans, tuples, lists, dictionaries (but only with string keys), Data, bytes, bytesarray or datetime.datetime objects.

Fnmatch: unix filename pattern matching. This module provides support for Unix shell-style wildcards, which are not the same as regular expressions.

Glob: unix style pathname pattern expansion. The glob module finds all the pathnames matching a specified pattern according to the rules used by the Unix shell, although results are returned in arbitrary order. No tilde expansion is done, but \*, ?, and character ranges expressed with [] will be correctly matched. This is done by using the os.scandir() and fnmatch.fnmatch() functions in concert, and not by actually invoking a subshell.

Pathlib: object-oriented filesystem paths. This module offers classes representing filesystem paths with semantics appropriate for different operating systems. Path classes are divided between 'pure paths', which provide purely computational operations without I/O, and 'concrete paths', which inherit from pure paths but also provide I/O operations.

Fileinput: iterate over lines from multiple input streams. This module implements a helper class and functions to quickly write a loop over standard input or a list of files.

Filecmp: file and Directory Comparisons. The filecmp module defines functions to compare files and directories, with various optional time/correctness trade-offs.

Tempfile: generate temporary files and directories. This module creates temporary files and directories. It works on all supported platforms. TemporaryFile, NamedTemporaryFile, TemporaryDirectory, and SpooledTemporaryFile are high-level interfaces which provide automatic cleanup and can be used as context managers. mkstemp() and mkdtemp() are lower-level functions which require manual cleanup.

Compiled and presented by Vakindu Philliam.