Python object serialization

Python object serialization and deserialization is an important aspect of any non-trivial program. If in Python you save something to a file, if you read a configuration file, or if you respond to an HTTP request, you do object serialization and deserialization.

The [**pickle**](https://docs.python.org/2/library/pickle.html#module-pickle) module implements a fundamental, but powerful algorithm for serializing and de-serializing a Python object structure. “Pickling” is the process whereby a Python object hierarchy is converted into a byte stream, and “unpickling” is the inverse operation, whereby a byte stream is converted back into an object hierarchy. Pickling (and unpickling) is alternatively known as “serialization”, “marshalling,” or “flattening”, however, to avoid confusion, the terms used here are “pickling” and “unpickling”.

Python has a more primitive serialization module called [marshal](https://docs.python.org/2/library/marshal.html#module-marshal), but in general [pickle](https://docs.python.org/2/library/pickle.html#module-pickle) should always be the preferred way to serialize Python objects. [marshal](https://docs.python.org/2/library/marshal.html#module-marshal) exists primarily to support Python’s  ‘.pyc ‘ files.

The [pickle](https://docs.python.org/3/library/pickle.html#module-pickle) module differs from [marshal](https://docs.python.org/3/library/marshal.html#module-marshal) in several significant ways:

* The [pickle](https://docs.python.org/3/library/pickle.html#module-pickle) module keeps track of the objects it has already serialized, so that later references to the same object won’t be serialized again. [marshal](https://docs.python.org/3/library/marshal.html#module-marshal) doesn’t do this.

This has implications both for recursive objects and object sharing. Recursive objects are objects that contain references to themselves. These are not handled by marshal, and in fact, attempting to marshal recursive objects will crash your Python interpreter. Object sharing happens when there are multiple references to the same object in different places in the object hierarchy being serialized. [pickle](https://docs.python.org/3/library/pickle.html#module-pickle) stores such objects only once, and ensures that all other references point to the master copy. Shared objects remain shared, which can be very important for mutable objects.

* [marshal](https://docs.python.org/3/library/marshal.html#module-marshal) cannot be used to serialize user-defined classes and their instances. [pickle](https://docs.python.org/3/library/pickle.html#module-pickle) can save and restore class instances transparently, however the class definition must be importable and live in the same module as when the object was stored.
* The [marshal](https://docs.python.org/3/library/marshal.html#module-marshal) serialization format is not guaranteed to be portable across Python versions. Because its primary job in life is to support.pyc files, the Python implementers reserve the right to change the serialization format in non-backwards compatible ways should the need arise. The [pickle](https://docs.python.org/3/library/pickle.html#module-pickle) serialization format is guaranteed to be backwards compatible across Python releases.

**Example:**

import pickle

>>> with open('entry.pickle', 'wb') as f:

...     pickle.dump(entry, f)

* This is still in Python Shell #1.
* Use the open() function to open a file. Set the file mode to 'wb' to open the file for writing [in binary mode](http://www.diveintopython3.net/files.html#binary). Wrap it in a [with statement](http://www.diveintopython3.net/files.html#with) to ensure the file is closed automatically when you’re done with it.
* The dump() function in the pickle module takes a serializable Python data structure, serializes it into a binary, Python-specific format using the latest version of the pickle protocol, and saves it to an open file.

The pickle module takes a Python data structure and saves it to a file.

To do this, it serializes the data structure using a data format called “the pickle protocol.”

The pickle protocol is Python-specific; there is no guarantee of cross-language compatibility. You probably couldn’t take the entry.pickle file you just created and do anything useful with it in Perl, php, Java, or any other language.

Not every Python data structure can be serialized by the pickle module. The pickle protocol has changed several times as new data types have been added to the Python language, but there are still limitations.

As a result of these changes, there is no guarantee of compatibility between different versions of Python itself. Newer versions of Python support the older serialization formats, but older versions of Python do not support newer formats (since they don’t support the newer data types).

Unless you specify otherwise, the functions in the pickle module will use the latest version of the pickle protocol. This ensures that you have maximum flexibility in the types of data you can serialize, but it also means that the resulting file will not be readable by older versions of Python that do not support the latest version of the pickle protocol.

The latest version of the pickle protocol is a binary format. Be sure to open your pickle files [in binary mode](http://www.diveintopython3.net/files.html#binary), or the data will get corrupted during writing.