Stable Application Binary Interface

Traditionally, the C API of Python will change with every release. Most changes will be source-compatible, typically by only adding API, rather than changing existing API or removing API (although some interfaces do get removed after being deprecated first).

Unfortunately, the API compatibility does not extend to binary compatibility (the ABI). The reason is primarily the evolution of struct definitions, where addition of a new field, or changing the type of a field, might not break the API, but can break the ABI. As a consequence, extension modules need to be recompiled for every Python release (although an exception is possible on Unix when none of the affected interfaces are used). In addition, on Windows, extension modules link with a specific pythonXY.dll and need to be recompiled to link with a newer one.

Since Python 3.2, a subset of the API has been declared to guarantee a stable ABI. Extension modules wishing to use this API (called “limited API”) need to define `Py_LIMITED_API`. A number of interpreter details then become hidden from the extension module; in return, a module is built that works on any 3.x version (x>=2) without recompilation.

In some cases, the stable ABI needs to be extended with new functions. Extension modules wishing to use these new APIs need to set `Py_LIMITED_API` to the `PY_VERSION_HEX` value (see API and ABI Versioning) of the minimum Python version they want to support (e.g. `0x03030000` for Python 3.3). Such modules will work on all subsequent Python releases, but fail to load (because of missing symbols) on the older releases.

As of Python 3.2, the set of functions available to the limited API is documented in **PEP 384**. In the C API documentation, API elements that are not part of the limited API are marked as “Not part of the limited API.”