Status of stable API and stable ABI in Python 3.8

Victor Stinner

Include subdirectories



- Include/: public stable API
- Include/cpython/: unstable CPython specific API
- Include/internal/: internal API, used by CPython
- Reduce risk of adding an API into the public stable API by mistake



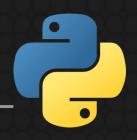
static inline



- "#define MACRO(...)" is errorprone
- static inline PyObject*
 PyObject_INIT(PyObject *op, PyTypeObject *typeobj)
- result and parameter types
- Better variable scope, avoid conflict
- No "do { ... } while (0)" hack



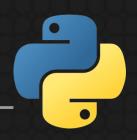
Internal API



- Now installed by make install
- C extension must opt-in: define
 Py_BUILD_CORE_MODULE
- Many "_Py" functions made internal, ex: _PyBuiltin_Init()
- API surrounded by "#ifdef
 Py_BUILD_CORE" moved to internal,
 ex: _PyGC_Fini()



Debug build



- Py_TRACE_REFS disabled in debug build: debug and release are ABI compatible
- C extensions no longer linked to libpython on Unix, but Android
- Debug build looks for release libraries
 - .cpython-38d-x86_64-linux-gnu.so
 - .cpython-38-x86_64-linux-gnu.so (new!)
 - .abi3.so (new!)
- Debug runs more checks at runtime python

Getting rid of global



- Eric Snow's PEP 554: Multiple Interpreters in the Stdlib
- Load 2 C module instance
- Release memory at exit.
- Any shared resource means locking: bad for parallelism. One "GIL" per interpreter.
- See Petr Viktorin's talk :-)



TODO



- Guidelines to prevent mistakes when adding new APIs
- Big issue: borrowed references, ex: Py_TYPE(obj)
- "Py_REFCNT(obj) = 1": Add Py_SET_REFCNT(obj, refnct)?
- Static PyTypeObject: big blocker issue for stable ABI. PyType_FromSpec()?
- Hide more implementation details

