Python C API

Victor Stinner

My C API long term goals

- More **freedom** to change Python **internals**
- Reduce 3rd party **maintenance burden** when updating Python (goal: no change)
- Reduce **friction/stress** related to the C API, on the Python side, and on the 3rd party side

My C API long term goals

Limited C API by default!
Stable ABI for all!

It's a long term goal,
not the exact purpose of this talk

Challenges

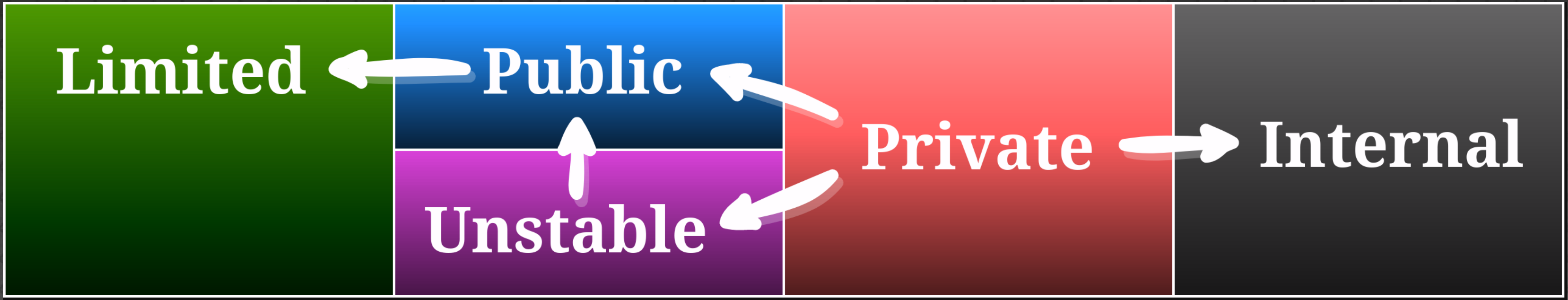
- **Need to change** some 3rd party code to reach this long term goal
- Usually **1-10 lines** per Python release in the few **impacted projects**
- **Unknown number** of impacted projects per change

Present (3.13)

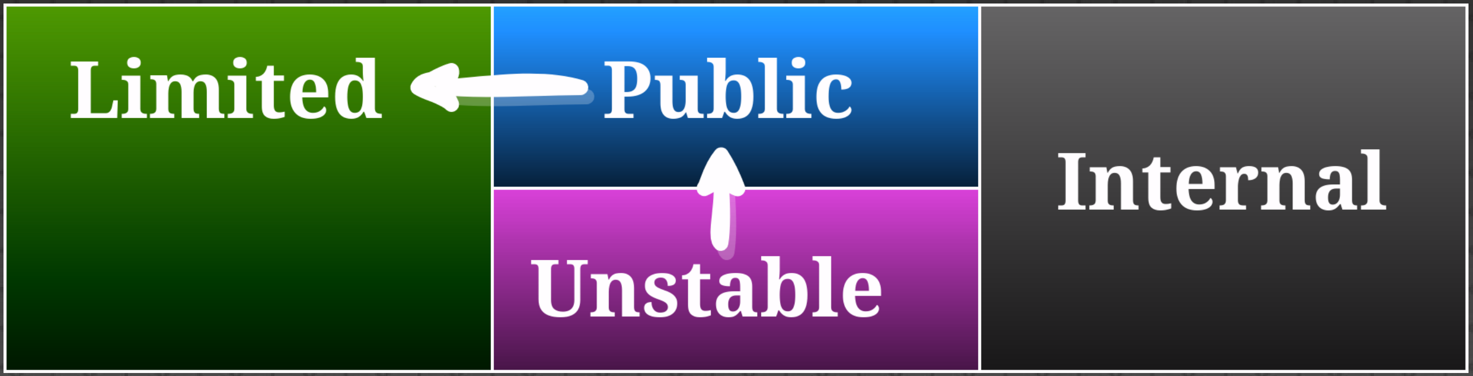
Remove private funcs

- Removed 300+ private functions in Python 3.13 alpha1
- Some removals impacted 5-30 projects
- As planned, removals causing most trouble have been reverted (mostly in alpha2)
- 3.13 beta1: 264 functions removed

Remove private funcs



Remove private funcs



Better public functions

- Promote private to **public** functions
- Add **backward compatibility warranty**
- Add **documentation** and **tests**
- C API Working Group designs a **better** API
- Enforce error checking
- Avoid inefficient `PyErr_Occurred()` check

PyDict_Pop() example

- 3.12: `PyObject* _PyDict_Pop(PyObject *dict, PyObject *key, PyObject *deflt)`
  → NULL means "not found" or error

- 3.13: `int PyDict_Pop(PyObject *op, PyObject *key, PyObject **result)`

- → Return -1 on error and set an exception
  → Return 0 if not found (NULL result)
  → Return 1 if found (non-NULL result)

Argument Clinic Limited API

- Use limited C API if code uses `#define Py_LIMITED_API` …
- Only use public functions of the limited C API
- Inline code to avoid calling private/internal functions

AC: fcntl.fcntl() clinic input

```
fd: fildes
cmd as code: int
arg: object(c_default='NULL') = 0
/
```

AC: fcntl.fcntl() code in 3.12

```
if (!_PyArg_CheckPositional(
          "fcntl", nargs, 2, 3)) …
if (!_PyLong_FileDescriptor_Converter(
          args[0], &fd)) …
code = _PyLong_AsInt(args[1]);
```

AC: fcntl.fcntl() code in 3.13

```
if (nargs < 2) { PyErr_Format(…)… }
if (nargs > 3) { PyErr_Format(…)… }
fd = PyObject_AsFileDescriptor(args[0]);
if (fd < 0) …
code = PyLong_AsInt(args[1]);
```

Build stdlib ext with limited

- Build 16 stdlib extensions with the limited C API
- errno, md5, resource, _uuid, ...
- Test our own limited C API
- PEP 737 "Unify type formatting" adds %T and %N formats for error messages

And Python <= 3.12 ?

pythoncapi-compat provides new Python 3.13 functions on Python 3.12 and older

- `PyDict_GetItemRef()`
- `PyLong_AsInt()`
- `PyUnicode_EqualToUTF8()`
- …

Stable ABI fix

- Py_None no longer reads the private symbol: `&_Py_NoneStruct`
- Limited C API 3.13 calls the public function: `Py_GetConstantBorrowed(Py_CONSTANT_NONE)`
- New `Py_GetConstant(id)` API

Future (3.14)

Limited C API & stakeholders

- Cython: opt-in build mode
- PyO3 (Rust): opt-in build mode; consider making it the default!
- pybind11 (C++11): not supported
- nanobind (C++11): opt-in build mode, 100% feature complete
- cffi: not supported

abi3audit

- Test if a binary uses the stable ABI or not
- `https://github.com/trailofbits/abi3audit`

TODO

- Add API for PyFrameObject
  Cython, greenlet, eventlet, gevent, ...
- Add API for PyThreadState
  Cython, ...
- Design a migration path away from PyTypeObject members direct access