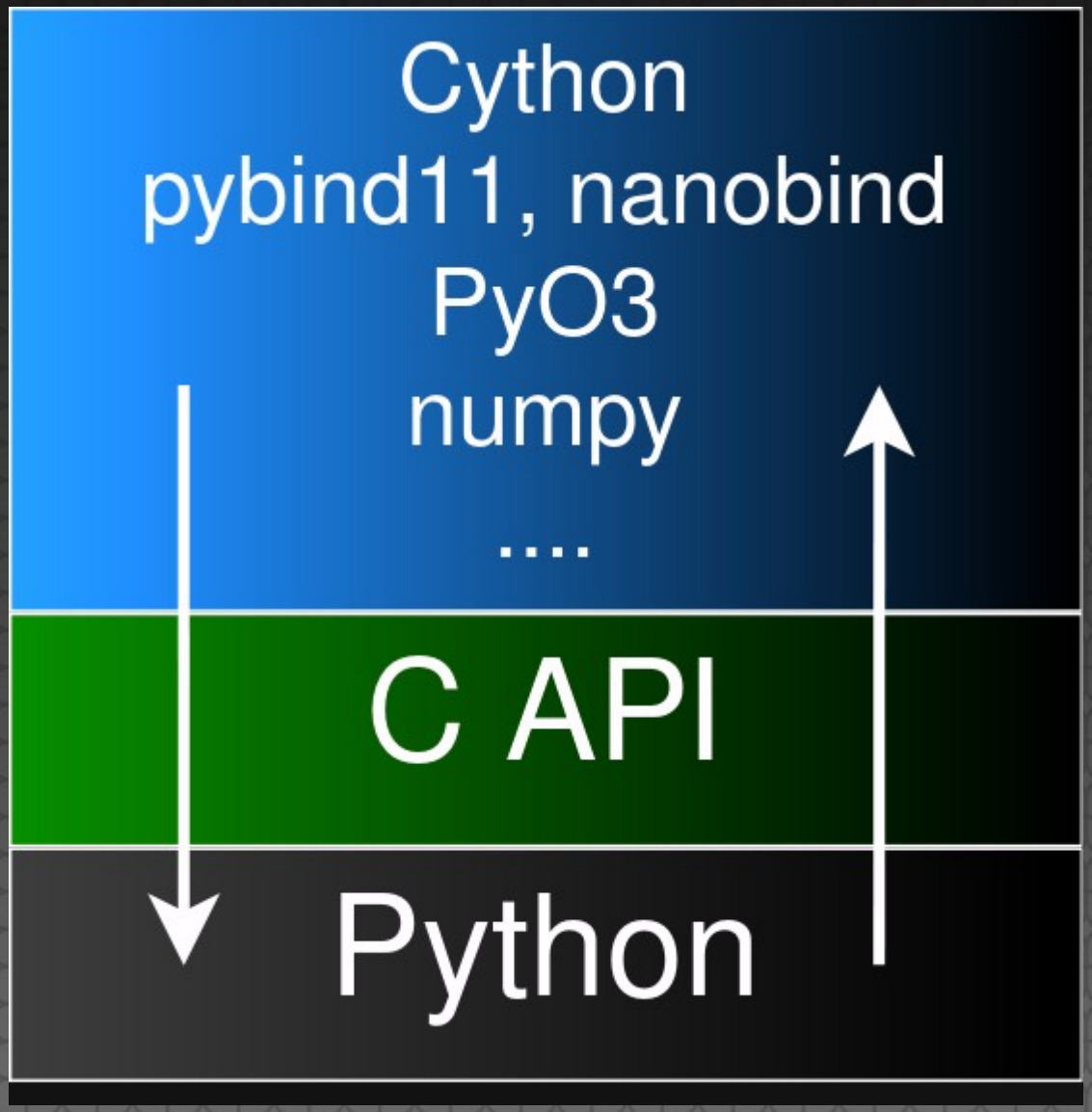

Python
C API
Victor
Stinner

Victor Stinner

- CPython **core developer** for 14 years
- Maintain **Python CI** (GitHub Actions, buildbots), fix regressions, memory leaks, race conditions
- Work for **Red Hat**: backport fixes up to Python 3.6 and fix security vulnerabilities
- Happy **vim** and **Fedora** user!

C API



Red Hat

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

Stable ABI

- **Build once**, work on all Python versions
- Less wheel packages to distribute (one per "platform")
- Platform: {OS, CPU, libc}
- Added to **Python 3.2** (2011)
- Used by **500**+ C extensions on PyPI: PySide6, cryptography, ...

Red Hat

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

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


Python 3.13

C API Working Group

- PEP 731 "C API Working Group Charter"
- Created in November 2023
- New C API must be approved by the WG
- 6 members: Erlend Aasland, Michael Droettboom, Petr Viktorin, Serhiy Storchaka, Steve Dower, Victor Stinner
- PEP 733 "An Evaluation of Python's Public C API"

Red Hat

C API WG projects

- **decisions**: decide on an API
- **api-evolution**: list small API changes
- **api-revolution**: list disruptive API changes

https://github.com/capi-workgroup/

Red Hat

Remove private funcs

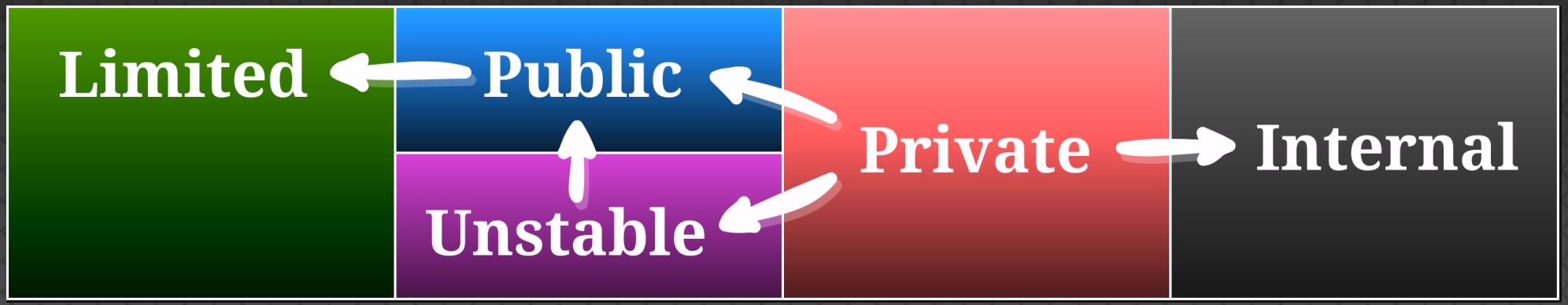
- No doc, no test
- No backward compatibility support
- Private functions are not part of the limited C API
- Replace them with public functions

Red Hat

Remove private funcs

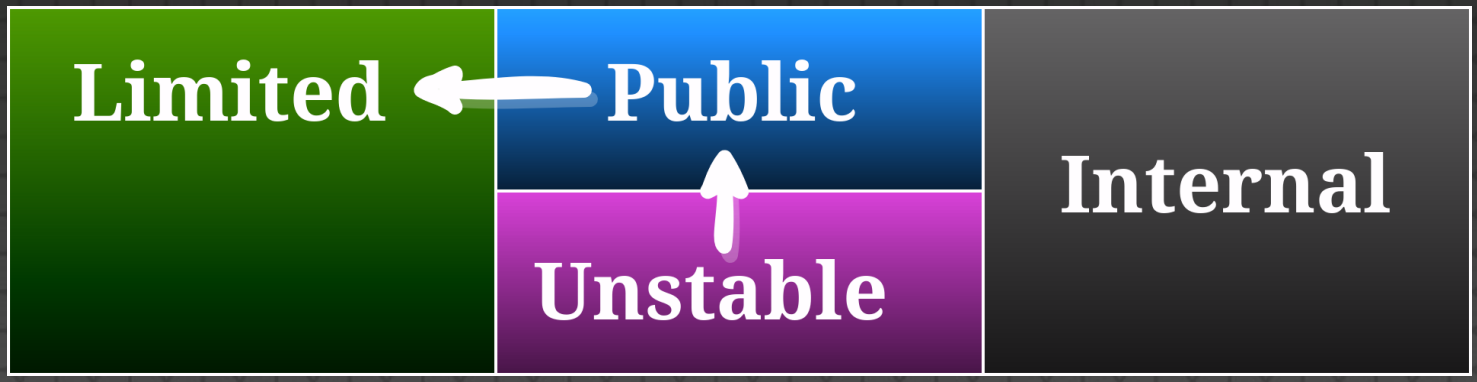
- Removed 300+ private functions in Python 3.13 alpha1
- Some removals impacted 5-30 projects
- As planned, removals causing most trouble have been reverted (mostly in alpha2)
- 3.13 beta1: 264 functions removed

Remove private funcs



Red Hat

Remove private funcs



Red Hat

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

Red Hat

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)

And Python <= 3.12 ?

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

- `PyDict_GetItemRef()`
- `PyLong_AsInt()`
- `PyUnicode_EqualToUTF8()`
- ...

Red Hat


Python 3.13
Limited C API

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

Red Hat

AC: fcntl.fcntl() clinic input

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

Red Hat

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]);
```

Red Hat

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

- Limited C API was not really used/tested by Python itself
- Test the limited C API
- Make sure that it works with non trivial code
- "Eating your own dog food"

Build stdlib ext with limited

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


Future
(3.14)

HPy

- New C API which looks like the Python C API
- More efficient on PyPy
- "Universal mode" which is a stable ABI working any CPython version and any PyPy version
- numpy is being porting to HPy

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

Use the Limited C API

- CPython: define `Py_LIMITED_API` macro
- nanobind: define `Py_LIMITED_API` macro
- Cython: define `CYTHON_LIMITED_API` macro

  The macro must be set to a Python "hex" version.
  Example: `0x03080000` for Python 3.8.

Red Hat

abi3audit

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

Red Hat


Q & A