API <=> ABI



- C API macro:
 #define PyList_GET_ITEM(op, I)
 ((PyListObject *)op)→ob_item[i]
- Machine code: (PyObject **)(((char*)list) + OFFSET)[i]
- Release build: OFFSET=24
- Debug build: OFFSET=40





PyPy existing solution



- Change C macro for function call: #define PyList_GET_ITEM(op, I) PyList_GetItem(op, i)
- Machine code:CALL PyList_GetItem()
 - → Function call: ABI compatibility
- API remains compatible.





What is a bad API?



- Avoid PyObject** in the API:
 PyObject** PySequence_Fast_ITEMS(ob)
- Avoid borrowed references:

```
PyDict_GetItem()
```

PySys_GetObject()

PyTuple_GetItem()





Roadmap

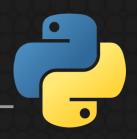


- Identify problematic API
- Change macros: use function calls
- Opt-in API without impl. details: hide PyObject.ob_refcnt
- Optional: Split Include/ into subdirectories
- Remove problematic functions, default API remains unchanged





Stable ABI

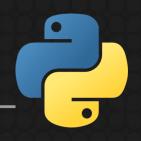


- Compile your C extension once with Python 3.8
- Binary works on 3.8, 3.9, etc.
- Binary works on release and debug builds
- Binary works on default API with implementation details





Long-term roadmap



- Maybe change the default API to no implementation detail
- Need to measure the perf. overhead
- Check how many projects are broken without impl. details
- Reduce the C API size
- PyPy wants to remove the finalizer
 API and PEP 393 (new uicode) API





Unknown future



- Tagged pointers?
- Specialized lists for small integers?
 PyObject** → int32_t*
- Move reference counters to a different memory block? Avoid "Copy-on-Read" after fork issue
- Switch to tracing garbage collector?



