

Dr. nanobind_json or: How I Learned to Stop Worrying and Fix an Unmaintained Package

Gracjan Adamus @ Pykonik 84

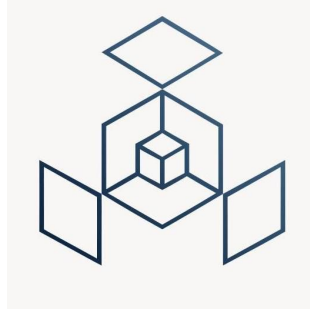
#whoami

- Third-year Applied Computer Science student @ AGH



#whoami

- Third-year Applied Computer Science student @ AGH
- President of KNI Kernel



#whoami

- Third-year Applied Computer Science student @ AGH
- President of KNI Kernel
- C++ Developer @ digatus



#whoami

- Third-year Applied Computer Science student @ AGH
- President of KNI Kernel
- C++ Developer @ digatus
- 2025 Summer Intern @ Paul Scherrer Institute



#whoami

- Third-year Applied Computer Science student @ AGH
- President of KNI Kernel
- C++ Developer @ digatus
- 2025 Summer Intern @ Paul Scherrer Institute
- Certified Mole lover



#whoami

- Third-year Applied Computer Science student @ AGH
- President of KNI Kernel
- C++ Developer @ digatus
- 2025 Summer Intern @ Paul Scherrer Institute
- Certified Mole lover
- Interested in HPC, GPUs and all the fun stuff



#whoami

- Third-year Applied Computer Science student @ AGH
- President of KNI Kernel
- C++ Developer @ digatus
- 2025 Summer Intern @ Paul Scherrer Institute
- Certified Mole lover
- Interested in HPC, GPUs and all the fun stuff
- Griger5 @ Github



What is pybind11?

*py*bind**11**

What is pybind11?



- Created by Wenzel Jakob, associate professor at EPFL

What is pybind11?



- Created by Wenzel Jakob, associate professor at EPFL
- Header-only C++11 library

What is pybind11?



- Created by Wenzel Jakob, associate professor at EPFL
- Header-only C++11 library
- Exposes C++ to Python and vice versa

What is pybind11?



- Created by Wenzel Jakob, associate professor at EPFL
- Header-only C++11 library
- Exposes C++ to Python and vice versa
- A convenient wrapper over Python's C API (**Python.h**)

What is pybind11?

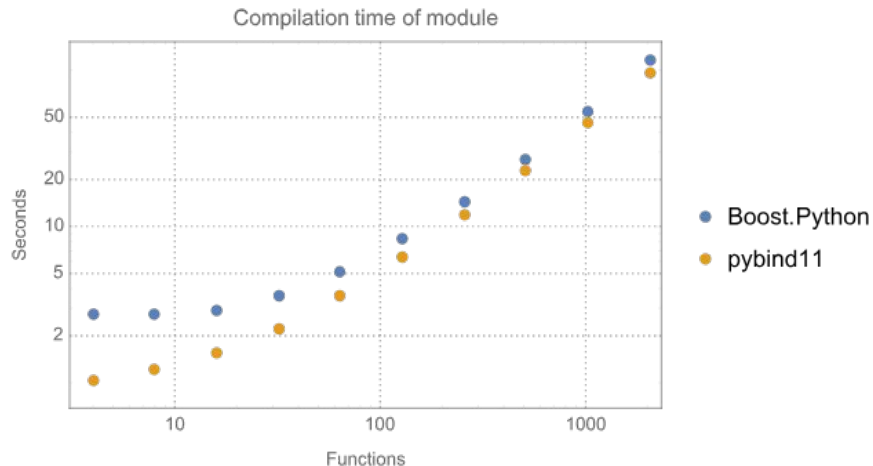
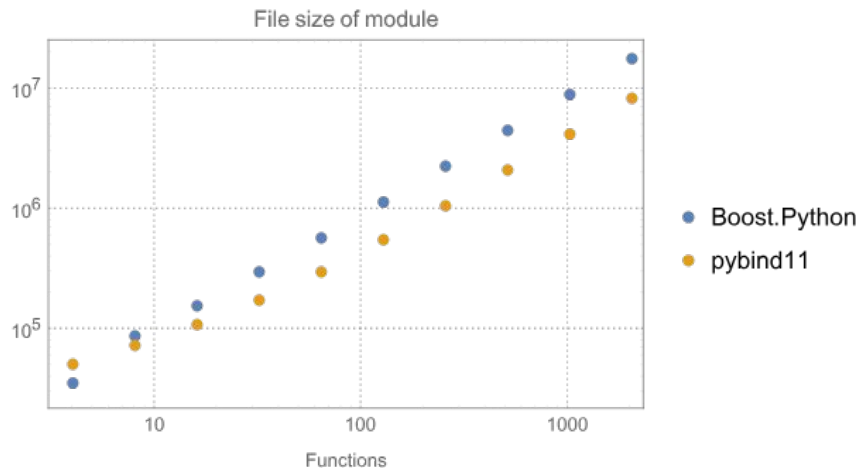


- Created by Wenzel Jakob, associate professor at EPFL
- Header-only C++11 library
- Exposes C++ to Python and vice versa
- A convenient wrapper over Python's C API (**Python.h**)
- Minimal boilerplate

What is pybind11?

*pybind***11**

- Created by Wenzel Jakob, associate professor at EPFL
- Header-only C++11 library
- Exposes C++ to Python and vice versa
- A convenient wrapper over Python's C API (**Python.h**)
- Minimal boilerplate



Does anybody even use pybind11?

Does anybody even use pybind11?



PyTorch



scikit-image
image processing in python



SciPy



OpenCV

ROS2

Code comparison: Python.h vs pybind11

Code comparison: Python.h vs pybind11

```
3  void sort(int *a, size_t size) {
4      int curr;
5      int j;
6
7      for (size_t i = 1; i < size; i++) {
8          curr = a[i];
9          j = i - 1;
10
11         while (j >= 0 && a[j] > curr) {
12             a[j+1] = a[j];
13             j = j - 1;
14         }
15
16         a[j+1] = curr;
17     }
18 }
```

Code comparison: Python.h vs pybind11

```
1 static PyObject *mod_sort(PyObject *self, PyObject *args) {
2     PyObject *array = NULL;
3
4     if (!PyArg_ParseTuple(args, "O", &array)) {
5         return NULL;
6     }
7
8     int size = PyList_Size(array);
9     int a[size];
10
11     for (int i = 0; i < size; i++) {
12         a[i] = PyLong_AsLong(PyList_GetItem(array, i))
13     }
14
15     PyObject *sorted = PyList_New(size);
16
17     sort(a, size);
18
19     for (int i = 0; i < size; i++) {
20         PyList_SetItem(sorted, i, PyLong_FromLong(a[i]));
21     }
22
23     return Py_BuildValue("O", sorted);
24 }
25
```

```
26 static PyMethodDef mod_methods[] = {
27     {"sort", (PyCFunction)mod_sort, METH_VARARGS, "Insert
28     {NULL, NULL, 0, NULL}
29 };
30
31
32 static struct PyModuleDef modmodule = {
33     PyModuleDef_HEAD_INIT,
34     "mod",
35     NULL,
36     -1,
37     mod_methods
38 };
39
40 PyMODINIT_FUNC PyInit_mod(void){
41     return PyModule_Create(&modmodule);
42 }
```

Code comparison: Python.h vs pybind11

```
5  std::vector<int> sort_vec(std::vector<int> a) {  
6      sort(a.data(), a.size());  
7      return a;  
8  }  
9  
10 PYBIND11_MODULE(sorting, m) {  
11     m.def("sort", &sort_vec, "Insertion sort");  
12 }
```

And what is nanobind?



And what is nanobind?

- Also created by Wenzel Jakob, as a successor to pybind11



And what is nanobind?

- Also created by Wenzel Jakob, as a successor to pybind11
- Fully utilizes C++17 features



And what is nanobind?

- Also created by Wenzel Jakob, as a successor to pybind11
- Fully utilizes C++17 features
- Faster compilation



And what is nanobind?

- Also created by Wenzel Jakob, as a successor to pybind11
- Fully utilizes C++17 features
- Faster compilation
- Smaller binary sizes



And what is nanobind?

- Also created by Wenzel Jakob, as a successor to pybind11
- Fully utilizes C++17 features
- Faster compilation
- Smaller binary sizes
- Lower runtime overhead



And what is nanobind?

- Also created by Wenzel Jakob, as a successor to pybind11
- Fully utilizes C++17 features
- Faster compilation
- Smaller binary sizes
- Lower runtime overhead
- Very similar API

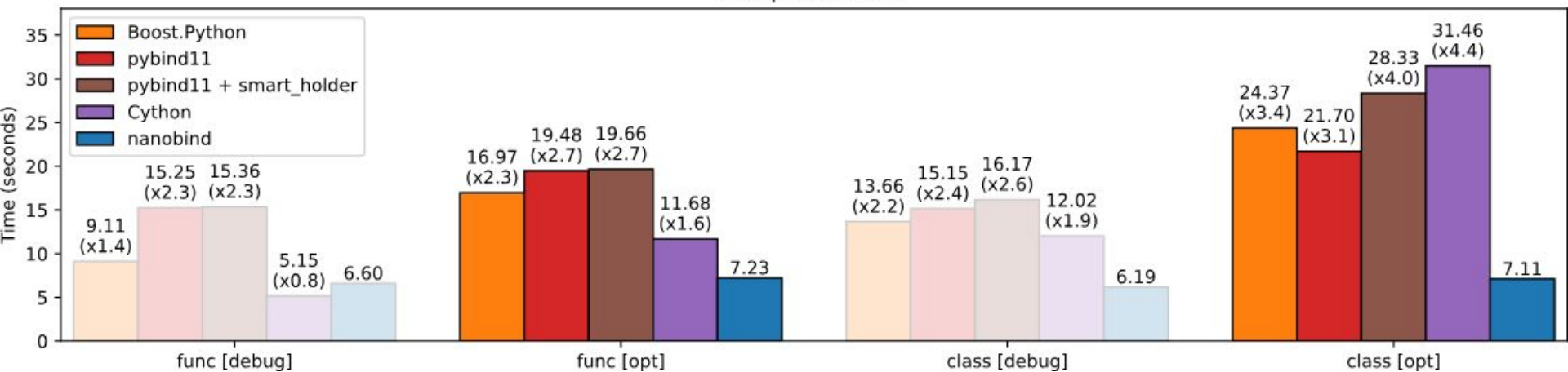


And what is nanobind?

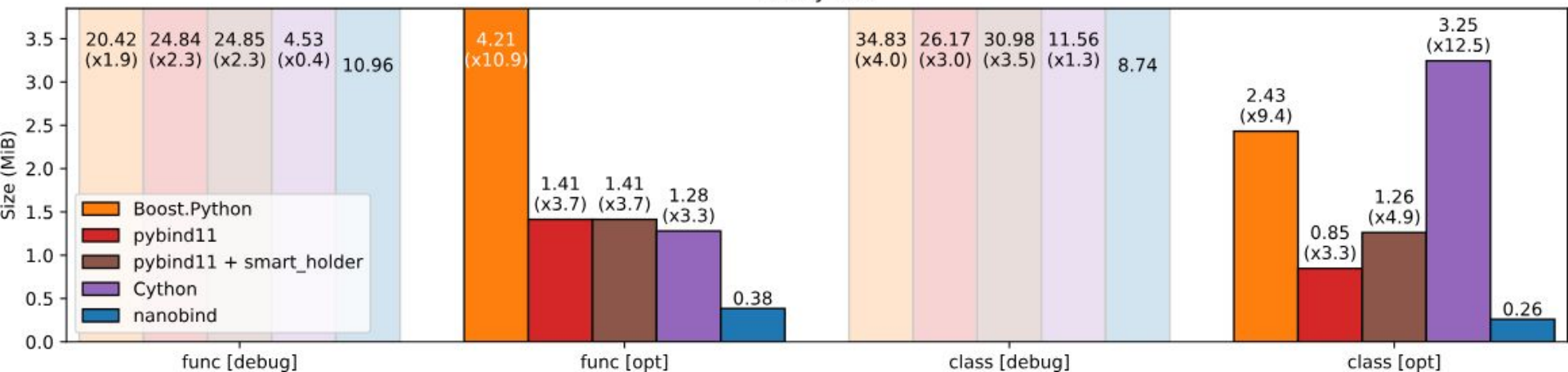
- Also created by Wenzel Jakob, as a successor to pybind11
- Fully utilizes C++17 features
- Faster compilation
- Smaller binary sizes
- Lower runtime overhead
- Very similar API
- Build on top of years of feedback and lessons learned from pybind11



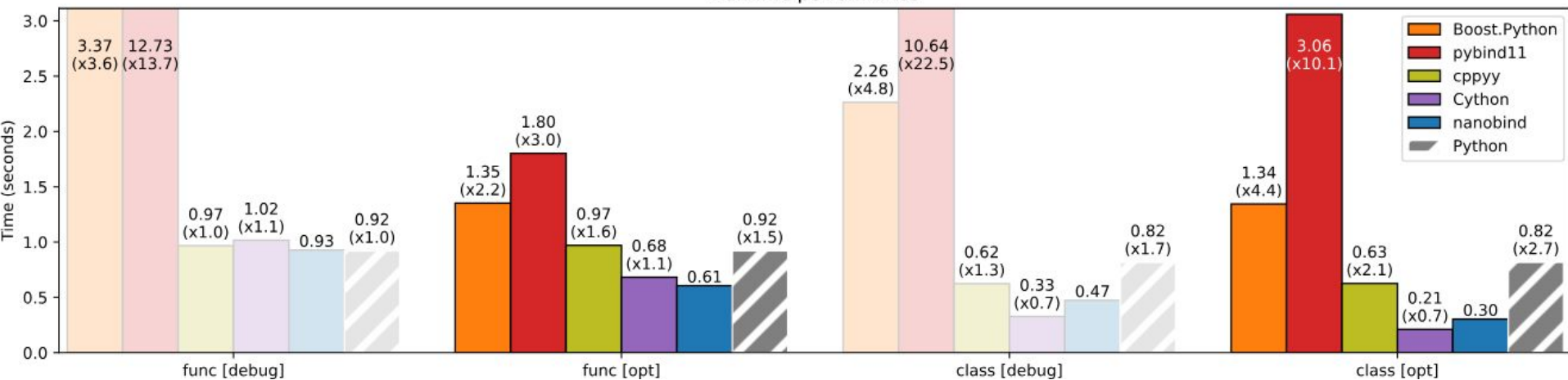
Compilation time

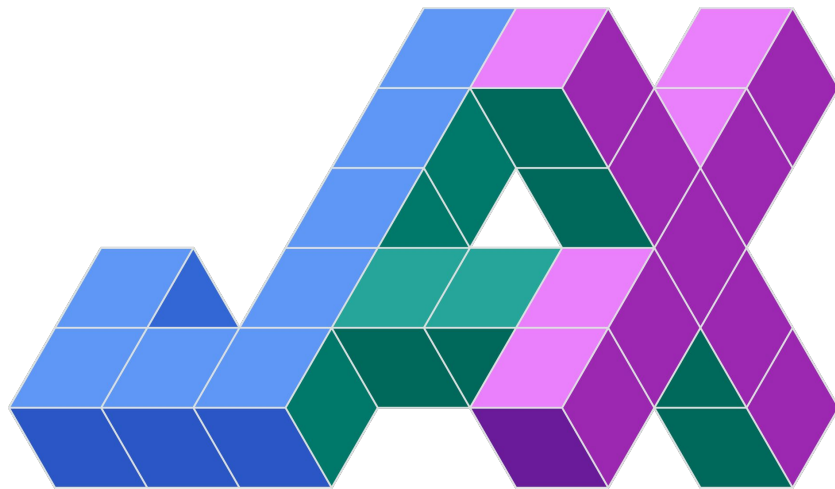
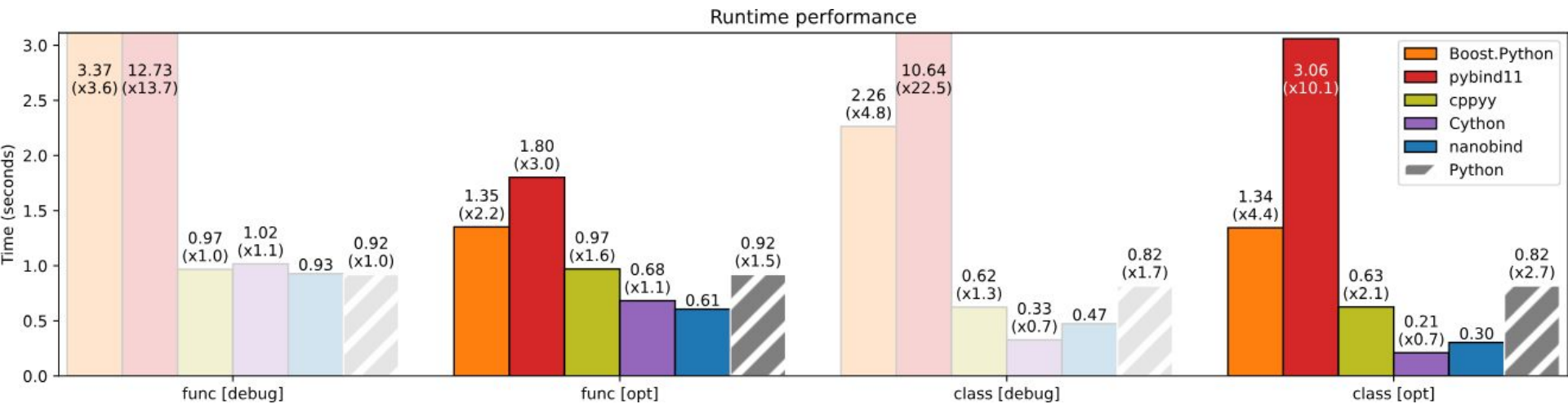


Binary size



Runtime performance





What is pybind11_json?

What is pybind11_json?

- Header-only library that bridges pybind11 and nlohmann/json

What is pybind11_json?

- Header-only library that bridges pybind11 and nlohmann/json
- Automatic conversion between **nlohmann::json** and Python types (dict, list, string, tuple etc.)

What is pybind11_json?

- Header-only library that bridges pybind11 and nlohmann/json
- Automatic conversion between **nlohmann::json** and Python types (dict, list, string, tuple etc.)
- Just include one header and it works! No manual conversion needed!

```
1  #include <pybind11/pybind11.h>
2  #include <pybind11_json/pybind11_json.hpp>
3  #include <nlohmann/json.hpp>
4
5  nlohmann::json process(nlohmann::json data) {
6      data["processed"] = true;
7      return data;
8  }
9
10 PYBIND11_MODULE(example, m) {
11     m.def("process", &process);
12 }
```

```
1  #include <pybind11/pybind11.h>
2  #include <pybind11_json/pybind11_json.hpp>
3  #include <nlohmann/json.hpp>
4
5  nlohmann::json process(nlohmann::json data) {
6      data["processed"] = true;
7      return data;
8  }
9
10 PYBIND11_MODULE(example, m) {
11     m.def("process", &process);
12 }
```

```
Python 3.12.3 (main, Mar 23 2026, 19:04:32) [GCC 13.3.0] on linux
Type "help", "copyright", "credits" or "license" for more information.
>>> import example
>>> a = {"name": "Pykonik", "year": 2026}
>>> example.process(a)
{'name': 'Pykonik', 'processed': True, 'year': 2026}
>>> 
```

Does anybody even use pybind11_json?

Does anybody even use pybind11_json?

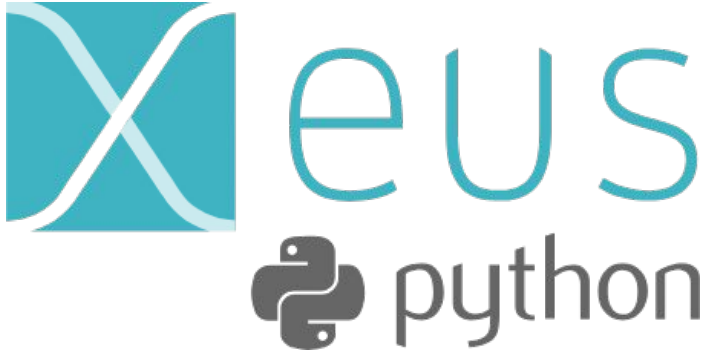
- Well...

Does anybody even use pybind11_json?

- Well... Yeah

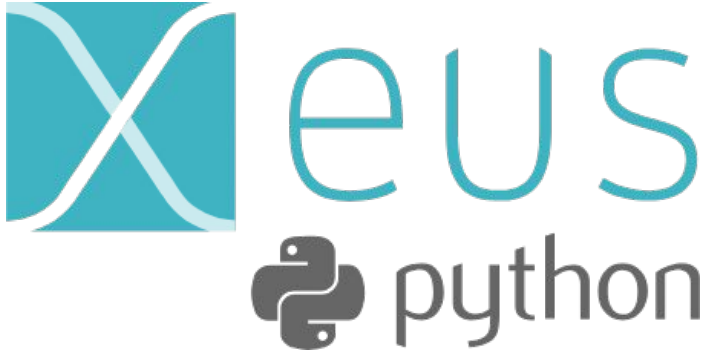
Does anybody even use pybind11_json?

- Well... Yeah



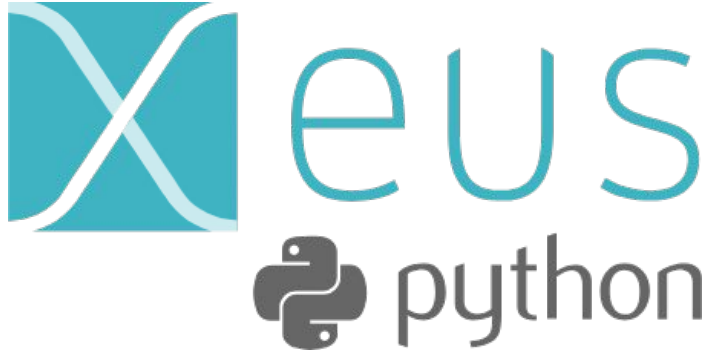
Does anybody even use pybind11_json?

- Well... Yeah



Does anybody even use pybind11_json?

- Well... Yeah

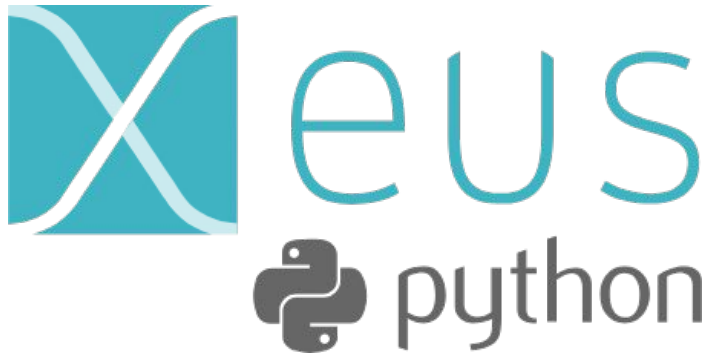


ROS2



Does anybody even use pybind11_json?

- Well... Yeah



ROS2

I contribute here!



Quickly on PyPartMC



Quickly on PyPartMC



- Pythonic wrapper for PartMC (“Particle-resolved Monte Carlo code for atmospheric aerosol simulation”)

Quickly on PyPartMC



- Pythonic wrapper for PartMC (“Particle-resolved Monte Carlo code for atmospheric aerosol simulation”)
- Developed at University of Illinois Urbana-Champaign and AGH University of Kraków

Quickly on PyPartMC



- Pythonic wrapper for PartMC (“Particle-resolved Monte Carlo code for atmospheric aerosol simulation”)
- Developed at University of Illinois Urbana-Champaign and AGH University of Kraków
- 4 languages in the codebase!

Quickly on PyPartMC



- Pythonic wrapper for PartMC (“Particle-resolved Monte Carlo code for atmospheric aerosol simulation”)
- Developed at University of Illinois Urbana-Champaign and AGH University of Kraków
- 4 languages in the codebase! (C, C++, Fortran, Python)

Quickly on PyPartMC



- Pythonic wrapper for PartMC (“Particle-resolved Monte Carlo code for atmospheric aerosol simulation”)
- Developed at University of Illinois Urbana-Champaign and AGH University of Kraków
- 4 languages in the codebase! (C, C++, Fortran, Python)
- Available on Linux, Windows and macOS

Quickly on PyPartMC



- Pythonic wrapper for PartMC (“Particle-resolved Monte Carlo code for atmospheric aerosol simulation”)
- Developed at University of Illinois Urbana-Champaign and AGH University of Kraków
- 4 languages in the codebase! (C, C++, Fortran, Python)
- Available on Linux, Windows and macOS
- Uses pybind11_json under the hood to provide a more Pythonic experience, and enable running without any auxiliary files

Quickly on PyPartMC



- Pythonic wrapper for PartMC (“Particle-resolved Monte Carlo code for atmospheric aerosol simulation”)
- Developed at University of Illinois Urbana-Champaign and AGH University of Kraków
- 4 languages in the codebase! (C, C++, Fortran, Python)
- Available on Linux, Windows and macOS
- Uses pybind11_json under the hood to provide a more Pythonic experience, and enable running without any auxiliary files
- We decided to switch to **nanobind**, we achieved an almost x4 speed-up on compilation!

So let's switch!

So let's switch!

- Changing pybind11 code to nanobind is relatively simple, most of the API is the same

So let's switch!

- Changing pybind11 code to nanobind is relatively simple, most of the API is the same
- We need an alternative to pybind11_json

So let's switch!

- Changing pybind11 code to nanobind is relatively simple, most of the API is the same
- We need an alternative to pybind11_json
- Luckily, it exists!

So let's switch!

- Changing pybind11 code to nanobind is relatively simple, most of the API is the same
- We need an alternative to pybind11_json
- Luckily, it exists!



Tip

Looking for an equivalent with nanobind? Check this out https://github.com/ianhbell/nanobind_json !

Hmm... just a small error

Hmm... just a small error

```
nanobind_json/include/nanobind_json/nanobind_json.hpp:16:10: fatal error: nanobind/nanobind.hpp: No such file or directory
   16 | #include "nanobind/nanobind.hpp"
      |           ^~~~~~
compilation terminated.
```

Hmm... just a small error

```
nanobind_json/include/nanobind_json/nanobind_json.hpp:16:10: fatal error: nanobind/nanobind.hpp: No such file or directory
```

```
16 | #include "nanobind/nanobind.hpp"
```

```
    | ^~~~~~
```

```
compilation terminated.
```

include/nanobind

- > eigen
- > intrusive
- > stl

eval.h

make_iterator.h

nanobind.h

```

/home/graca/Desktop/temp/temp1/gitmodules/nanobind/include/nanobind/nb_func.h: In instantiation of 'PyObject* nanobind::detail::func_create(Func&&, Return (*)(Args ...), std::index_sequence<Is2 ...>, const Extra& ...) [with bool ReturnRef = false; bool CheckGuard = true; Func = void (*)(nlohmann::json_abi_v3_12_0::basic_json<>); Return = void; Args = {nlohmann::json_abi_v3_12_0::basic_json<std::map, std::vector, std::__cxx11::basic_string<char, std::char_traits<char>, std::allocator<char> >, bool, long int, long unsigned int, double, std::allocator, nlohmann::json_abi_v3_12_0::adl_serializer, std::vector<unsigned char, std::allocator<unsigned char> >, void>}; long unsigned int ...Is = {0}; Extra = {nanobind::scope, nanobind::name}; PyObject = _object; std::index_sequence<Is2 ...> = std::integer_sequence<long unsigned int, 0>]':
/home/graca/Desktop/temp/temp1/gitmodules/nanobind/include/nanobind/nb_func.h:332:37:   required from 'void nanobind::cpp_function_def(Return (*)(Args ...), const Extra& ...) [with <template-parameter-1-1> = void; Return = void; Args = {nlohmann::json_abi_v3_12_0::basic_json<std::map, std::vector, std::__cxx11::basic_string<char, std::char_traits<char>, std::allocator<char> >, bool, long int, long unsigned int, double, std::allocator, nlohmann::json_abi_v3_12_0::adl_serializer, std::vector<unsigned char, std::allocator<unsigned char> >, void>}; Extra = {scope, name}]'
/home/graca/Desktop/temp/temp1/gitmodules/nanobind/include/nanobind/nb_func.h:409:21:   required from 'nanobind::module_& nanobind::module::_def(const char*, Func&&, const Extra& ...) [with Func = void (*)(nlohmann::json_abi_v3_12_0::basic_json<>); Extra = {}]'
/home/graca/Desktop/temp/temp1/bindings.cpp:25:10:   required from here
/home/graca/Desktop/temp/temp1/gitmodules/nanobind/include/nanobind/nb_func.h:178:64: error: 'Name' is not a member of 'nanobind::detail::make_caster<nlohmann::json_abi_v3_12_0::basic_json<> >'
178 |         make_caster<remove_opt_mono_t<intrinsic_t<Args>>>::Name)... +
    |                                     ~~~~~^~~~~
/home/graca/Desktop/temp/temp1/gitmodules/nanobind/include/nanobind/nb_func.h:254:41: error: 'struct nanobind::detail::type_caster<nlohmann::json_abi_v3_12_0::basic_json<> >' has no member named 'from_python'
254 |         if (!in.template get<Is>().from_python(args[Is], args_flags[Is],
    |                                     ~~~~~^~~~~
In file included from /home/graca/Desktop/temp/temp1/gitmodules/nanobind/include/nanobind/nanobind.h:53:
/home/graca/Desktop/temp/temp1/gitmodules/nanobind/include/nanobind/nb_cast.h: In substitution of 'template<class T> using nanobind::detail::cast_t = typename nanobind::detail::make_caster<T>::Cast<T> [with T = nlohmann::json_abi_v3_12_0::basic_json<>]':
/home/graca/Desktop/temp/temp1/gitmodules/nanobind/include/nanobind/nb_func.h:264:54:   required from 'PyObject* nanobind::detail::func_create(Func&&, Return (*)(Args ...), std::index_sequence<Is2 ...>, const Extra& ...) [with bool ReturnRef = false; bool CheckGuard = true; Func = void (*)(nlohmann::json_abi_v3_12_0::basic_json<>); Return = void; Args = {nlohmann::json_abi_v3_12_0::basic_json<std::map, std::vector, std::__cxx11::basic_string<char, std::char_traits<char>, std::allocator<char> >, bool, long int, long unsigned int, double, std::allocator, nlohmann::json_abi_v3_12_0::adl_serializer, std::vector<unsigned char, std::allocator<unsigned char> >, void>}; long unsigned int ...Is = {0}; Extra = {nanobind::scope, nanobind::name}; PyObject = _object; std::index_sequence<Is2 ...> = std::integer_sequence<long unsigned int, 0>]':
/home/graca/Desktop/temp/temp1/gitmodules/nanobind/include/nanobind/nb_func.h:332:37:   required from 'void nanobind::cpp_function_def(Return (*)(Args ...), const Extra& ...) [with <template-parameter-1-1> = void; Return = void; Args = {nlohmann::json_abi_v3_12_0::basic_json<std::map, std::vector, std::__cxx11::basic_string<char, std::char_traits<char>, std::allocator<char> >, bool, long int, long unsigned int, double, std::allocator, nlohmann::json_abi_v3_12_0::adl_serializer, std::vector<unsigned char, std::allocator<unsigned char> >, void>}; Extra = {scope, name}]'
/home/graca/Desktop/temp/temp1/gitmodules/nanobind/include/nanobind/nb_func.h:409:21:   required from 'nanobind::module_& nanobind::module::_def(const char*, Func&&, const Extra& ...) [with Func = void (*)(nlohmann::json_abi_v3_12_0::basic_json<>); Extra = {}]'
/home/graca/Desktop/temp/temp1/bindings.cpp:25:10:   required from here
/home/graca/Desktop/temp/temp1/gitmodules/nanobind/include/nanobind/nb_cast.h:59:29: error: no class template named 'Cast' in 'nanobind::detail::make_caster<nlohmann::json_abi_v3_12_0::basic_json<> >' {aka 'struct nanobind::detail::type_caster<nlohmann::json_abi_v3_12_0::basic_json<> >'}
59 | template <typename T> using cast_t = typename make_caster<T>::template Cast<T>;
    |                                     ~~~~~^~~~~

```

The problem begins... 439 lines of errors!

```
/home/graca/Desktop/temp/temp1/gitmodules/nanobind/include/nanobind/nb_func.h: In instantiation of 'PyObject* nanobind::detail::func_create(Func&&, Return (*)(Args ...), std::index_sequence<Is2 ...>, const Extra& ...) [with bool ReturnRef = false; bool CheckGuard = true; Func = void (*)(nlohmann::json_abi_v3_12_0::basic_json<>); Return = void; Args = {nlohmann::json_abi_v3_12_0::basic_json<std::map, std::vector, std::__cxx11::basic_string<char, std::char_traits<char>, std::allocator<char>>, bool, long int, long unsigned int, double, std::allocator, nlohmann::json_abi_v3_12_0::adl_serializer, std::vector<unsigned char, std::allocator<unsigned char>>, void>}; long unsigned int ...Is = {0}; Extra = {nanobind::scope, nanobind::name}; PyObject = _object; std::index_sequence<Is2 ...> = std::integer_sequence<long unsigned int, 0>]':
/home/graca/Desktop/temp/temp1/gitmodules/nanobind/include/nanobind/nb_func.h:332:37:   required from 'void nanobind::cpp_function_def(Return (*)(Args ...), const Extra& ...) [with <template-parameter-1-1> = void; Return = void; Args = {nlohmann::json_abi_v3_12_0::basic_json<std::map, std::vector, std::__cxx11::basic_string<char, std::char_traits<char>, std::allocator<char>>, bool, long int, long unsigned int, double, std::allocator, nlohmann::json_abi_v3_12_0::adl_serializer, std::vector<unsigned char, std::allocator<unsigned char>>, void>}; Extra = {scope, name}]'
/home/graca/Desktop/temp/temp1/gitmodules/nanobind/include/nanobind/nb_func.h:409:21:   required from 'nanobind::module_& nanobind::module_::def(const char*, Func&&, const Extra& ...) [with Func = void (*)(nlohmann::json_abi_v3_12_0::basic_json<>); Extra = {}]'
/home/graca/Desktop/temp/temp1/bindings.cpp:25:10:   required from here
/home/graca/Desktop/temp/temp1/gitmodules/nanobind/include/nanobind/nb_func.h:178:64: error: 'Name' is not a member of 'nanobind::detail::make_caster<nlohmann::json_abi_v3_12_0::basic_json<>>'
   178 |         make_caster<remove_opt_mono_t<intrinsic_t<Args>>>::Name>...) +
       |                                     ~~~~~^~~~~
/home/graca/Desktop/temp/temp1/gitmodules/nanobind/include/nanobind/nb_func.h:254:41: error: 'struct nanobind::detail::type_caster<nlohmann::json_abi_v3_12_0::basic_json<>>' has no member named 'from_python'
   254 |         if ((lin.template get<Is>()).from_python(args[Is], args_flags[Is],
In file included from /home/graca/Desktop/temp/temp1/gitmodules/nanobind/include/nanobind/nanobind.h:53:
/home/graca/Desktop/temp/temp1/gitmodules/nanobind/include/nanobind/nb_cast.h: In substitution of 'template<class T> using nanobind::detail::cast_t = typename nanobind::detail::make_caster<T>::Cast<T> [with T = nlohmann::json_abi_v3_12_0::basic_json<>]':
/home/graca/Desktop/temp/temp1/gitmodules/nanobind/include/nanobind/nb_func.h:264:54:   required from 'PyObject* nanobind::detail::func_create(Func&&, Return (*)(Args ...), std::index_sequence<Is2 ...>, const Extra& ...) [with bool ReturnRef = false; bool CheckGuard = true; Func = void (*)(nlohmann::json_abi_v3_12_0::basic_json<>); Return = void; Args = {nlohmann::json_abi_v3_12_0::basic_json<std::map, std::vector, std::__cxx11::basic_string<char, std::char_traits<char>, std::allocator<char>>, bool, long int, long unsigned int, double, std::allocator, nlohmann::json_abi_v3_12_0::adl_serializer, std::vector<unsigned char, std::allocator<unsigned char>>, void>}; long unsigned int ...Is = {0}; Extra = {nanobind::scope, nanobind::name}; PyObject = _object; std::index_sequence<Is2 ...> = std::integer_sequence<long unsigned int, 0>]':
/home/graca/Desktop/temp/temp1/gitmodules/nanobind/include/nanobind/nb_func.h:332:37:   required from 'void nanobind::cpp_function_def(Return (*)(Args ...), const Extra& ...) [with <template-parameter-1-1> = void; Return = void; Args = {nlohmann::json_abi_v3_12_0::basic_json<std::map, std::vector, std::__cxx11::basic_string<char, std::char_traits<char>, std::allocator<char>>, bool, long int, long unsigned int, double, std::allocator, nlohmann::json_abi_v3_12_0::adl_serializer, std::vector<unsigned char, std::allocator<unsigned char>>, void>}; Extra = {scope, name}]'
/home/graca/Desktop/temp/temp1/gitmodules/nanobind/include/nanobind/nb_func.h:409:21:   required from 'nanobind::module_& nanobind::module_::def(const char*, Func&&, const Extra& ...) [with Func = void (*)(nlohmann::json_abi_v3_12_0::basic_json<>); Extra = {}]'
/home/graca/Desktop/temp/temp1/bindings.cpp:25:10:   required from here
/home/graca/Desktop/temp/temp1/gitmodules/nanobind/include/nanobind/nb_cast.h:59:29: error: no class template named 'Cast' in 'nanobind::detail::make_caster<nlohmann::json_abi_v3_12_0::basic_json<>>'
   59 | template<typename T> using cast_t = typename make_caster<T>::template Cast<T>;
```

Let's get in touch with the maintainer!

Let's get in touch with the maintainer!

Hi there 🖐️

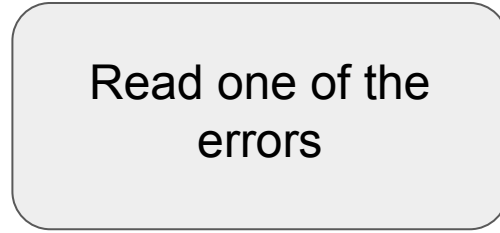
As of January 10, 2025 this account will not be actively maintained as no longer work at NIST. For support, please see my contact info on LinkedIn.

So let's fix it ourselves!

So let's fix it ourselves!

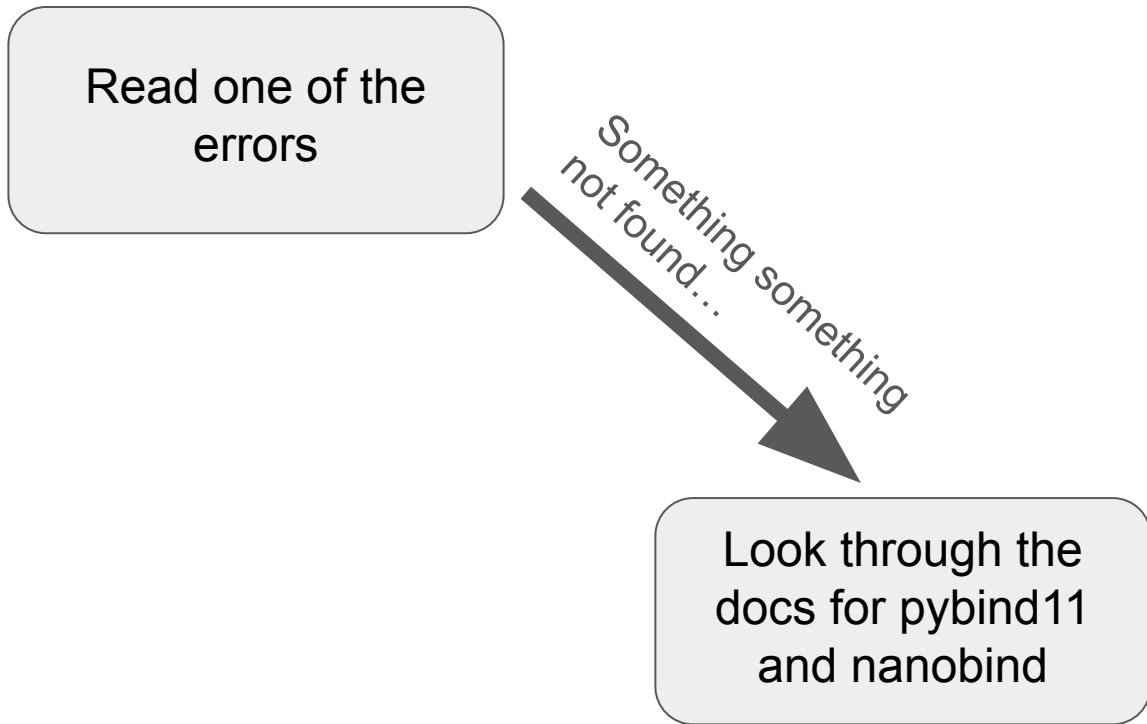
Read one of the
errors

So let's fix it ourselves!

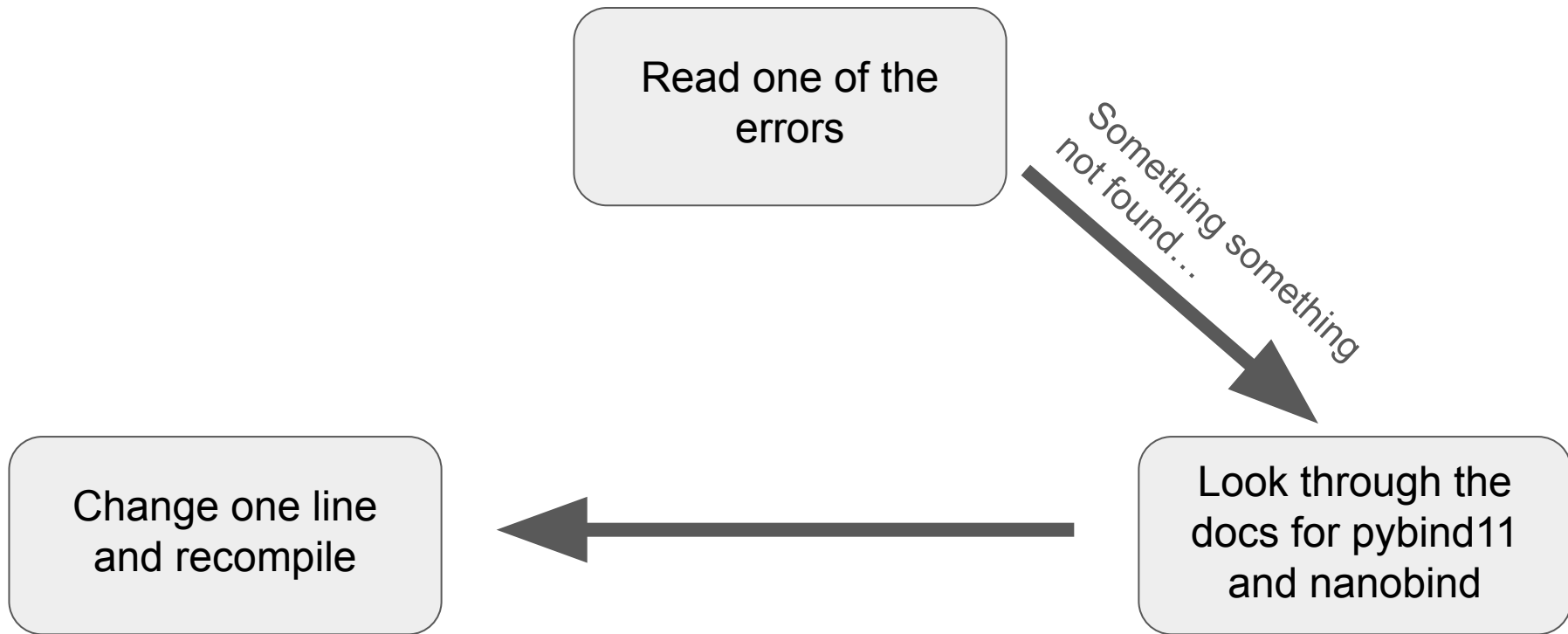


Something something
not found...

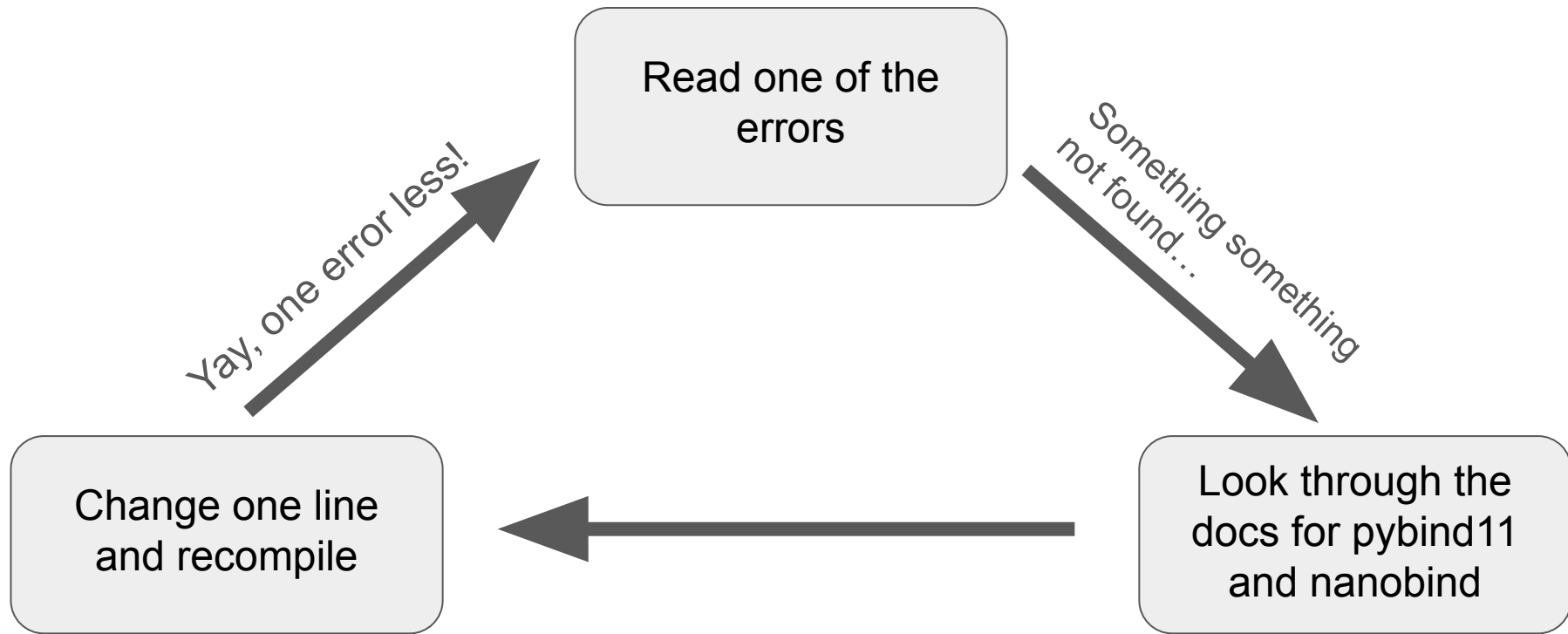
So let's fix it ourselves!



So let's fix it ourselves!



So let's fix it ourselves!



Example: cast()

Example: cast()

```
class handle : public detail::object_api<handle>
```

Holds a reference to a Python object (no reference counting)

```
template<typename T>  
    T cast() const
```

Attempt to cast the Python object into the given C++ type. A `cast_error` will be throw upon failure.

Example: cast()

```
class handle : public detail::object_api<handle>
```

Holds a reference to a Python object (no reference counting)

```
template<typename T>  
    T cast() const
```

Attempt to cast the Python object into the given C++ type. A `cast_error` will be throw upon failure.

```
pybind11::handle a = pybind11::int_(5);  
int b = a.cast<int>();
```

Example: cast()

```
template<typename T, typename Derived>
```

```
    T cast(const detail::api<Derived> &value, bool convert = true)
```

Convert the Python object `value` (typically a `handle` or a `object` subclass) into a C++ object of type `T`.

```
[](nb::object a) {  
    int b = nb::cast<int>(a);  
    return b + 1;  
};
```

Example: load() and cast()

Example: load() and cast()

Python to C++

```
bool load(handle src,  
         bool convert) {
```

C++ to Python

```
static handle cast(  
    const YourType &number,  
    return_value_policy policy,  
    handle parent) {
```

Example: load() and cast()

Python to C++

```
bool load(handle src,  
         bool convert) {
```

Python to C++

```
bool from_python(handle src,  
                uint8_t flags,  
                cleanup_list *cleanup)  
noexcept {
```

C++ to Python

```
static handle cast(  
    const YourType &number,  
    return_value_policy policy,  
    handle parent) {
```

C++ to Python

```
static handle from_cpp(  
    YourType &&value,  
    rv_policy policy,  
    cleanup_list *cleanup)  
noexcept {
```

Other problems:

Other problems: Circular References

Other problems: Circular References

```
@staticmethod
@pytest.mark.skipif(platform.machine() == "arm64", reason="TODO #348")
def test_fixed_segfault_case_on_circular_reference():
    # arrange
    aero_data = ppmc.AeroData(AERO_DATA_CTOR_ARG_MINIMAL)
    fishy_ctor_arg = copy.deepcopy(AERO_MODE_CTOR_LOG_NORMAL)
    fishy_ctor_arg["test_mode"]["mass_frac"].append(
        fishy_ctor_arg["test_mode"]["mass_frac"]
    )

    # act
    with pytest.raises(RuntimeError) as exc_info:
        ppmc.AeroMode(aero_data, fishy_ctor_arg)
```

Other problems: Circular References

```
@staticmethod
@pytest.mark.skipif(platform.machine() == "arm64", reason="TODO #348")
def test_fixed_segfault_case_on_circular_reference():
    # arrange
    aero_data = ppmc.AeroData(AERO_DATA_CTOR_ARG_MINIMAL)
    fishy_ctor_arg = copy.deepcopy(AERO_MODE_CTOR_LOG_NORMAL)
    fishy_ctor_arg["test_mode"]["mass_frac"].append(
        fishy_ctor_arg["test_mode"]["mass_frac"]
    )
```

Avoid crash when converting dict with circular reference #74



Merged

[martinRenou](#) merged 1 commit into [pybind:master](#) from [dbaston:fix-73](#)



on Dec 9, 2024

Other problems: Circular References

```
inline nl::json to_json(const py::handle& obj, std::set<const PyObject*>& refs)
{
```

Other problems: Circular References

```
inline nl::json to_json(const py::handle& obj, std::set<const PyObject*>& refs)
{
```

```
    auto insert_ret = refs.insert(obj.ptr());
    if (!insert_ret.second) {
        throw std::runtime_error("Circular reference detected");
    }
```

Other problems: Circular References

```
inline nl::json to_json(const py::handle& obj, std::set<const PyObject*>& refs)
{
```

```
    auto insert_ret = refs.insert(obj.ptr());
    if (!insert_ret.second) {
        throw std::runtime_error("Circular reference detected");
    }
```

```
    refs.erase(insert_ret.first);
```

Everything works!

Everything works!

- PyPartMC compiles

Everything works!

- PyPartMC compiles
- All 400 tests are passing

Everything works!

- PyPartMC compiles
- All 400 tests are passing
- Let's fork nanobind_json and add our changes!

Time to fork

Time to fork

ianhbell / nanobind_json

Search Type / to search

Code Issues Pull requests 1 Agents Actions Projects Security and quality Insights

nanobind_json Public

Watch 3 Fork 4

main 1 Branch 0 Tags

Go to file T Add file <> Code

ianhbell Update verbiage e195353 · 2 years ago 3 Commits

.github/workflows	First commit	2 years ago
include/nanobind_json	First commit	2 years ago
test	First commit	2 years ago
.gitignore	First commit	2 years ago
CMakeLists.txt	First commit	2 years ago
LICENSE	First commit	2 years ago
README.md	Update verbiage	2 years ago
nanobind_jsonConfig.cmake.in	First commit	2 years ago
pixi.lock	First commit	2 years ago

About

nanobind wrappers
nlhmann::json libra

Readme

BSD-3-Clause licen

Activity

11 stars

3 watching

4 forks

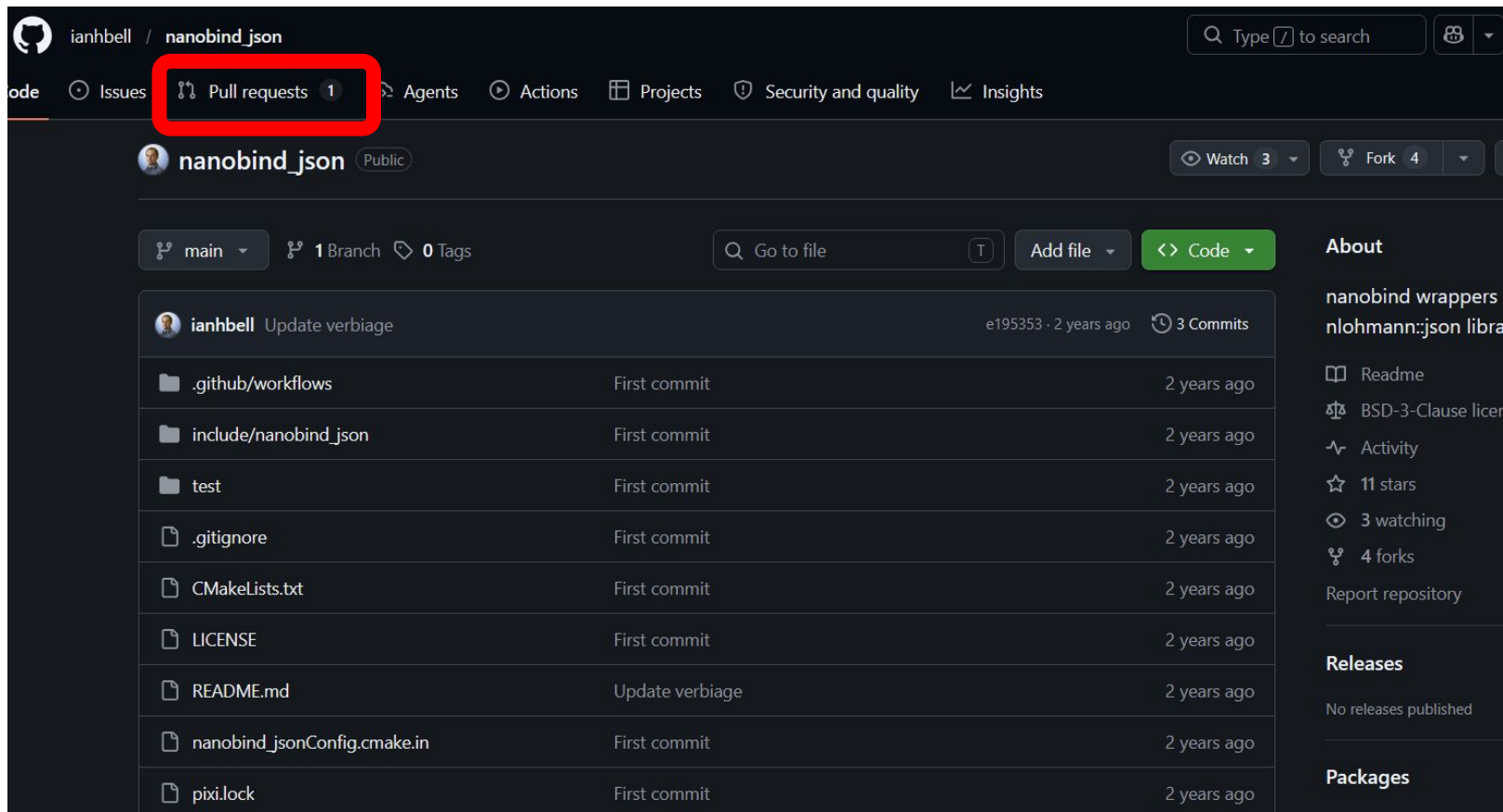
Report repository

Releases

No releases published

Packages

Time to fork



ianhbell / nanobind_json

code Issues Pull requests 1 Agents Actions Projects Security and quality Insights

nanobind_json Public

Watch 3 Fork 4

main 1 Branch 0 Tags

Go to file Add file Code

ianhbell Update verbiage e195353 · 2 years ago 3 Commits

.github/workflows	First commit	2 years ago
include/nanobind_json	First commit	2 years ago
test	First commit	2 years ago
.gitignore	First commit	2 years ago
CMakeLists.txt	First commit	2 years ago
LICENSE	First commit	2 years ago
README.md	Update verbiage	2 years ago
nanobind_jsonConfig.cmake.in	First commit	2 years ago
pixi.lock	First commit	2 years ago

About

nanobind wrappers
nlohmann::json library

Readme
BSD-3-Clause license
Activity
11 stars
3 watching
4 forks
Report repository

Releases

No releases published

Packages

Time to fork



Update for nanobind v2.5.0

#1 opened on Mar 25, 2025 by pearzt

Time to fork



Update for nanobind v2.5.0

#1 opened on Mar 25, 2025 by pearzt



pearzt commented [on Mar 25, 2025](#)



I'm not particularly familiar with the internals of neither `nlohmann_json` nor `nanobind`, but I managed to update this for `nanobind v2.5.0` and make it work for my use case. I'm posting this here just in case it's helpful for other people.



1

The Plan:

The Plan:

- Create a fork, add a temporary branch with the working setup for PyPartMC

The Plan:

- Create a fork, add a temporary branch with the working setup for PyPartMC
- Merge pearzt's PR

The Plan:

- Create a fork, add a temporary branch with the working setup for PyPartMC
- Merge pearzt's PR
- Adjust where necessary

The Plan:

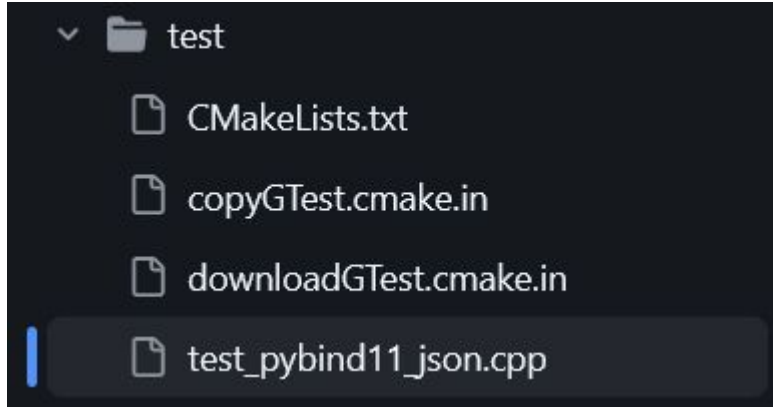
- Create a fork, add a temporary branch with the working setup for PyPartMC
- Merge pearzt's PR
- Adjust where necessary
- Write tests, covering the same scope as pybind11_json

The Plan:

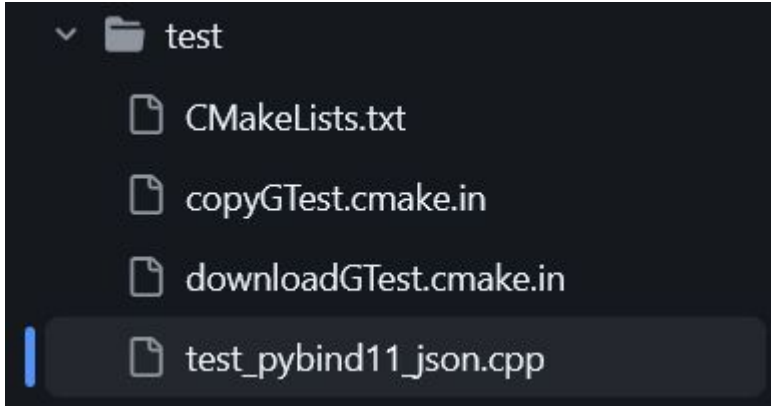
- Create a fork, add a temporary branch with the working setup for PyPartMC
- Merge pearzt's PR
- Adjust where necessary
- Write tests, covering the same scope as pybind11_json
- Add CI workflow for testing

Problem: No embedded interpreter

Problem: No embedded interpreter



Problem: No embedded interpreter



```
#include "pybind11/embed.h"
```

Problem: No embedded interpreter

test

CMakeLists.txt

copyGTest.cmake.in

downloadGTest.cmake.in

test_pybind11_json.cpp

```
#include "pybind11/embed.h"
```

```
TEST(nljson_serializers_tojson, bool_)
{
    py::scoped_interpreter guard;
    py::bool_ obj(false);
    nl::json j = obj;

    ASSERT_TRUE(j.is_boolean());
    ASSERT_FALSE(j.get<bool>());
}
```

Problem: No embedded interpreter

```
test
├── CMakeLists.txt
├── copyGTest.cmake.in
├── downloadGTest.cmake.in
└── test_pybind11_json.cpp
```

```
#include "pybind11/embed.h"
```

```
TEST(nljson_serializers_tojson, bool_)
{
    py::scoped_interpreter guard;
    py::bool_ obj(false);
    nl::json j = obj;

    ASSERT_TRUE(j.is_boolean());
    ASSERT_FALSE(j.get<bool>());
}
```

- ○ **Multi-interpreter, Embedding:** The ability to embed Python in an executable or run several independent Python interpreters in the same process is unsupported. Nanobind caters to bindings only. Multi-interpreter support would require TLS lookups for nanobind data structures, which is undesirable.

Problem: No embedded interpreter

 test_accessor.cpp

 test_accessor.py

 test_callbacks.cpp

 test_callbacks.py

 test_chrono.cpp

 test_chrono.py

 test_classes.cpp

 test_classes.h

 test_classes.py

Problem: No embedded interpreter

test_accessor.cpp

test_accessor.py

test_callbacks.cpp

test_callbacks.py

test_chrono.cpp

test_chrono.py

test_classes.cpp

test_classes.h

test_classes.py

```
m.def("vec_copyable_in_value", [](std::vector<Copyable> x) {  
    if (x.size() != 10)  
        fail();  
    for (int i = 0; i < 10; ++i)  
        if (x[i].value != i)  
            fail();  
});
```

Problem: No embedded interpreter

test_accessor.cpp

test_accessor.py

test_callbacks.cpp

test_callbacks.py

test_chrono.cpp

test_chrono.py

test_classes.cpp

test_classes.h

test_classes.py

```
m.def("vec_copyable_in_value", [](std::vector<Copyable> x) {  
    if (x.size() != 10)  
        fail();  
    for (int i = 0; i < 10; ++i)  
        if (x[i].value != i)  
            fail();  
});
```

```
def test25_vec_movable_in_value(clean):  
    t.vec_copyable_in_value([t.Copyable(i) for i in range(10)])  
    assert_stats(value_constructed=10, copy_constructed=10, destructed=20)
```

First test: Simple boolean

First test: Simple boolean

```
m.def("nljson_bool_fromjson", []() {  
  return "false"_json;  
});  
  
m.def("nljson_bool_tojson", [](nl::json bool_json) {  
  if (!bool_json.is_boolean()) {  
    fail();  
  }  
  else if (bool_json.get<bool>() != false) {  
    fail();  
  }  
});
```

First test: Simple boolean

```
m.def("nljson_bool_fromjson", []() {  
  return "false"_json;  
});  
  
m.def("nljson_bool_tojson", [](nl::json bool_json) {  
  if (!bool_json.is_boolean()) {  
    fail();  
  }  
  else if (bool_json.get<bool>() != false) {  
    fail();  
  }  
});
```

```
def test_nljson_bool_from_json():  
    json = t.nljson_bool_fromjson()  
  
    assert isinstance(json, bool)  
    assert json == False
```

```
def test_nljson_bool_to_json():  
    t.nljson_bool_tojson(False)
```

First test: Simple boolean

```
rootdir: /home/gadamus/zzz/nanobind_json
collected 2 items

../../tests/test_json.py::test_nljson_bool_from_json PASSED [ 50%]
../../tests/test_json.py::test_nljson_bool_to_json FAILED [100%]

===== FAILURES =====
----- test_nljson_bool_to_json -----

    def test_nljson_bool_to_json():
>         t.nljson_bool_tojson(False)
E         RuntimeError: std::exception

../../tests/test_json.py:22: RuntimeError
===== short test summary info =====
FAILED ../../tests/test_json.py::test_nljson_bool_to_json - Runtime
eError: std::exception
===== 1 failed, 1 passed in 0.03s =====
```

First test: Simple boolean

```
m.def("nljson_bool_tojson", [](nl::json bool_json) {  
    if (!bool_json.is_boolean()) {  
        std::cerr << "is not boolean\n";  
        fail();  
    }  
    else if (bool_json.get<bool>() != false) {  
        std::cerr << "is not false\n";  
        fail();  
    }  
});
```

First test: Simple boolean

```
rootdir: /home/gadamus/zzz/nanobind_json
collected 2 items

../tests/test_json.py::test_nljson_bool_from_json PASSED [ 50%]
../tests/test_json.py::test_nljson_bool_to_json FAILED [100%]

===== FAILURES =====
----- test_nljson_bool_to_json -----

    def test_nljson_bool_to_json():
>         t.nljson_bool_tojson(False)
E         RuntimeError: std::exception

../tests/test_json.py:22: RuntimeError
----- Captured stderr call -----
is not boolean
===== short test summary info =====
FAILED ../tests/test_json.py::test_nljson_bool_to_json - Runtime
eError: std::exception
===== 1 failed, 1 passed in 0.03s =====
```

First test: Simple boolean

```
m.def("nljson_bool_tojson", [](nl::json bool_json) {  
    std::cerr << bool_json.dump() << '\n';  
    if (!bool_json.is_boolean()) {  
        fail();  
    }  
    else if (bool_json.get<bool>() != false) {  
        fail();  
    }  
});
```

First test: Simple boolean

```
rootdir: /home/gadamus/zzz/nanobind_json
collected 2 items

../..../tests/test_json.py::test_nljson_bool_from_json PASSED [ 50%]
../..../tests/test_json.py::test_nljson_bool_to_json FAILED [100%]

===== FAILURES =====
----- test_nljson_bool_to_json -----

    def test_nljson_bool_to_json():
>         t.nljson_bool_tojson(False)
E         RuntimeError: std::exception

../..../tests/test_json.py:22: RuntimeError
----- Captured stderr call -----
null
===== short test summary info =====
FAILED ../..../tests/test_json.py::test_nljson_bool_to_json - Runtime
eError: std::exception
===== 1 failed, 1 passed in 0.03s =====
```

What's going on?

What's going on?

- Only `from_python()` fails

What's going on?

- Only **from_python()** fails
- PyPartMC works fine, despite sending JSONs from Python to C++

What's going on?

- Only **from_python()** fails
- PyPartMC works fine, despite sending JSONs from Python to C++
- Receives “null” instead of bool

What's going on?

- Only **from_python()** fails
- PyPartMC works fine, despite sending JSONs from Python to C++
- Receives “null” instead of bool
- Sending dictionaries or lists also fails

What's going on?

- Only **from_python()** fails
- PyPartMC works fine, despite sending JSONs from Python to C++
- Receives “null” instead of bool
- Sending dictionaries or lists also fails
- So what's wrong?

What's going on?

- Only **from_python()** fails
- PyPartMC works fine, despite sending JSONs from Python to C++
- Receives “null” instead of bool
- Sending dictionaries or lists also fails
- So what's wrong?
- Oh, PyPartMC uses a different branch, where we didn't merge that other pull request

The culprit revealed

The culprit revealed

```
bool from_python(handle src, uint8_t flags, cleanup_list *cleanup) {  
    try {  
        auto value = nbjson::to_json<nl::json>(src);  
        return true;  
    }  
    catch (nbjson::CircularReferenceError &e) {  
        throw e;  
    }  
    catch (...) {  
        return false;  
    }  
}
```

The culprit revealed

```
bool from_python(handle src, uint8_t flags, cleanup_list *cleanup) {  
    try {  
        auto value = nbjson::to_json<nl::json>(src);  
        return true;  
    }  
    catch (nbjson::CircularReferenceError &e) {  
        throw e;  
    }  
    catch (...) {  
        return false;  
    }  
}
```

The culprit revealed

```
bool from_python(handle src, uint8_t flags, cleanup_list *cleanup) {  
    try {  
        value = nbjson::to_json<nl::json>(src);  
        return true;  
    }  
    catch (nbjson::CircularReferenceError &e) {  
        throw e;  
    }  
    catch (...) {  
        return false;  
    }  
}
```

Let's add CI testing on Github Actions!

Let's add CI testing on Github Actions!

```
steps:
- uses: actions/checkout@v4

- name: Setup Python ${ matrix.python }
  uses: actions/setup-python@v5
  with:
    python-version: ${ matrix.python }
    cache: 'pip'

- name: Install pytest
  run: python -m pip install pytest

- name: Configure CMake
  run: cmake -S $GITHUB_WORKSPACE -B $GITHUB_WORKSPACE/build/ -DBUILD_TESTS=1

- name: Build test modules
  run: cmake --build $GITHUB_WORKSPACE/build/

- name: Run tests
  run: |
    cd build
    python -m pytest ../tests
```

  add: draft CI workflow for testing	 b3d2eaf
  fix: typo	 84827b7
  update: download nlohmann_json with FetchContent	 9796c65
  fix: directory name change	 3f5f098
  change includes in test file	 b6a9fd7
  include nlohmann headers	 ea9da44
  fix: CMake	 32542fb
  fix includes, add git tags	 90aa765
  fix includes	 41bdd2c
  try to debug windows CI	 02d1591
  set python-version in setup-python	 4382256
  add dummy pyproject.toml	 345ce28
  enable tmate session	 af86e0c
  debug CI	 84a02a4
  use nanobind_install runtime dlls	 acd14de
  try to run pytest from build/Debug	 9877155
  fix	 47091b1
  set cmake msvc runtime libs	 d104ec4

  add: draft CI workflow for testing	  download Dependencies CLI tool	 6816743
  fix: typo	  chore	 b2e64bf
  update: download nlohmann_json with Fet	  change build type to release	 34d4d7d
  fix: directory name change	  config release	 0c19154
  change includes in test file	  fix	 9f46508
  include nlohmann headers	  fix	 a8462ab
  fix: CMake	  tmate session	 1219a90
  fix includes, add git tags	  fix	 5d4b288
  fix includes	  VC Redistributable	 64f477c
  try to debug windows CI	  env	 942528f
  set python-version in setup-python	  try a dll workaround	 c515875
  add dummy pyproject.toml		
  enable tmate session	 af86e0c	
  debug CI	 84a02a4	
  use nanobind_install_runtime_dlls	 acd14de	
  try to run pytest from build/Debug	 9877155	
  fix	 47091b1	
  set cmake msvc runtime libs	 d104ec4	

 add: draft CI workflow for testing	 download Dependencies CLI tool	 6816743
 fix: typo	 chore	 b2e64bf
 update: download nlohmann_json with Fet	 change build type to release	 34d4d7d
 fix: directory name change	 config release	 0c19154
 change includes in test file	 fix	 9f46508
 include nlohmann headers	 fix	 a8462ab
 fix: CMake	 tmate session	 1219a90
 fix includes, add git tags	 fix	 5d4b288
 fix includes	 VC Redistributable	 64f477c
 try to debug windows CI	 env	 942528f
 set python-version in setup-python	 try a dll workaround	 c515875
 add dummy pyproject.toml		
 enable tmate session	 af86e0c	
 debug CI		
 use nanobind_install_runtime_dlls		
 try to run pytest from build/Debug	 9877155	
 fix	 47091b1	
 set cmake msvc runtime libs	 d104ec4	

Griger5 merged 59 commits into **main** from **ci-workflow**

Let's look at some failing workflow runs

Let's look at some failing workflow runs



The logs for this run have expired and are no longer available.

- ✓ Build and run tests on ubuntu-latest
- ✓ Build and run tests on ubuntu-latest
- ✓ Build and run tests on ubuntu-latest
- ✓ Build and run tests on ubuntu-latest
- ✓ Build and run tests on macos-latest
- ✓ Build and run tests on macos-latest
- ✓ Build and run tests on macos-latest
- ✓ Build and run tests on macos-latest
- ✗ Build and run tests on windows-latest
- ✗ Build and run tests on windows-latest
- ✗ Build and run tests on windows-latest
- ✗ Build and run tests on windows-latest

What were the problems?

What were the problems?

- Wrong yaml syntax

What were the problems?

- Wrong yaml syntax
- Wrong paths (specifically Windows because of MSVC)

What were the problems?

- Wrong yaml syntax
- Wrong paths (specifically Windows because of MSVC)
- Python package not found (screwed up installation)

What were the problems?

- Wrong yaml syntax
- Wrong paths (specifically Windows because of MSVC)
- Python package not found (screwed up installation)
- But what took most of my time was...

A head-scratching time waster

✗ Build and run tests on
ubuntu-latest with Python3.8

✗ Build and run tests on
ubuntu-latest with Python3.9

✗ **Build and run tests on
ubuntu-latest with
Python3.10**

✗ Build and run tests on
ubuntu-latest with Python3.11

✓ Build and run tests on
ubuntu-latest with Python3.12

✓ Build and run tests on
ubuntu-latest with Python3.13

Traceback:

```
/opt/hostedtoolcache/Python/3.10.20/x64/lib/python3.10/importlib/__init__.py:126: in  
import_module
```

```
    return _bootstrap._gcd_import(name[level:], package, level)
```

```
../tests/test_json.py:5: in <module>
```

```
    import test_json_ext as t
```

```
E ImportError: /home/runner/work/nanobind_json/nanobind_json/build/  
test_json_ext.cpython-310-x86_64-linux-gnu.so: undefined symbol: PyObject_Vectorcall
```

```
===== short test summary info =====
```

```
ERROR ../tests/test_json.py
```

```
!!!!!!!!!!!!!!!!!!!! Interrupted: 1 error during collection !!!!!!!!!!!!!!!!!!!!!
```

A head-scratching time waster

- ✗ Build and run tests on macos-latest with Python3.8
- ✗ Build and run tests on macos-latest with Python3.9
- ✗ Build and run tests on macos-latest with Python3.10
- ✗ Build and run tests on macos-latest with Python3.11**
- ✗ Build and run tests on macos-latest with Python3.12
- ✗ Build and run tests on macos-latest with Python3.13

Traceback:

```
/Library/Frameworks/Python.framework/Versions/3.11/lib/python3.11/importlib/
__init__.py:126: in import_module
    return _bootstrap._gcd_import(name[level:], package, level)
    ^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^
../tests/test_json.py:5: in <module>
    import test_json_ext as t
E ImportError: dlopen(/Users/runner/work/nanobind_json/nanobind_json/build/
test_json_ext.cpython-311-darwin.so, 0x0002): symbol not found in flat namespace
'_PyDict_GetItemRef'
===== short test summary info =====
ERROR ../tests/test_json.py
!!!!!!!!!!!!!!!!!!!! Interrupted: 1 error during collection !!!!!!!!!!!!!!!!!!!!!
```

A head-scratching time waster

✓ Build and run tests on windows-latest with Python3.8

✓ Build and run tests on windows-latest with Python3.9

✓ Build and run tests on windows-latest with Python3.10

✓ Build and run tests on windows-latest with Python3.11

✓ Build and run tests on windows-latest with Python3.12

✗ Build and run tests on windows-latest with Python3.13

Traceback:

```
C:\hostedtoolcache\windows\Python\3.13.13\x64\Lib\importlib\__init__.py:88: in
import_module
    return _bootstrap._gcd_import(name[level:], package, level)
    ^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^
..\..\tests\test_json.py:5: in <module>
    import test_json_ext as t
E   ImportError: DLL load failed while importing test_json_ext: The specified module could
not be found.

===== short test summary info =====
ERROR ..\..\tests\test_json.py
!!!!!!!!!!!!!!!!!!!! Interrupted: 1 error during collection !!!!!!!!!!!!!!!!!!!!!
```

The second culprit revealed

The second culprit revealed

```
-- The C compiler identification is MSVC 19.44.35226.0
-- The CXX compiler identification is MSVC 19.44.35226.0
-- Detecting C compiler ABI info
-- Detecting C compiler ABI info - done
-- Check for working C compiler: C:/Program Files/Microsoft Visual Studio/2022/Enterprise/VC/Tools/MSVC/14.44.35207/bin/Hostx64/x64/cl.exe - skipped
-- Detecting C compile features
-- Detecting C compile features - done
-- Detecting CXX compiler ABI info
-- Detecting CXX compiler ABI info - done
-- Check for working CXX compiler: C:/Program Files/Microsoft Visual Studio/2022/Enterprise/VC/Tools/MSVC/14.44.35207/bin/Hostx64/x64/cl.exe - skipped
-- Detecting CXX compile features
-- Detecting CXX compile features - done
-- Found Python: C:/hostedtoolcache/windows/Python/3.13.13/x64/python3.exe (found suitable version "3.13.13", minimum required is "3.9") found components: Interpreter Development.Module Development.SABIModule
-- Using the multi-header code from D:/a/nanobind_json/nanobind_json/build/_deps/nlohmann_json-src/include/
-- Found Python: C:/hostedtoolcache/windows/Python/3.14.4/x64/python3.exe (found version "3.14.4") found components: Interpreter Development.Module Development.Embed
-- Configuring done (32.3s)
-- Generating done (0.1s)
-- Build files have been written to: D:/a/nanobind_json/nanobind_json/build
```

The second culprit revealed

```
-- The C compiler identification is MSVC 19.44.35226.0
-- The CXX compiler identification is MSVC 19.44.35226.0
-- Detecting C compiler ABI info
-- Detecting C compiler ABI info - done
-- Check for working C compiler: C:/Program Files/Microsoft Visual Studio/2022/Enterprise/VC/Tools/MSVC/14.44.35207/bin/Hostx64/x64/cl.exe - skipped
-- Detecting C compile features
-- Detecting C compile features - done
-- Detecting CXX compiler ABI info
-- Detecting CXX compiler ABI info - done
-- Check for working CXX compiler: C:/Program Files/Microsoft Visual Studio/2022/Enterprise/VC/Tools/MSVC/14.44.35207/bin/Hostx64/x64/cl.exe - skipped
-- Detecting CXX compile features
-- Detecting CXX compile features - done
-- Found Python: C:/hostedtoolcache/windows/Python/3.13.13/x64/python3.exe (found suitable version "3.13.13",
minimum required is "3.9") found components: Interpreter Development.Module Development.SABIModule
-- Using the multi-header code from D:/a/nanobind_json/nanobind_json/build/_deps/nlohmann_json-src/include/
-- Found Python: C:/hostedtoolcache/windows/Python/3.14.4/x64/python3.exe (found version "3.14.4") found
components: Interpreter Development.Module Development.Embed
-- Configuring done (32.3s)
-- Generating done (0.1s)
-- Build files have been written to: D:/a/nanobind_json/nanobind_json/build
```

The second culprit revealed

```
-- The C compiler identification is MSVC 19.44.35226.0
-- The CXX compiler identification is MSVC 19.44.35226.0
-- Detecting C compiler ABI info
-- Detecting C compiler ABI info - done
-- Check for working C compiler: C:/Program Files/Microsoft Visual Studio/2022/Enterprise/VC/Tools/MSVC/14.44.35207/bin/Hostx64/x64/cl.exe - skipped
-- Detecting C compile features
-- Detecting C compile features - done
-- Detecting CXX compiler ABI info
-- Detecting CXX compiler ABI info - done
-- Check for working CXX compiler: C:/Program Files/Microsoft Visual Studio/2022/Enterprise/VC/Tools/MSVC/14.44.35207/bin/Hostx64/x64/cl.exe - skipped
-- Detecting CXX compile features
-- Detecting CXX compile features - done
-- Found Python: C:/hostedtoolcache/windows/Python/3.13.13/x64/python3.exe (found suitable version "3.13.13",
but the minimum required is "3.9") found components: Interpreter Development.Module Development.Embed
-- Using the multi-header code from D:/a/nanobind_json/nanobind_json/build/_deps/nlohmann_json-src/include/
-- Found Python: C:/hostedtoolcache/windows/Python/3.14.4/x64/python3.exe (found version "3.14.4") found
components: Interpreter Development.Module Development.Embed
-- Configuring done (32.3s)
-- Generating done (0.1s)
-- Build files have been written to: D:/a/nanobind_json/nanobind_json/build
```

The second culprit revealed

```
-- The C compiler identification is MSVC 19.44.35226.0
-- The CXX compiler identification is MSVC 19.44.35226.0
-- Detecting C compiler ABI info
-- Detecting C compiler ABI info - done
-- Check for working C compiler: C:/Program Files/Microsoft Visual Studio/2022/Enterprise/VC/Tools/MSVC/14.44.35207/bin/Hostx64/x64/cl.exe - skipped
-- Detecting C compile features
-- Detecting C compile features - done
-- Detecting CXX compiler ABI info
-- Detecting CXX compiler ABI info - done
-- Check for working CXX compiler: C:/Program Files/Microsoft Visual Studio/2022/Enterprise/VC/Tools/MSVC/14.44.35207/bin/Hostx64/x64/cl.exe - skipped
-- Detecting CXX compile features
-- Detecting CXX compile features - done
-- Found Python: C:/hostedtoolcache/windows/Python/3.13.13/x64/python3.exe (found suitable version "3.13.13",
minimum required is "3.9") found components: Interpreter Development.Module Development.Embed
-- Using the main header code from D:/a/nanobind_json/nanobind_json/build/_deps/nlohmann_json-src/include/
-- Found Python: C:/hostedtoolcache/windows/Python/3.14.4/x64/python3.exe (found version "3.14.4") found
components: Interpreter Development.Module Development.Embed
-- Configuring done (32.3s)
-- Generating done (0.1s)
-- Build files have been written to: D:/a/nanobind_json/nanobind_json/build
```

The second culprit revealed

```
-- The C compiler identification is MSVC 19.44.35226.0
-- The CXX compiler identification is MSVC 19.44.35226.0
-- Detecting C compiler ABI info
-- Detecting C compiler ABI info - done
-- Check for working C compiler: C:/Program Files/Microsoft Visual Studio/2022/Enterprise/VC/Tools/MSVC/14.44.35207/bin/Hostx64/x64/cl.exe - skipped
-- Detecting C compile features
-- Detecting C compile features - done
-- Detecting CXX compiler ABI info
-- Detecting CXX compiler ABI info - done
-- Check for working CXX compiler: C:/Program Files/Microsoft Visual Studio/2022/Enterprise/VC/Tools/MSVC/14.44.35207/bin/Hostx64/x64/cl.exe - skipped
-- Detecting CXX compile features
-- Detecting CXX compile features - done
-- Found Python: C:/Program Files/Microsoft Visual Studio/2022/Enterprise/VC/Tools/MSVC/14.44.35207/bin/Hostx64/x64/python3.exe (found suitable version "3.13.13", minimum required is "3.9") found components: Interpreter Development.Module Development.Embed
-- Detecting CXX pre-processor
-- Detecting CXX pre-processor - header code from D:/a/nanobind_json/nanobind_json/build/_deps/nanobind_json/include/nanobind_json/nanobind_json.hpp
-- Found Python: C:/Program Files/Microsoft Visual Studio/2022/Enterprise/VC/Tools/MSVC/14.44.35207/bin/Hostx64/x64/python3.exe (found version "3.14.4") found components: Interpreter Development Development.Module Development.Embed
-- Configuring done (32.3s)
-- Generating done (0.1s)
-- Build files have been written to: D:/a/nanobind_json/nanobind_json/build
```

The second culprit revealed

```
include(FetchContent)

FetchContent_Declare(
    nanobind
    GIT_REPOSITORY https://github.com/wjakob/nanobind.git
    GIT_TAG master
)

FetchContent_Declare(
    nlohmann_json
    GIT_REPOSITORY https://github.com/nlohmann/json.git
    GIT_TAG master
)

FetchContent_MakeAvailable(nanobind nlohmann_json)

include_directories(build/_deps/nlohmann_json-src/include)

set(nanobind_REQUIRED_VERSION 2.2.0)
set(nlohmann_json_REQUIRED_VERSION 3.2.0)

find_package(Python COMPONENTS Interpreter Development REQUIRED)
```

The End

The End

- ✓ Build and run tests on ubuntu-latest with Python 3.9
- ✓ Build and run tests on ubuntu-latest with Python 3.10
- ✓ Build and run tests on ubuntu-latest with Python 3.11
- ✓ Build and run tests on ubuntu-latest with Python 3.12
- ✓ Build and run tests on ubuntu-latest with Python 3.13
- ✓ Build and run tests on ubuntu-latest with Python 3.14

- ✓ Build and run tests on ubuntu-24.04-arm with Python 3.9
- ✓ Build and run tests on ubuntu-24.04-arm with Python 3.10
- ✓ Build and run tests on ubuntu-24.04-arm with Python 3.11
- ✓ Build and run tests on ubuntu-24.04-arm with Python 3.12
- ✓ Build and run tests on ubuntu-24.04-arm with Python 3.13
- ✓ Build and run tests on ubuntu-24.04-arm with Python 3.14

- ✓ Build and run tests on windows-latest with Python 3.9
- ✓ Build and run tests on windows-latest with Python 3.10
- ✓ Build and run tests on windows-latest with Python 3.11
- ✓ Build and run tests on windows-latest with Python 3.12
- ✓ Build and run tests on windows-latest with Python 3.13
- ✓ Build and run tests on windows-latest with Python 3.14

- ✓ Build and run tests on macos-latest with Python 3.9
- ✓ Build and run tests on macos-latest with Python 3.10
- ✓ Build and run tests on macos-latest with Python 3.11
- ✓ Build and run tests on macos-latest with Python 3.12
- ✓ Build and run tests on macos-latest with Python 3.13
- ✓ Build and run tests on macos-latest with Python 3.14

The End

Hi Gracjan,

this looks great -- could I ask you to make a PR in the nanobind repository to link to your code?

Best,
Wenzel

The End

Hi Gracjan,

this looks great -- could I ask you to make a PR in the nanobind repository to link to your code?

Best,
Wenzel

Is there a way to pass JSON objects between Python and C++? 🇵🇹

Yes, an unofficial, currently maintained, package supporting that can be found [here](#). It is based on a similar package for `pybind11` called `pybind11_json`.

The End

Hi Gracjan,

this looks great -- could I ask you to make a PR in the nanobind repository to link to your code?

Best,
Wenzel

Is there a way to pass JSON objects between Python and C++? 🐍

Yes, an unofficial, currently maintained, package supporting that can be found [here](#). It is based on a similar package for `pybind11` called `pybind11_json`.

https://github.com/Griger5/nanobind_json

The End (bonus!)

Ordered JSON support!

```
template <> struct type_caster<nlohmann::ordered_json> {  
public:  
    NB_TYPE_CASTER(nlohmann::ordered_json, const_name("OrderedJSON"));
```

support for `nlohmann::ordered_json` in addition to
`nlohmann::json` #63

Open



slayoo opened on May 18, 2023

Adding support for the `ordered_json` type in addition to currently used `json` would be beneficial for users intending to preserve order of elements in the JSON structures.

Help wanted! Contributions welcome!

Help wanted! Contributions welcome!

- If you find any bugs, feel free to create an issue or a PR!

Help wanted! Contributions welcome!

- If you find any bugs, feel free to create an issue or a PR!
- The library only supports nlohmann/json, it would be awesome to also include: **simdjson**, **Boost.JSON**, **picojson** and others!

Fun facts behind the scenes

Fun facts behind the scenes

```
# TODO: Run on other Python versions on Windows (Missing/not-matching DLLs error)
- name: Run tests
  if: ${ startsWith(matrix.os, 'windows') && matrix.python == '3.13' }
  run: |
    cd build/Release
    python -m pytest ../../tests
```

Fun facts behind the scenes

Throwaway: Pykonik84 testing #5 

 Draft

Griger5 wants to merge 40 commits into `ianhbell:main` from `Griger5:throwaway/pykonik84`



Fun facts behind the scenes

Throwaway: Pykonik84 testing #5

 Draft

[Griger5](#) wants to merge 40 commits into [ianhbell:main](#) from [Griger5:throwaway/pykonik84](#) 

 2 Open ✓ 3 Closed

Throwaway: Pykonik84 testing

#5 by Griger5 was closed now • Draft

Remove unnecessary semicolons

#4 by Griger5 was closed on Mar 8

Fix redundant move and semicolons

#3 by Griger5 was closed on Mar 2

Fun facts behind the scenes

The screenshot shows the GitHub repository page for `nanobind_json` by user `ianhbell`. The repository is public and has 3 watchers and 4 forks. The main branch is selected, with 1 branch and 0 tags. The file list shows the following files and their commit history:

File	Commit History	Time
<code>.github/workflows</code>	First commit	2 years ago
<code>include/nanobind_json</code>	First commit	2 years ago
<code>test</code>	First commit	2 years ago
<code>.gitignore</code>	First commit	2 years ago
<code>CMakeLists.txt</code>	First commit	2 years ago
<code>LICENSE</code>	First commit	2 years ago
<code>README.md</code>	Update verbiage	2 years ago
<code>nanobind_jsonConfig.cmake.in</code>	First commit	2 years ago
<code>pixi.lock</code>	First commit	2 years ago

The repository also includes a sidebar with the following information:

- About:** nanobind wrapper for nanlohmann::json library
- Readme:** BSD-3-Clause license
- Activity:** 11 stars, 3 watching, 4 forks
- Releases:** No releases published
- Packages:** (empty)

Fun facts behind the scenes

ianhbell / nanobind_json

Code Issues Pull requests 1 Agents Actions Projects Security and quality Insights

nanobind_json Public

Watch 3 Fork 4

Inspector Console Debugger Network Style Editor Performance Memory Storage Accessibility Application

HTML

```
<li class="prc-UnderlineNav-UnderlineNavItem-syRjR"></li> (flex)
  <li class="prc-UnderlineNav-UnderlineNavItem-syRjR"> (flex)
    <a class="prc-components-UnderlineItem-7fP-n" href="/ianhbell/nanobind_json/pulls" data-tab-item="pull-requests" data-turbo-frame="repo-content-turbo-frame" data-discover="true"> (event) (flex)
      <span data-component="icon"></span> (flex)
      <span data-component="text" data-content="Pull requests"></span>
      <span data-component="counter"> (flex)
        <span class="prc-CounterLabel-CounterLabel-X-kRU" aria-hidden="true" data-variant="secondary">1</span>
        <span class="prc-VisuallyHidden-VisuallyHidden-Q0qSB"></span>
      </span>
      ::after
    </a>
  </li>
</ul>
</nav>
<div class="d-none"></div>
<script id="__PRIMER_DATA_R_0__" type="application/json">{"resolvedServerColorMode":"night"}</script>
</div>
</react-partial>
```

Filter Styles

element :: { }

@layer primer-react { CounterLabel.module .prc-CounterLabel-CounterLabel-X-kRU:where([data-variant="secondary"]) :: { background-color: var(--bgColor-neutral muted #010b081f); color: var(--fgColor-default #fff); }

@layer primer-react { CounterLabel.module .prc-CounterLabel-CounterLabel-X-kRU :: { border: var(--borderWidth-thin #fff solid var(--counter-borderColor #fff)); font-size: var(--text-body-size-small #75rem); font-weight: var(--base-text-weight-semi #600); padding: var(--base-size-2 #125rem base-size-6 #375rem); border-radius: 20px; line-height: 1;

Fun facts behind the scenes

**Build and run tests
on windows-lates...**

Started 2h 24m 5s ago

 Search logs



Setup tmate session

2h 22m 10s

Fun facts behind the scenes

- My first time actually using git cherry-pick!
- I wanted to quickly present the library as a Lightning Talk on Pykonik 79, my laptop decided to not cooperate with HDMI

Special thanks

- PyPartMC developers (big shoutout to Sylwester Arabas (AGH), Nicole Riemer (UIUC) and Jeff Curtis (UIUC))
- Max Schmitt (tmate-session Github Action)
- Peter Arzt (PR fixing nanobind_json)
- Ian Bell (original nanobind_json)
- Aleksander Wadelski (moral support)
- Kacper Wojtowicz (moral suffering)
- pybind11_json team (for the library blueprints)
- Wenzel Jakob (creation of pybind11 and nanobind)

Thank you!