C++: Python bindings

Pallini Andrea

About me

About Leica Geosystems

Leica Geosystems

Road map of the presentation

Introduction to python bindings

Examples of python bindings

Setup for integration of python in Leica devices

Examples of python bindings in Leica

25 minutes

Python bindings

Why extending python with C++: for python developers

Performance

Wrapping existing libraries

Integrations

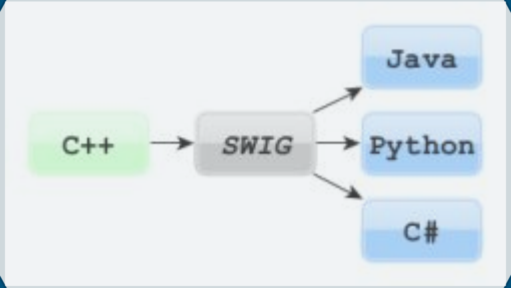
Why embedding python with C++: for C++ developers

Add python scripting in your app

Python-C++ extensions

Python code
- Ctypes

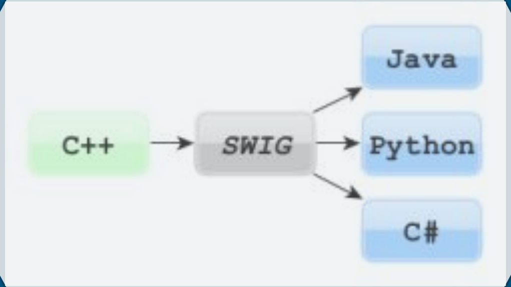
C C++ code
- Python C API (CPython)
- Pybind11
- Boost.Python



Python-C++ extensions

Python code
- Ctypes

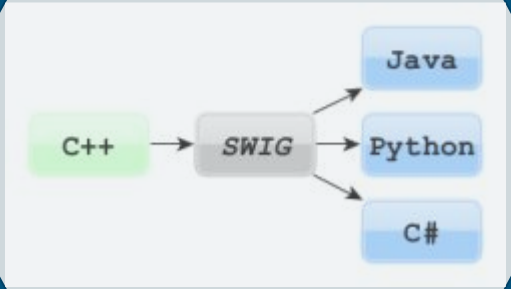
C C++ code
- Python C API (CPython)
- Pybind11
- Boost.Python



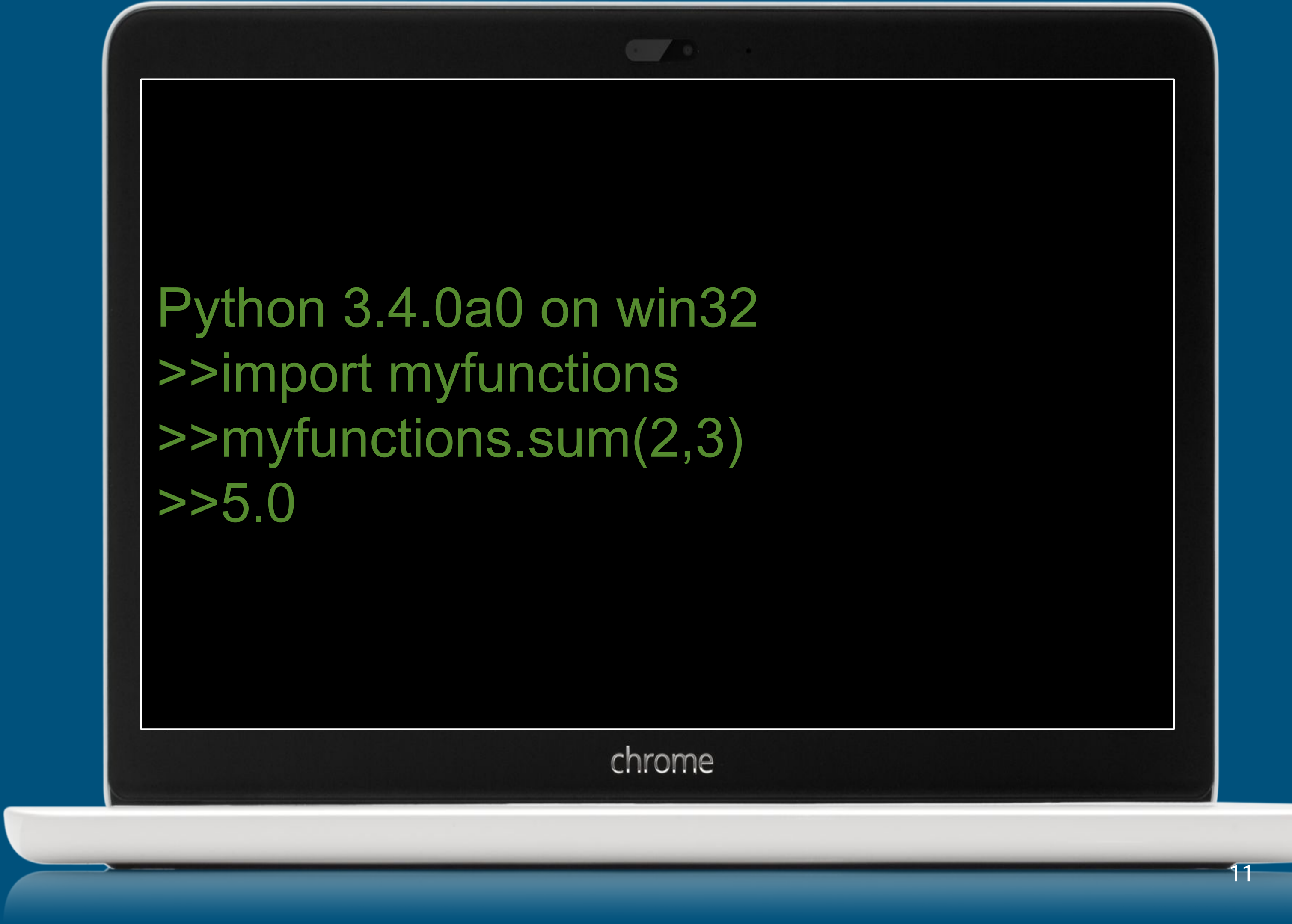
Python-C++ extensions

Python code
- Ctypes

C C++ code
- Python C API (CPython)
- Pybind11
- Boost.Python

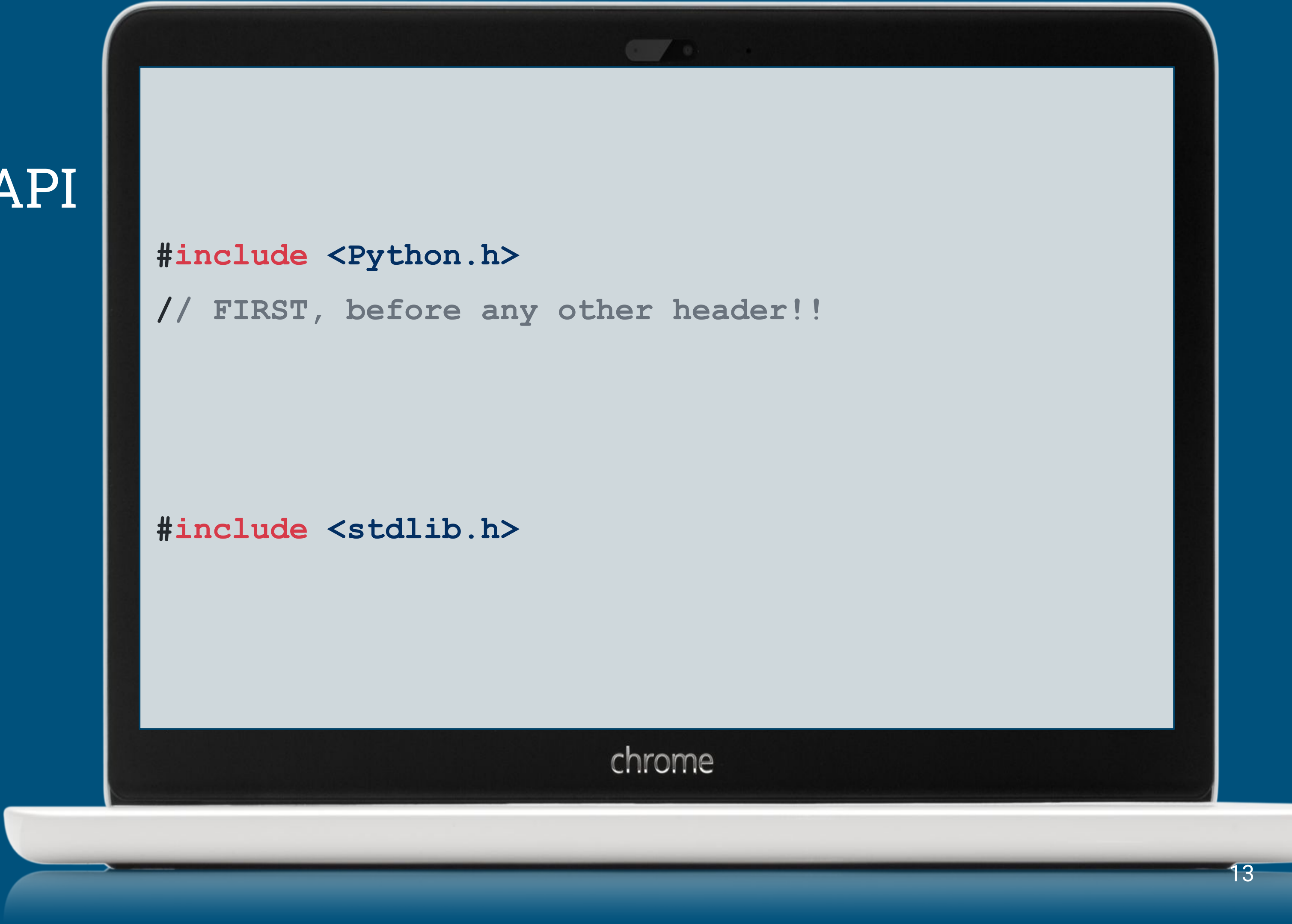


Goal


```
Python 3.4.0a0 on win32
>>import myfunctions
>>myfunctions.sum(2,3)
>>5.0
```

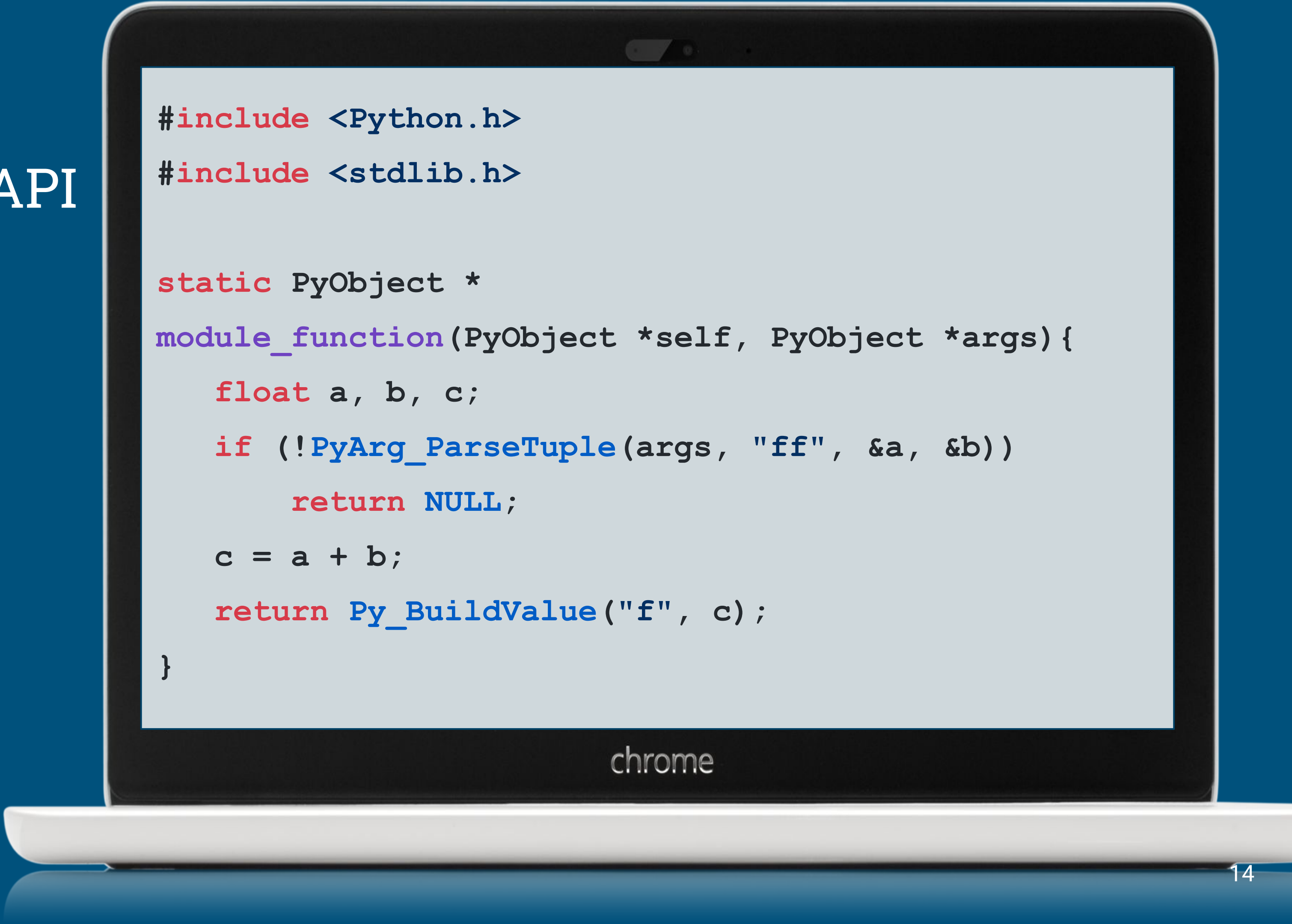
Approach 1: C-API

Python C API



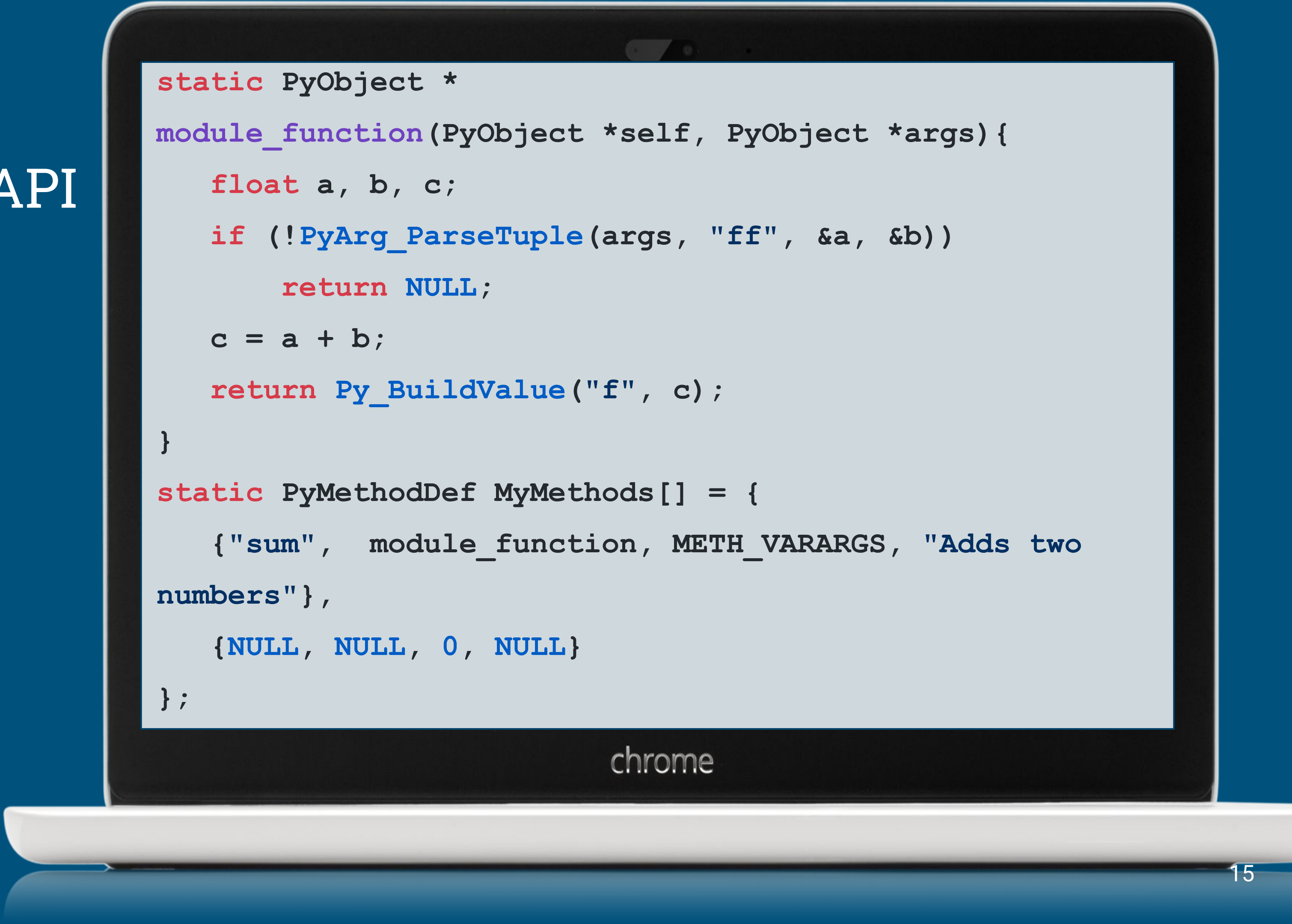
```c
#include <Python.h>
// FIRST, before any other header!!



#include <stdlib.h>
```

Python C API



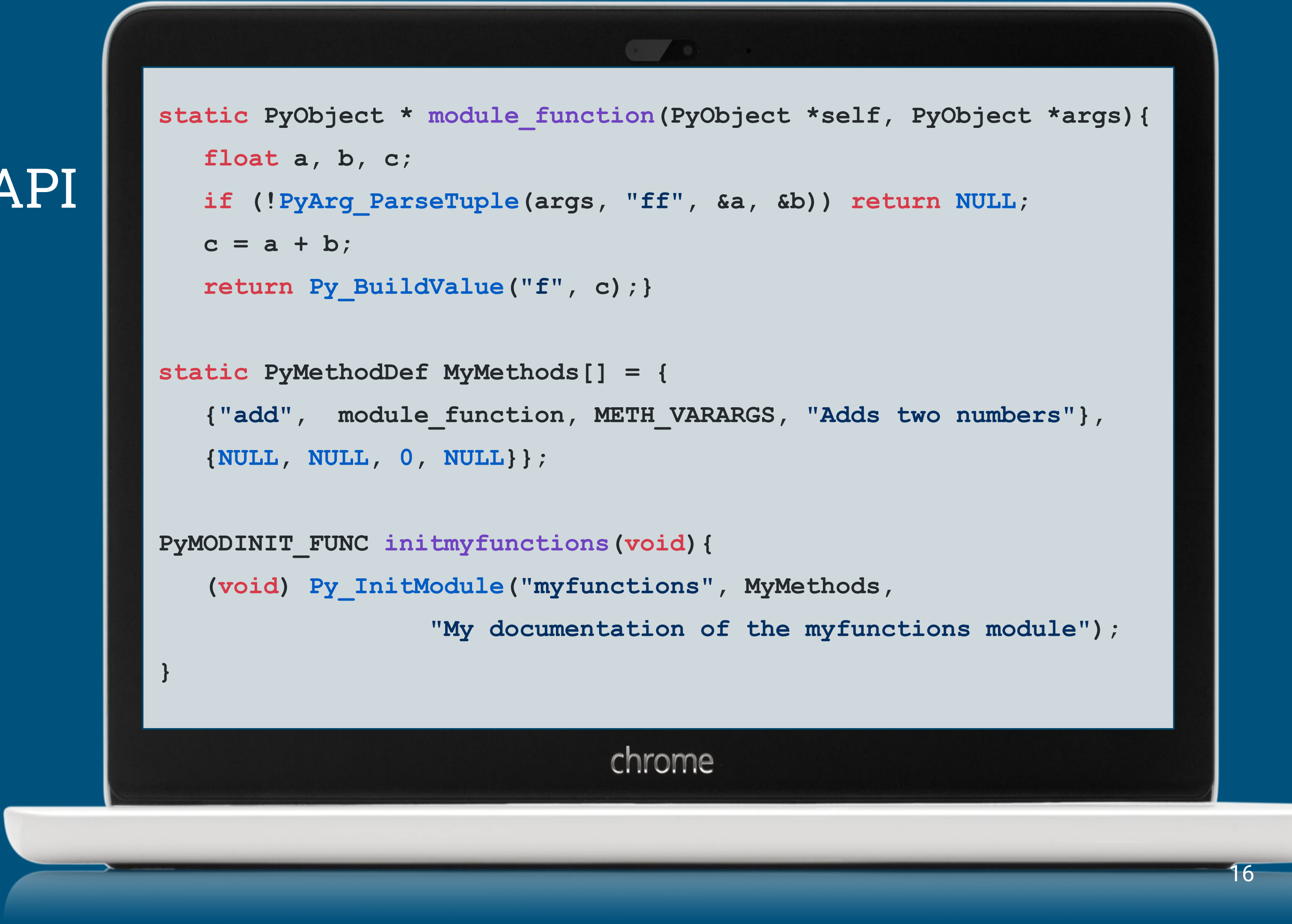
```
#include <Python.h>
#include <stdlib.h>

static PyObject *
module_function(PyObject *self, PyObject *args){
    float a, b, c;
    if (!PyArg_ParseTuple(args, "ff", &a, &b))
        return NULL;
    c = a + b;
    return Py_BuildValue("f", c);
}
```

Python C API



```
static PyObject *
module_function(PyObject *self, PyObject *args){
    float a, b, c;
    if (!PyArg_ParseTuple(args, "ff", &a, &b))
        return NULL;
    c = a + b;
    return Py_BuildValue("f", c);
}
static PyMethodDef MyMethods[] = {
    {"sum",  module_function, METH_VARARGS, "Adds two
numbers"},
    {NULL, NULL, 0, NULL}
};
```

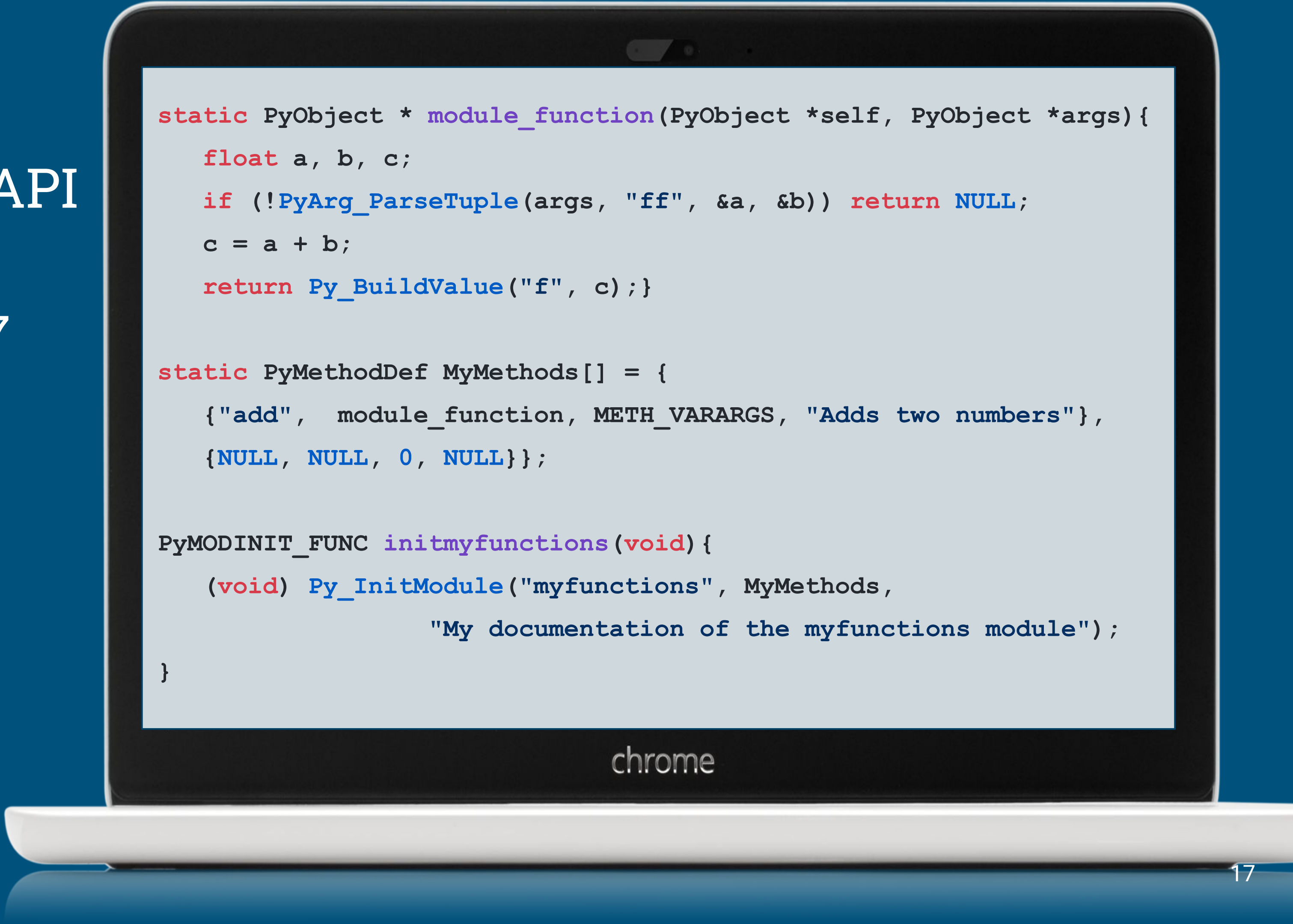
Python C API



```
static PyObject * module_function(PyObject *self, PyObject *args){
   float a, b, c;
   if (!PyArg_ParseTuple(args, "ff", &a, &b)) return NULL;
   c = a + b;
   return Py_BuildValue("f", c);}

static PyMethodDef MyMethods[] = {
   {"add",  module_function, METH_VARARGS, "Adds two numbers"},
   {NULL, NULL, 0, NULL}};

PyMODINIT_FUNC initmyfunctions(void){
   (void) Py_InitModule("myfunctions", MyMethods,
                  "My documentation of the myfunctions module");
}
```

Python C API

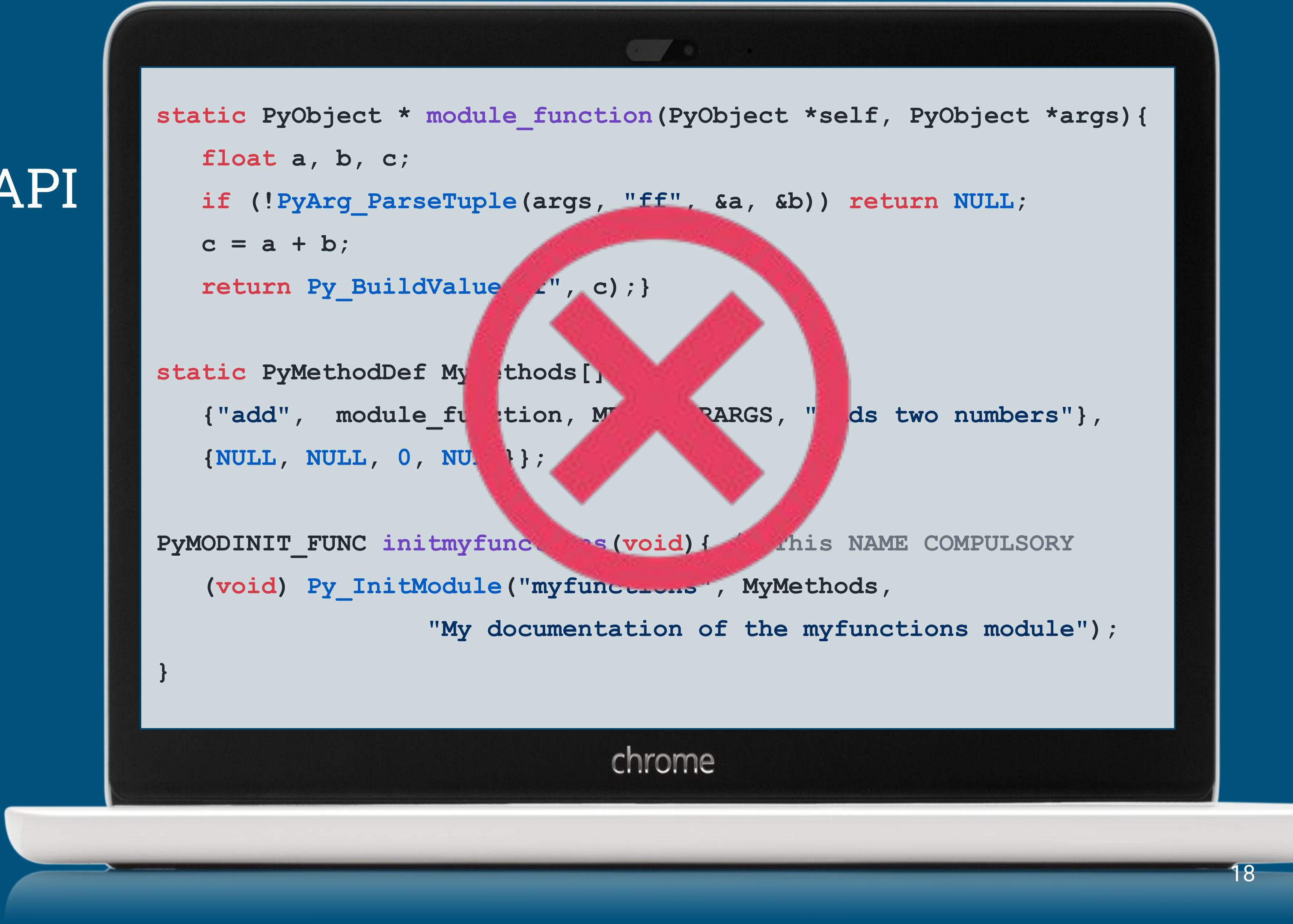
Python 2.7 API



```
static PyObject * module_function(PyObject *self, PyObject *args){
   float a, b, c;
   if (!PyArg_ParseTuple(args, "ff", &a, &b)) return NULL;
   c = a + b;
   return Py_BuildValue("f", c);}

static PyMethodDef MyMethods[] = {
   {"add",  module_function, METH_VARARGS, "Adds two numbers"},
   {NULL, NULL, 0, NULL}};

PyMODINIT_FUNC initmyfunctions(void){
   (void) Py_InitModule("myfunctions", MyMethods,
                    "My documentation of the myfunctions module");
}
```

Python C API

Python 3.X API



```
static PyObject * module_function(PyObject *self, PyObject *args){
   float a, b, c;
   if (!PyArg_ParseTuple(args, "ff", &a, &b)) return NULL;
   c = a + b;
   return Py_BuildValue[illegible]", c);}

static PyMethodDef My[illegible]ethods[]
   {"add",  module_fu[illegible]ction, M[illegible]ARGS, "[illegible]ds two numbers"},
   {NULL, NULL, 0, NU[illegible]}};

PyMODINIT_FUNC initmyfunc[illegible]s(void){ [illegible]his NAME COMPULSORY
   (void) Py_InitModule("myfunc[illegible]", MyMethods,
                  "My documentation of the myfunctions module");
}
```

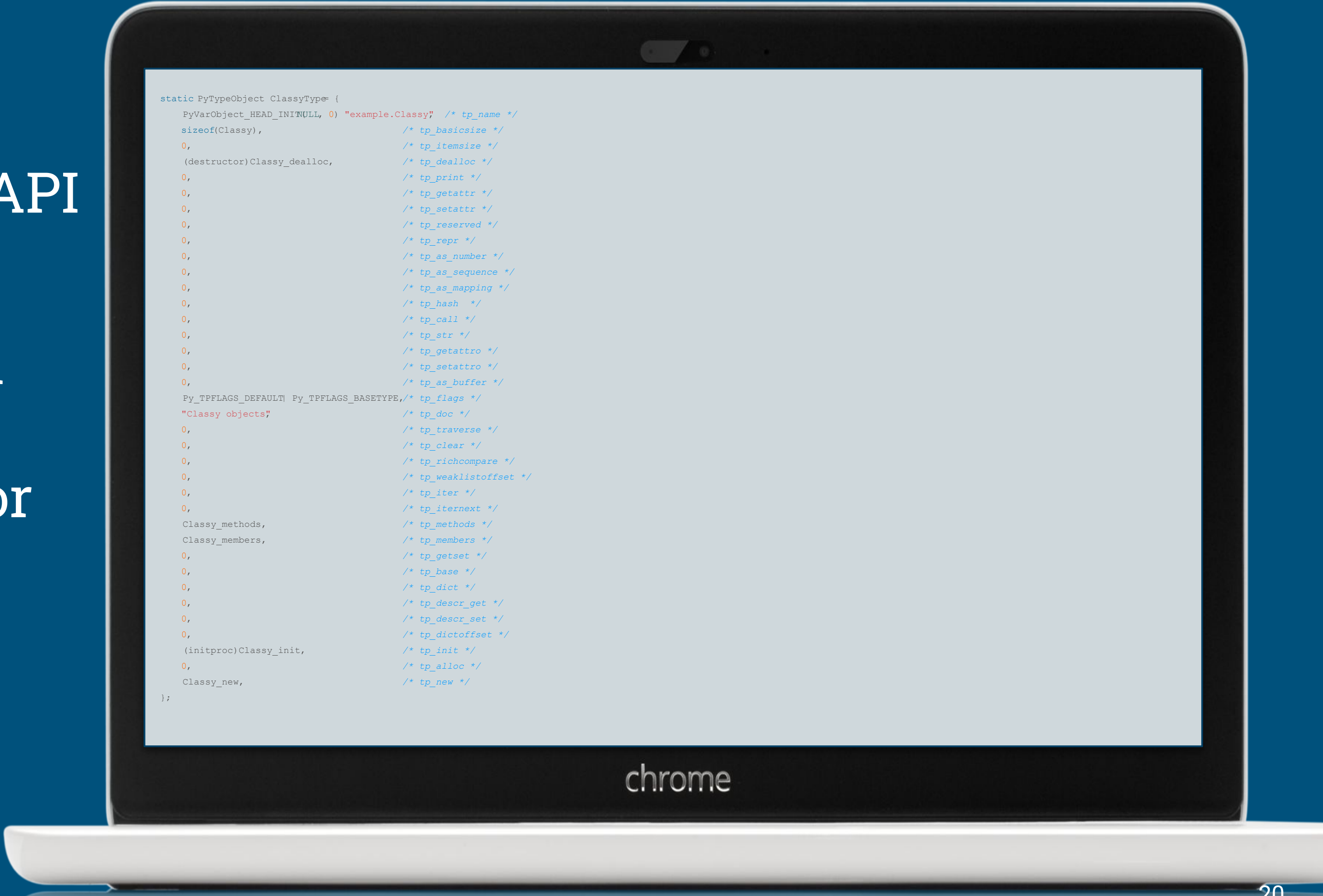
Python C API

Drawbacks

- Changes in API
- Creation of objects is complicated
- Emulation of constructor

Python C API

Emulation of constructor



```c
static PyTypeObject ClassyType = {
    PyVarObject_HEAD_INIT(NULL, 0) "example.Classy",  /* tp_name */
    sizeof(Classy),                            /* tp_basicsize */
    0,                                         /* tp_itemsize */
    (destructor)Classy_dealloc,                /* tp_dealloc */
    0,                                         /* tp_print */
    0,                                         /* tp_getattr */
    0,                                         /* tp_setattr */
    0,                                         /* tp_reserved */
    0,                                         /* tp_repr */
    0,                                         /* tp_as_number */
    0,                                         /* tp_as_sequence */
    0,                                         /* tp_as_mapping */
    0,                                         /* tp_hash  */
    0,                                         /* tp_call */
    0,                                         /* tp_str */
    0,                                         /* tp_getattro */
    0,                                         /* tp_setattro */
    0,                                         /* tp_as_buffer */
    Py_TPFLAGS_DEFAULT | Py_TPFLAGS_BASETYPE,/* tp_flags */
    "Classy objects",                          /* tp_doc */
    0,                                         /* tp_traverse */
    0,                                         /* tp_clear */
    0,                                         /* tp_richcompare */
    0,                                         /* tp_weaklistoffset */
    0,                                         /* tp_iter */
    0,                                         /* tp_iternext */
    Classy_methods,                            /* tp_methods */
    Classy_members,                            /* tp_members */
    0,                                         /* tp_getset */
    0,                                         /* tp_base */
    0,                                         /* tp_dict */
    0,                                         /* tp_descr_get */
    0,                                         /* tp_descr_set */
    0,                                         /* tp_dictoffset */
    (initproc)Classy_init,                     /* tp_init */
    0,                                         /* tp_alloc */
    Classy_new,                                /* tp_new */
};
```

https://www.hardikp.com/2017/12/30/python-cpp/

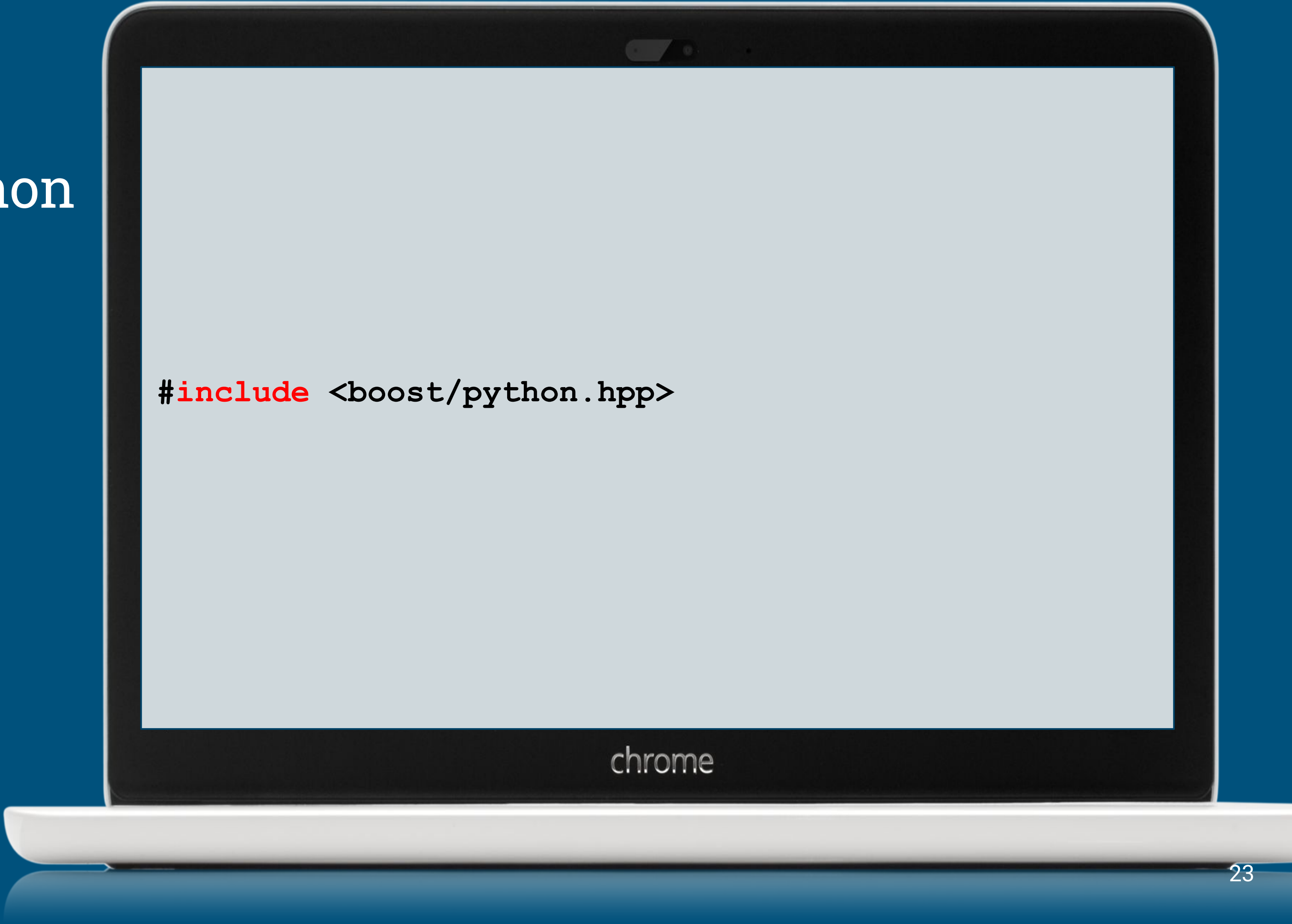
Approach 2: boost python

BOOST

https://www.boost.org/

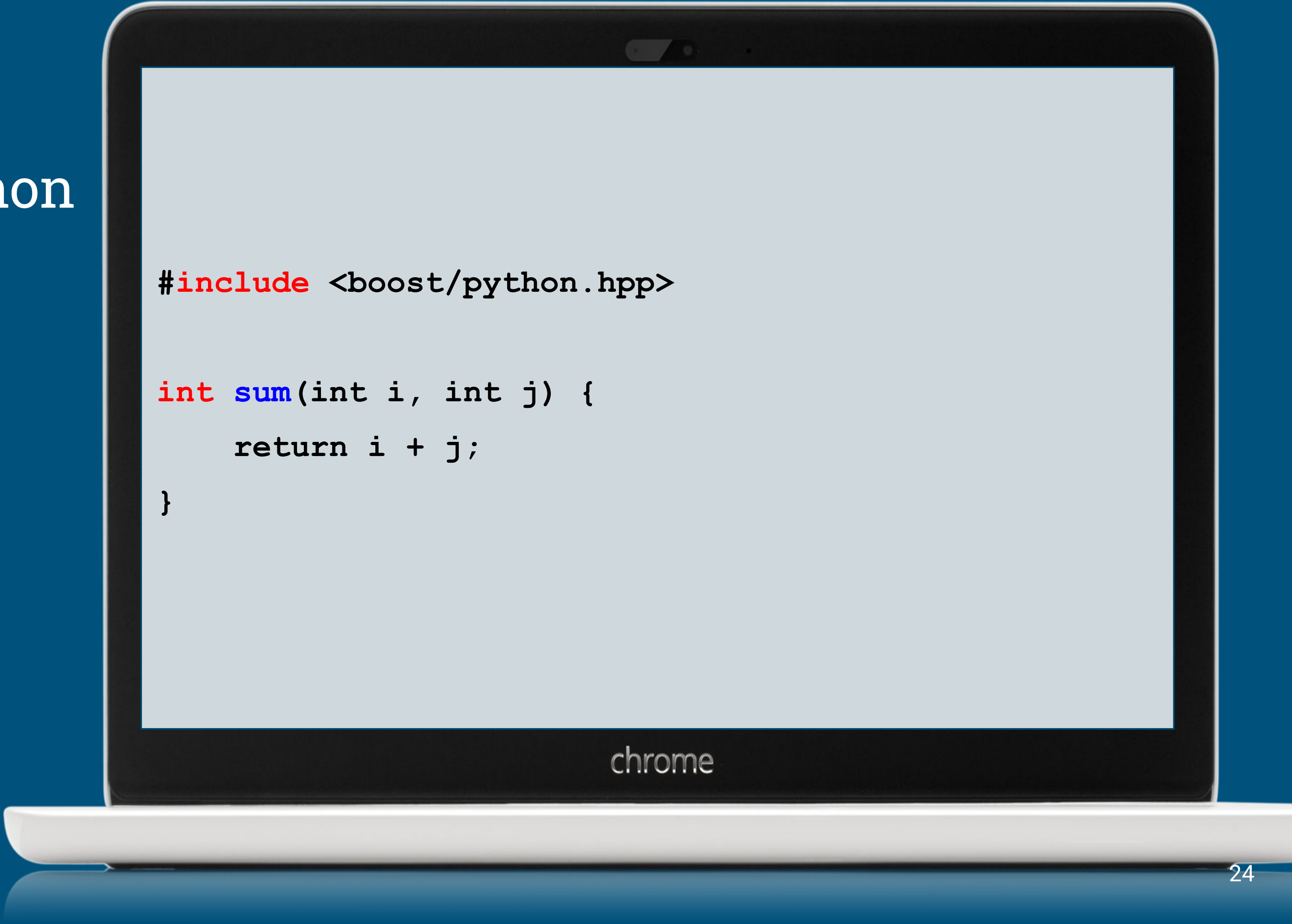
https://theboostcpplibraries.com/

Boost python



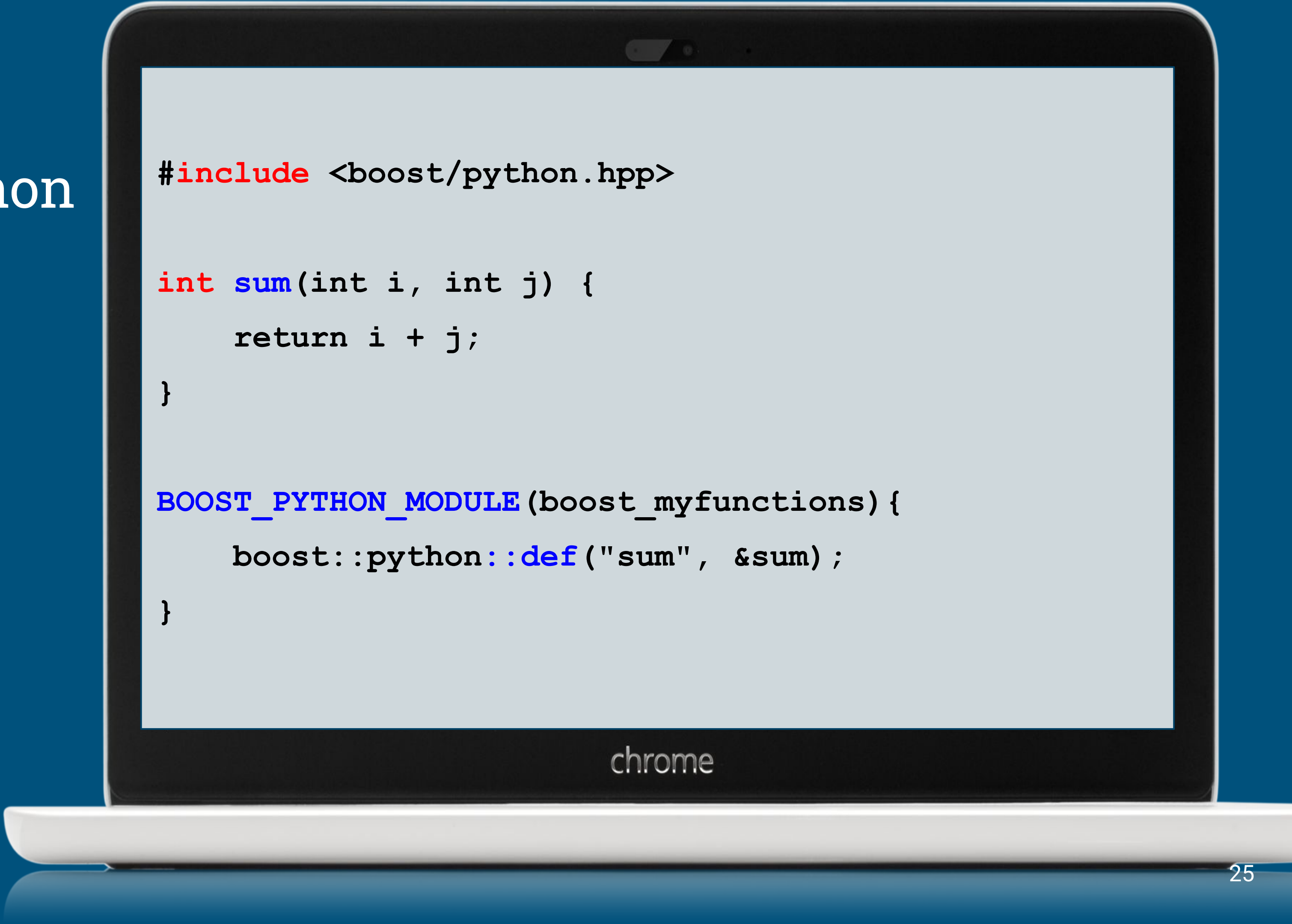
```
#include <boost/python.hpp>
```

Boost python



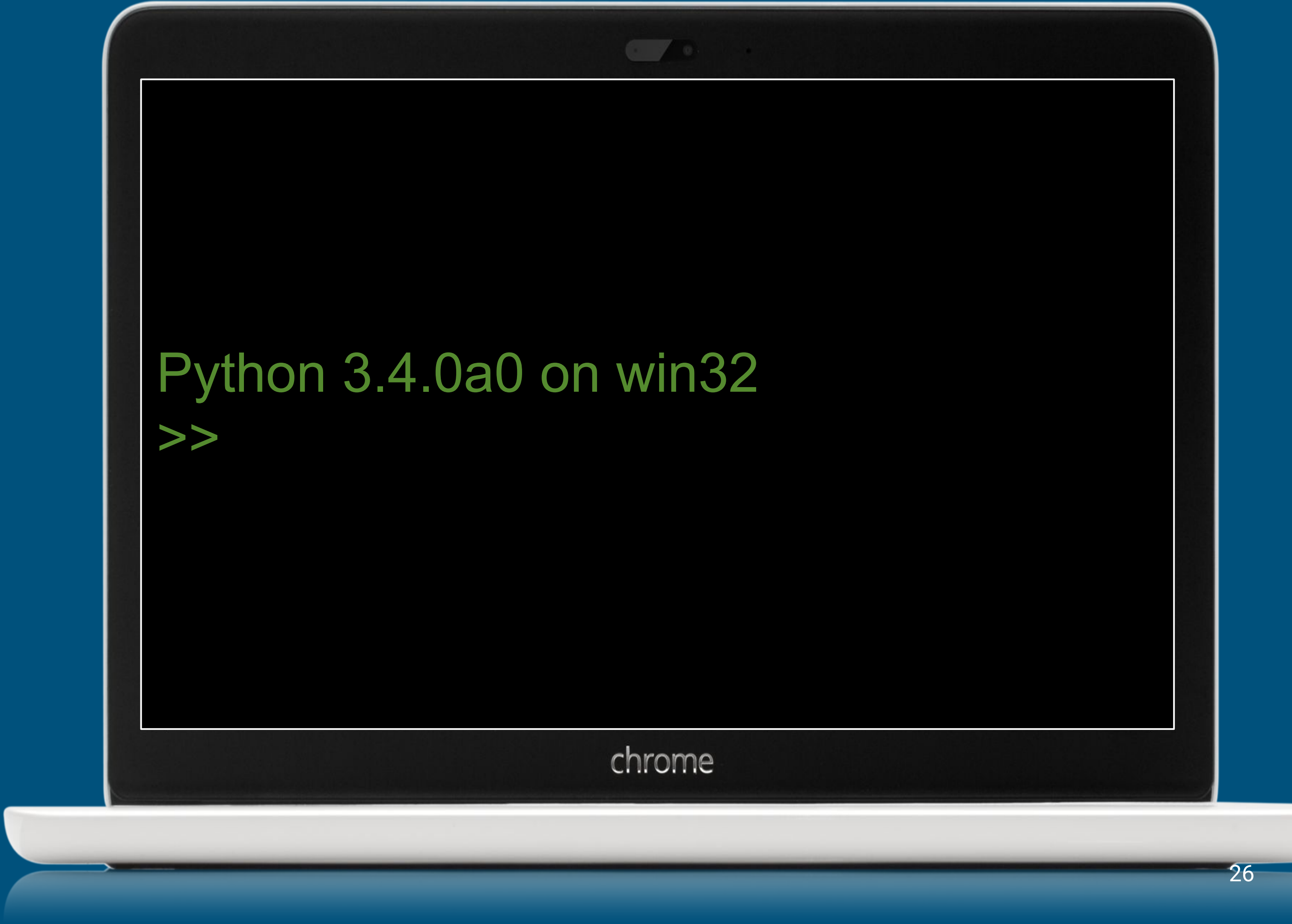
```
#include <boost/python.hpp>

int sum(int i, int j) {
    return i + j;
}
```

Boost python



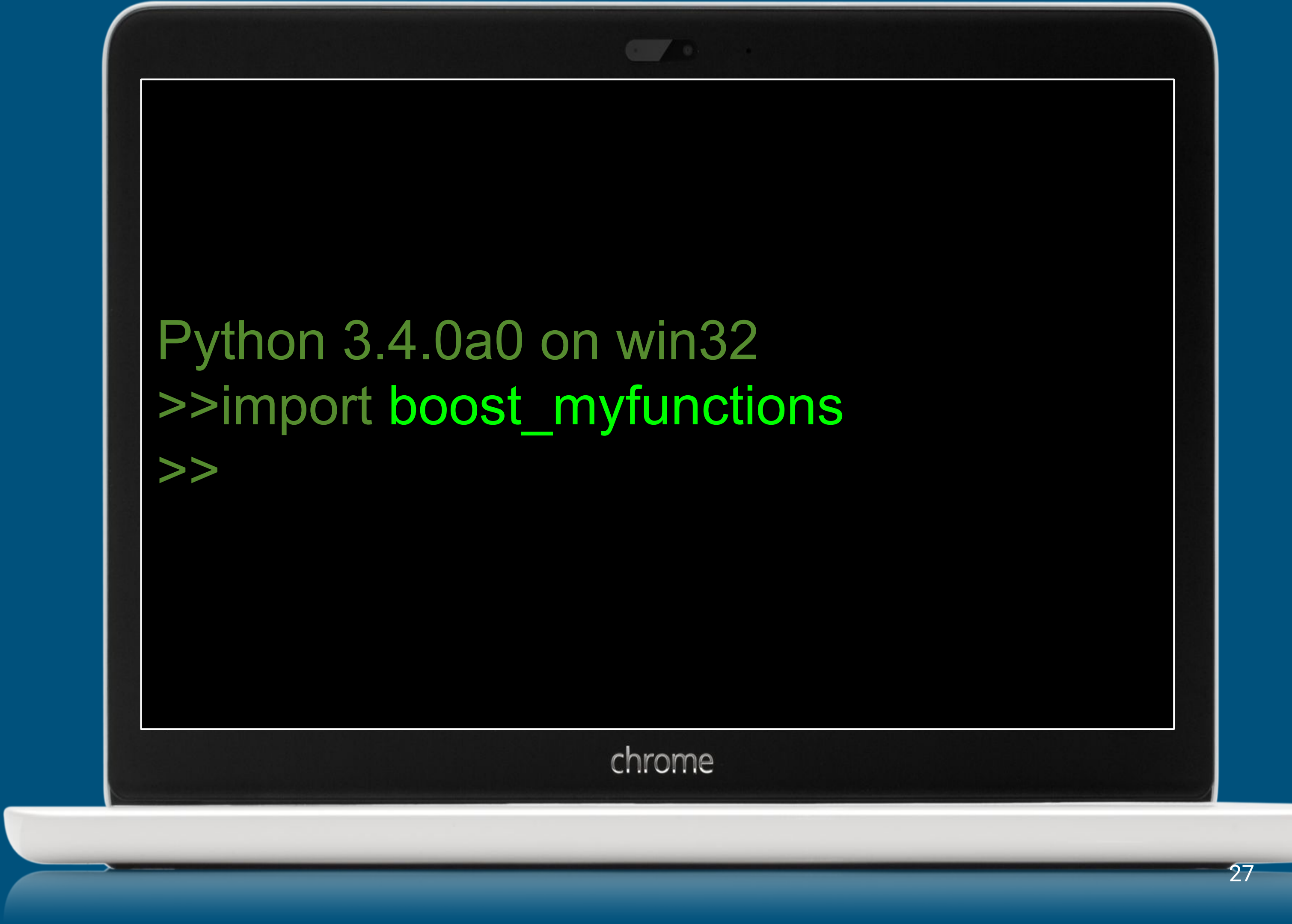
```
#include <boost/python.hpp>

int sum(int i, int j) {
    return i + j;
}

BOOST_PYTHON_MODULE(boost_myfunctions){
    boost::python::def("sum", &sum);
}
```

Goal

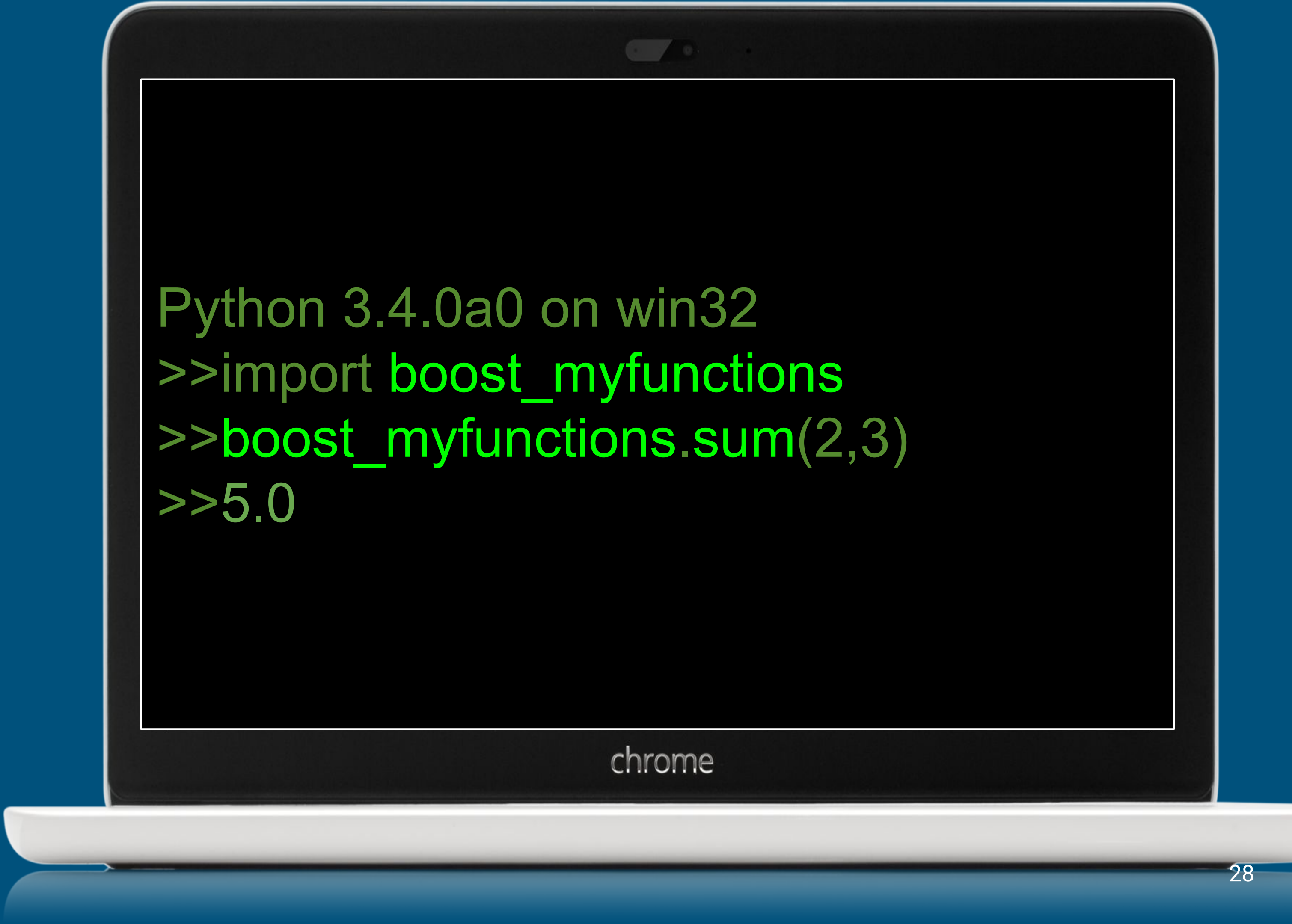


```
Python 3.4.0a0 on win32
>>
```

Goal


```
Python 3.4.0a0 on win32
>>import boost_myfunctions
>>
```

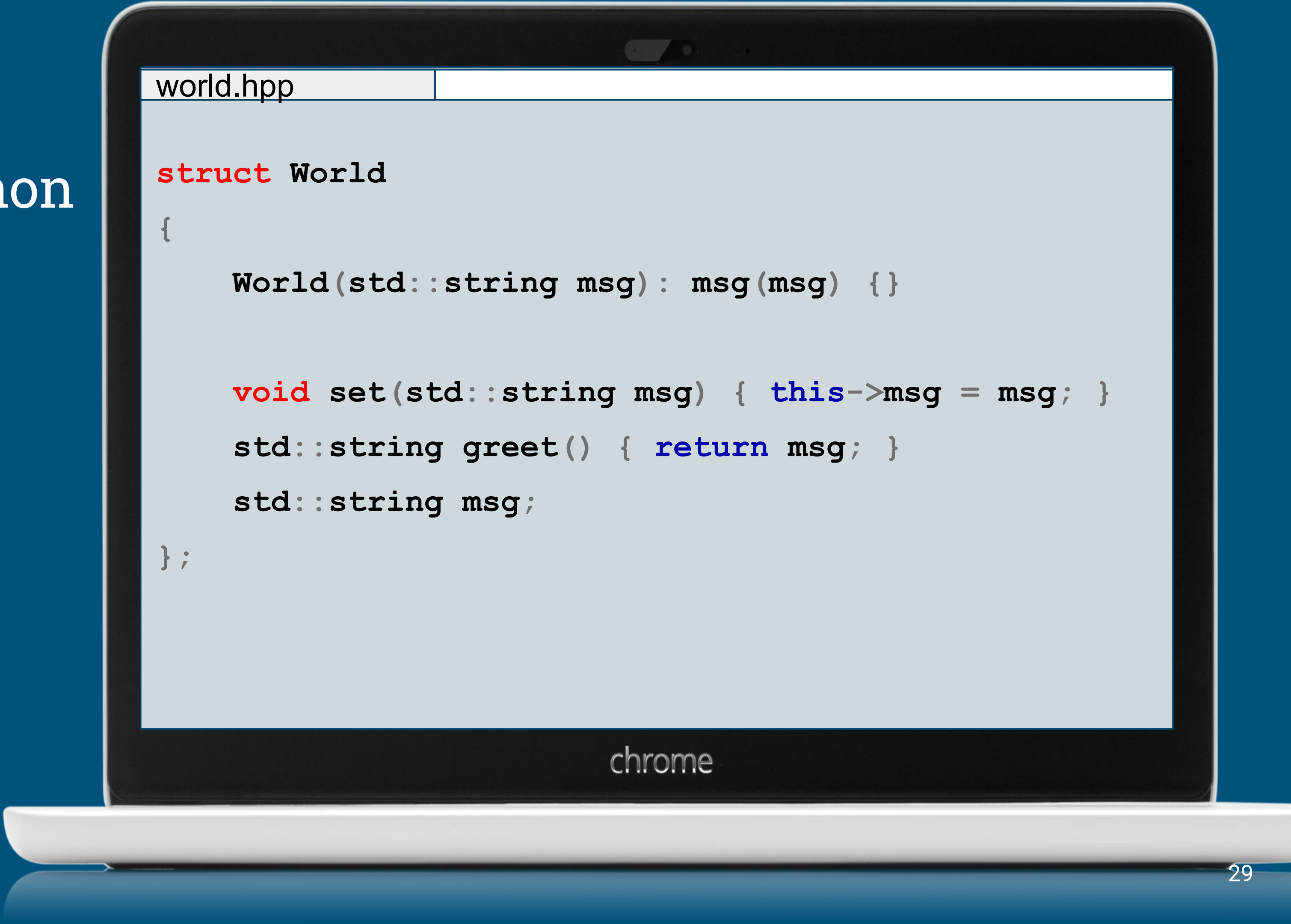
Goal



```
Python 3.4.0a0 on win32
>>import boost_myfunctions
>>boost_myfunctions.sum(2,3)
>>5.0
```

Boost python

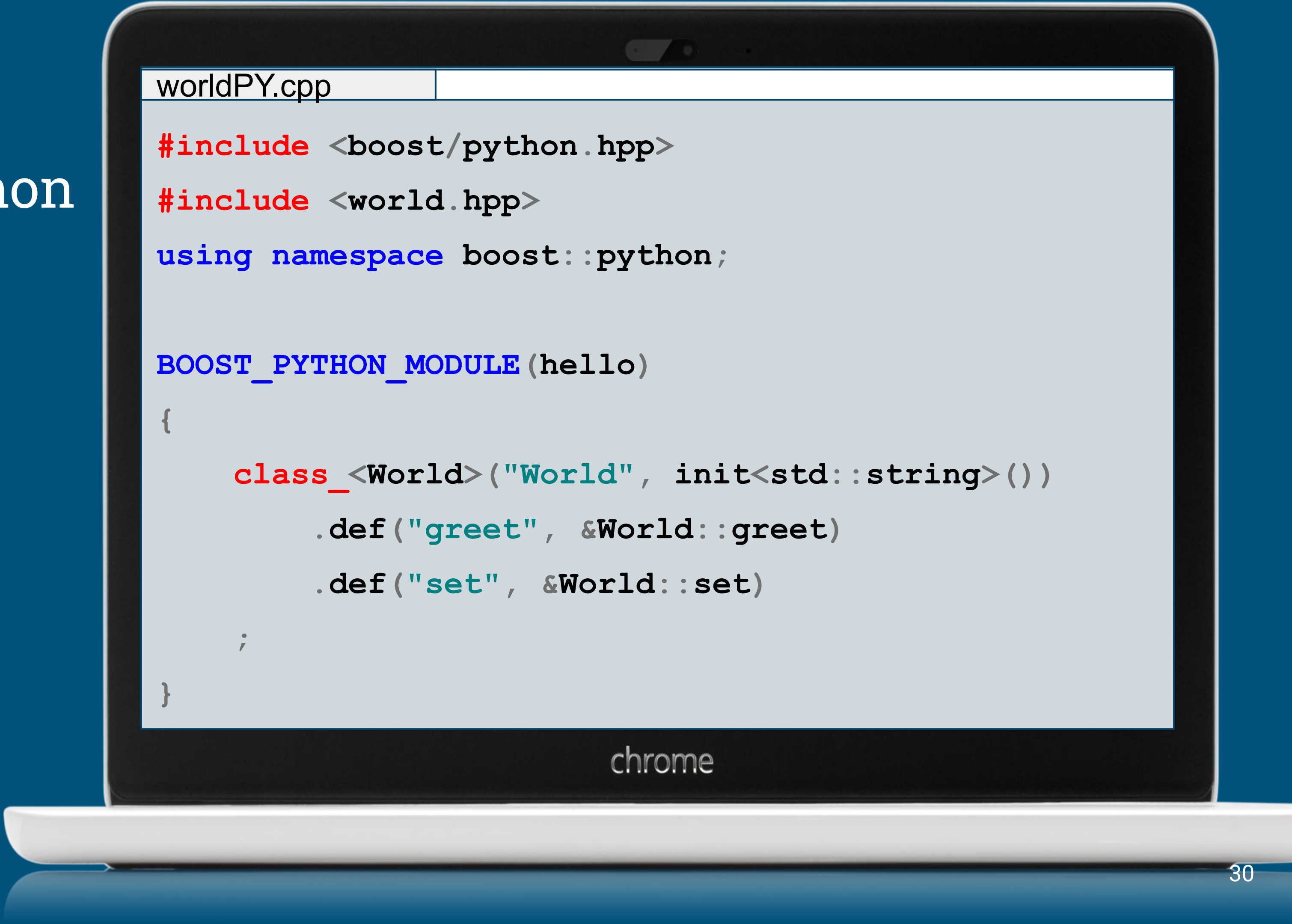
Export a class


world.hpp

```
struct World
{
    World(std::string msg): msg(msg) {}

    void set(std::string msg) { this->msg = msg; }
    std::string greet() { return msg; }
    std::string msg;
};
```

Boost python

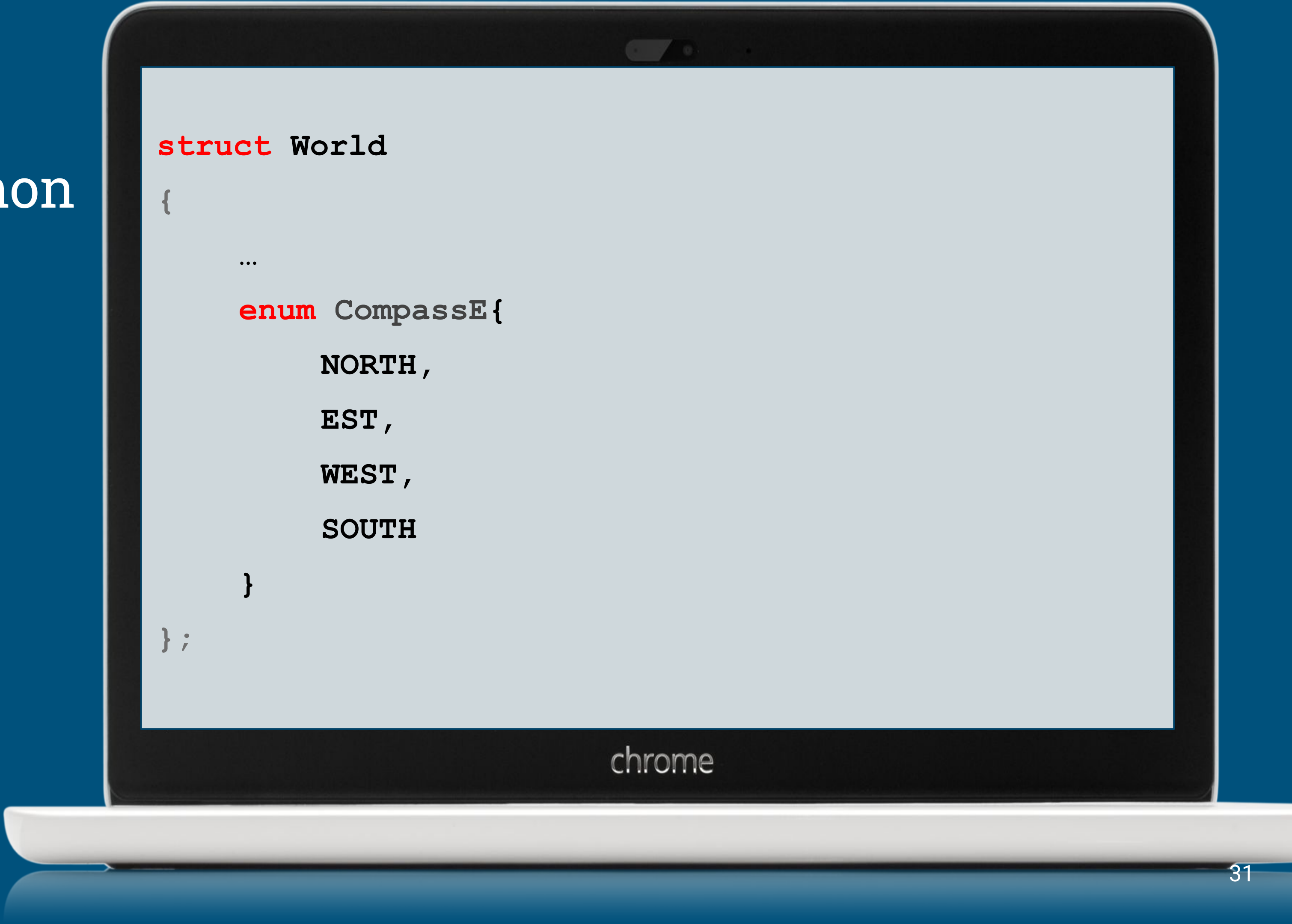
Export a class


worldPY.cpp

```cpp
#include <boost/python.hpp>
#include <world.hpp>
using namespace boost::python;

BOOST_PYTHON_MODULE(hello)
{
    class_<World>("World", init<std::string>())
        .def("greet", &World::greet)
        .def("set", &World::set)
    ;
}
```

Boost python

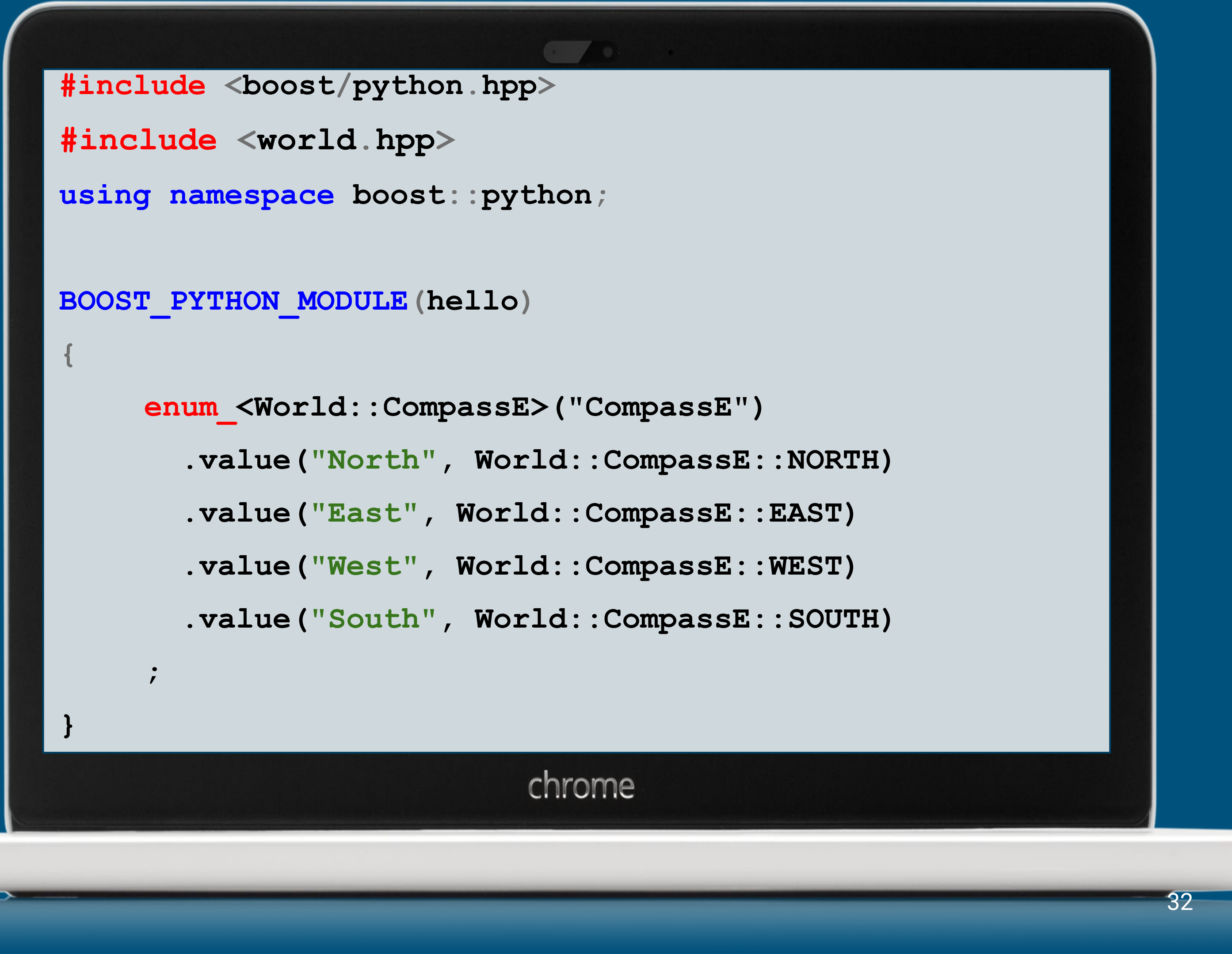
ENUM



```
struct World
{
    …
    enum CompassE{
        NORTH,
        EST,
        WEST,
        SOUTH
    }
};
```

Boost python

ENUM

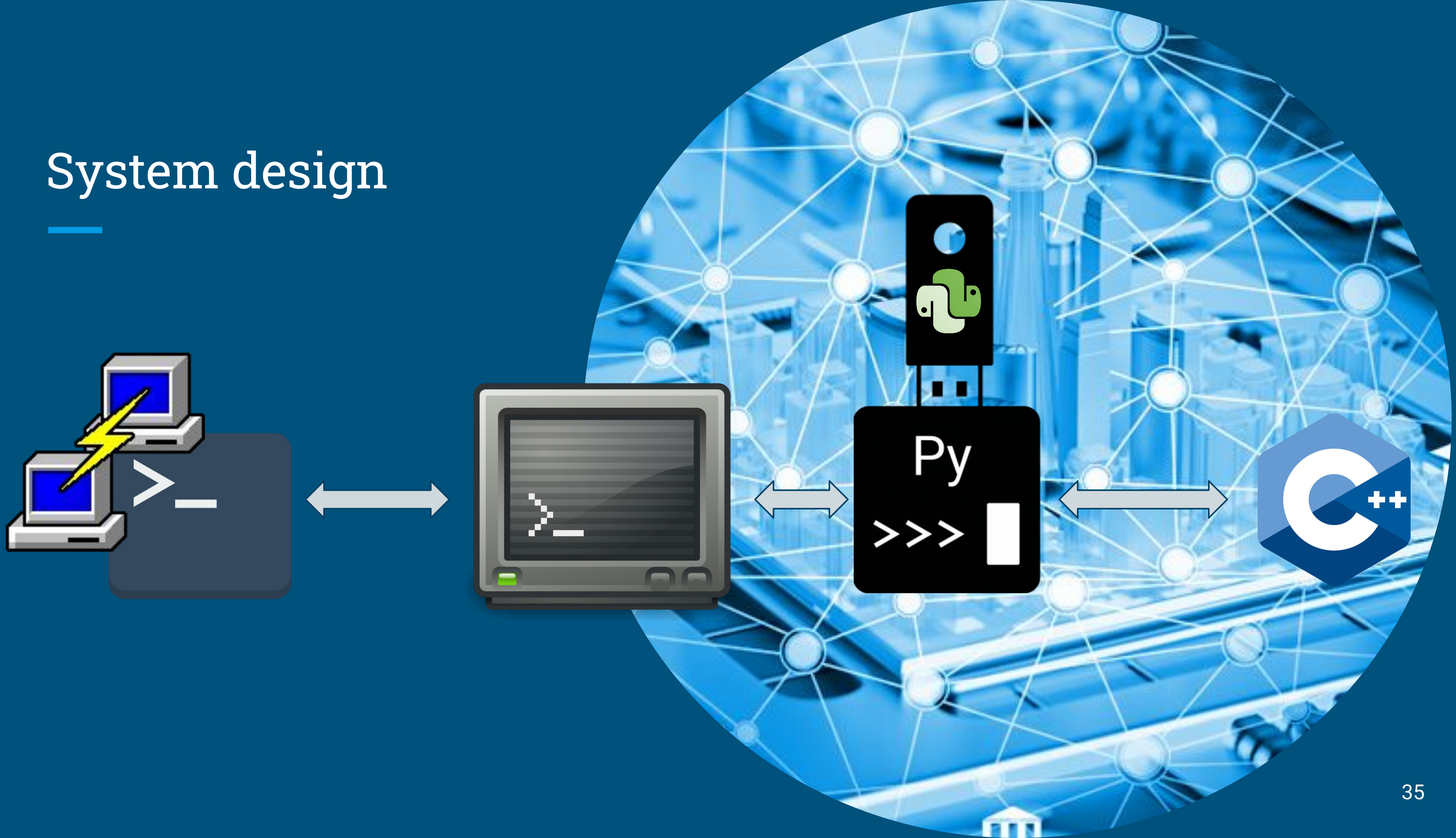


```
#include <boost/python.hpp>
#include <world.hpp>
using namespace boost::python;

BOOST_PYTHON_MODULE(hello)
{
    enum_<World::CompassE>("CompassE")
      .value("North", World::CompassE::NORTH)
      .value("East", World::CompassE::EAST)
      .value("West", World::CompassE::WEST)
      .value("South", World::CompassE::SOUTH)
    ;
}
```

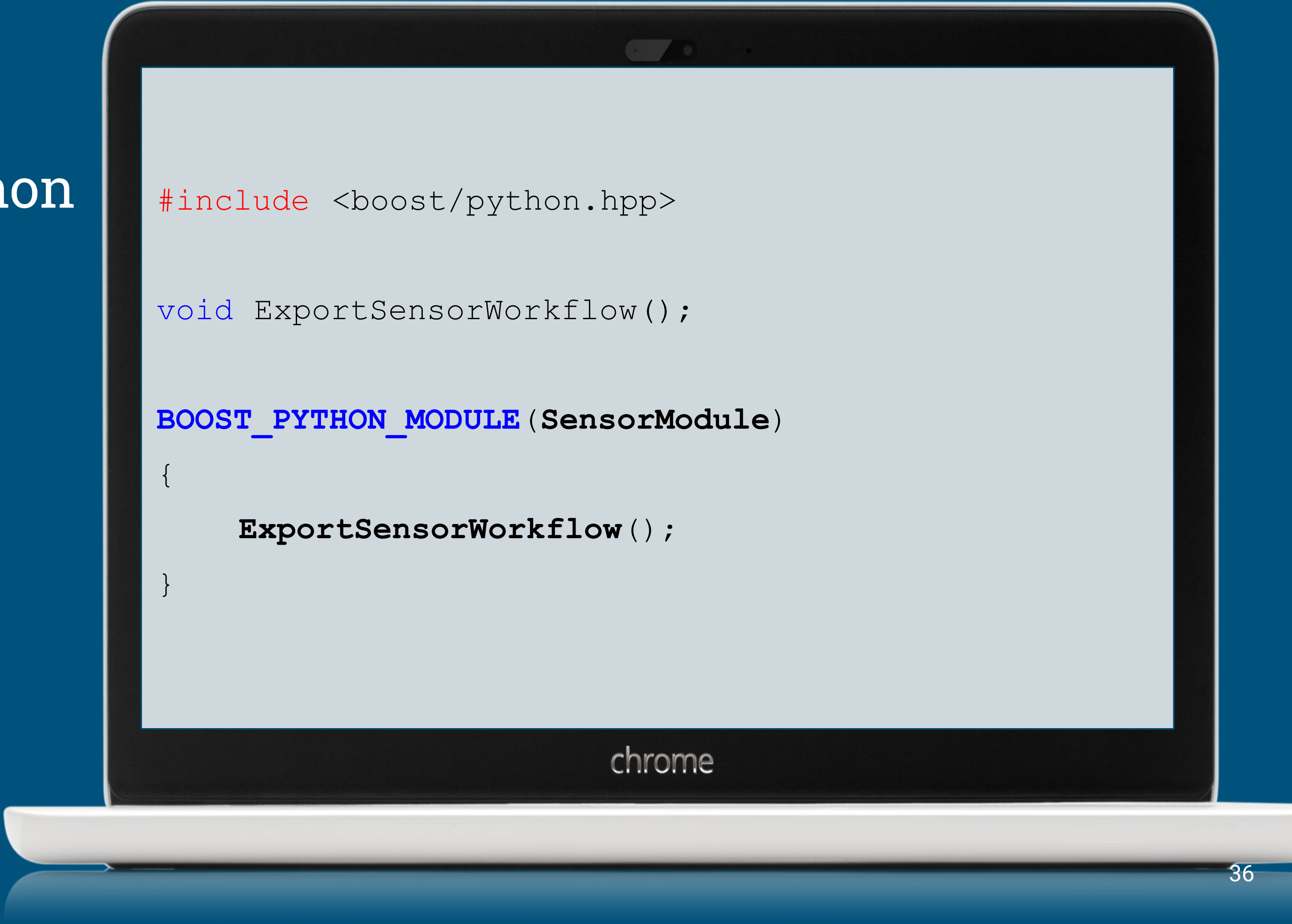
Boost python at Leica

Boost python @Leica

System design

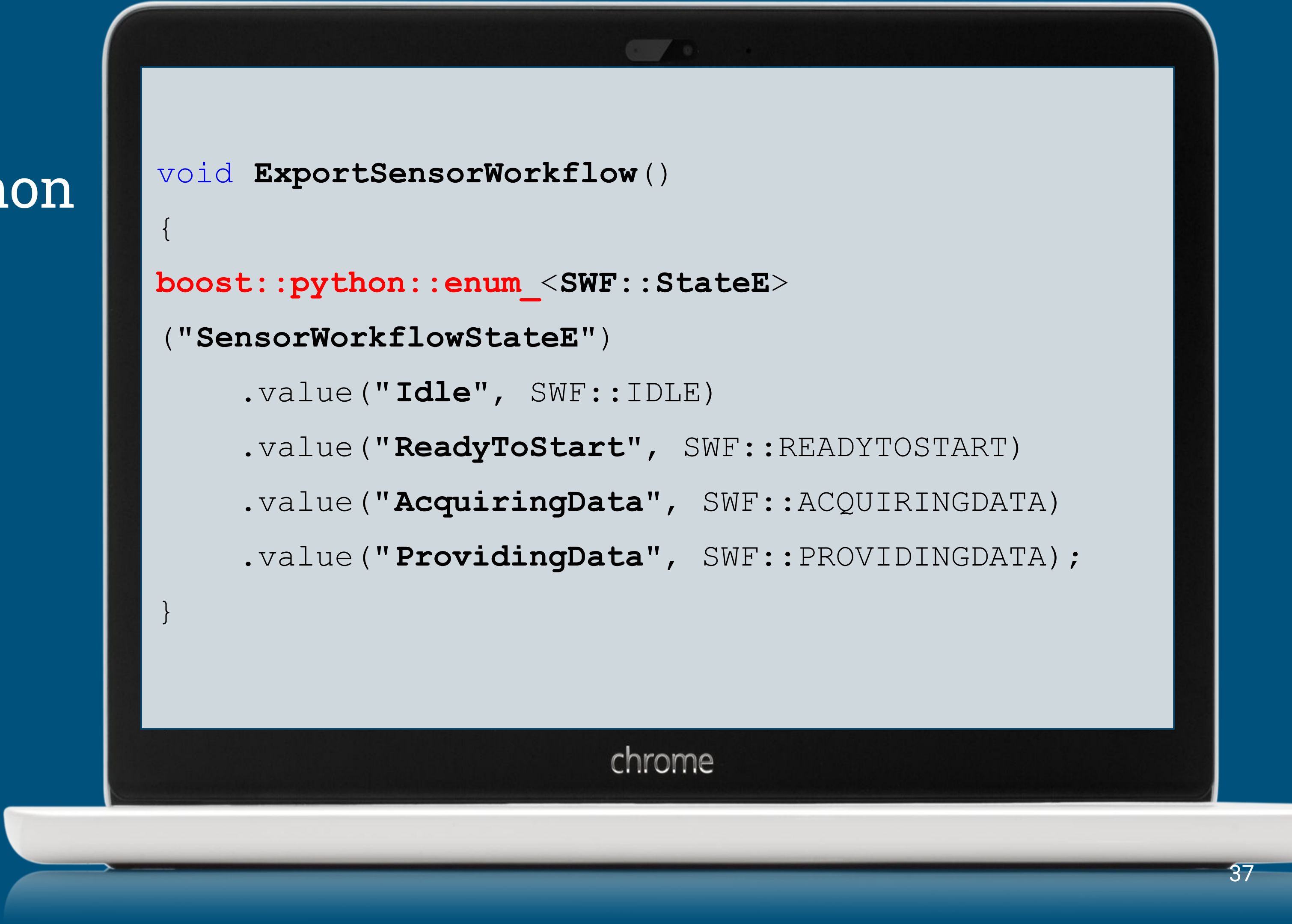


Boost python



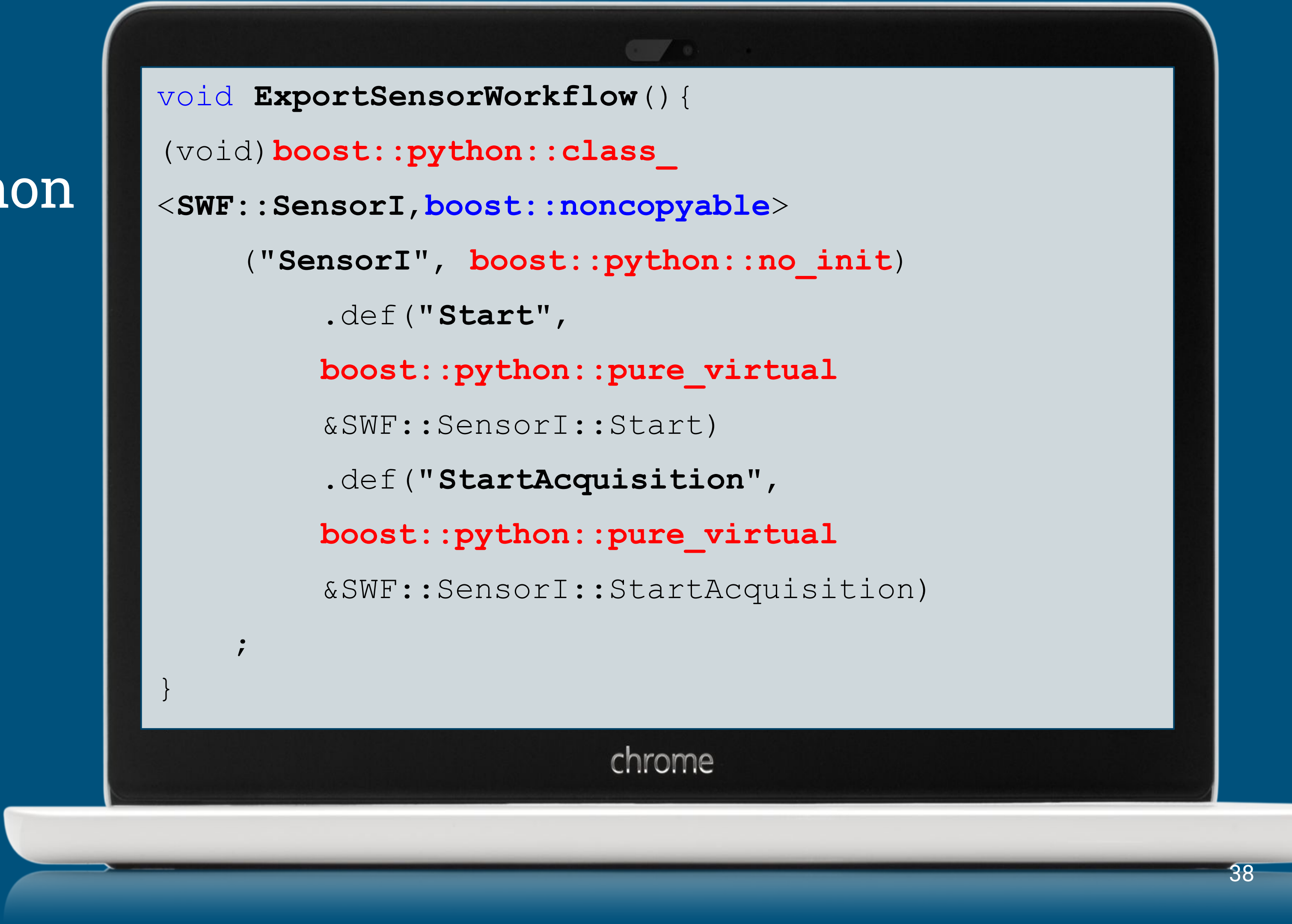
```
#include <boost/python.hpp>

void ExportSensorWorkflow();

BOOST_PYTHON_MODULE(SensorModule)
{
    ExportSensorWorkflow();
}
```

Boost python



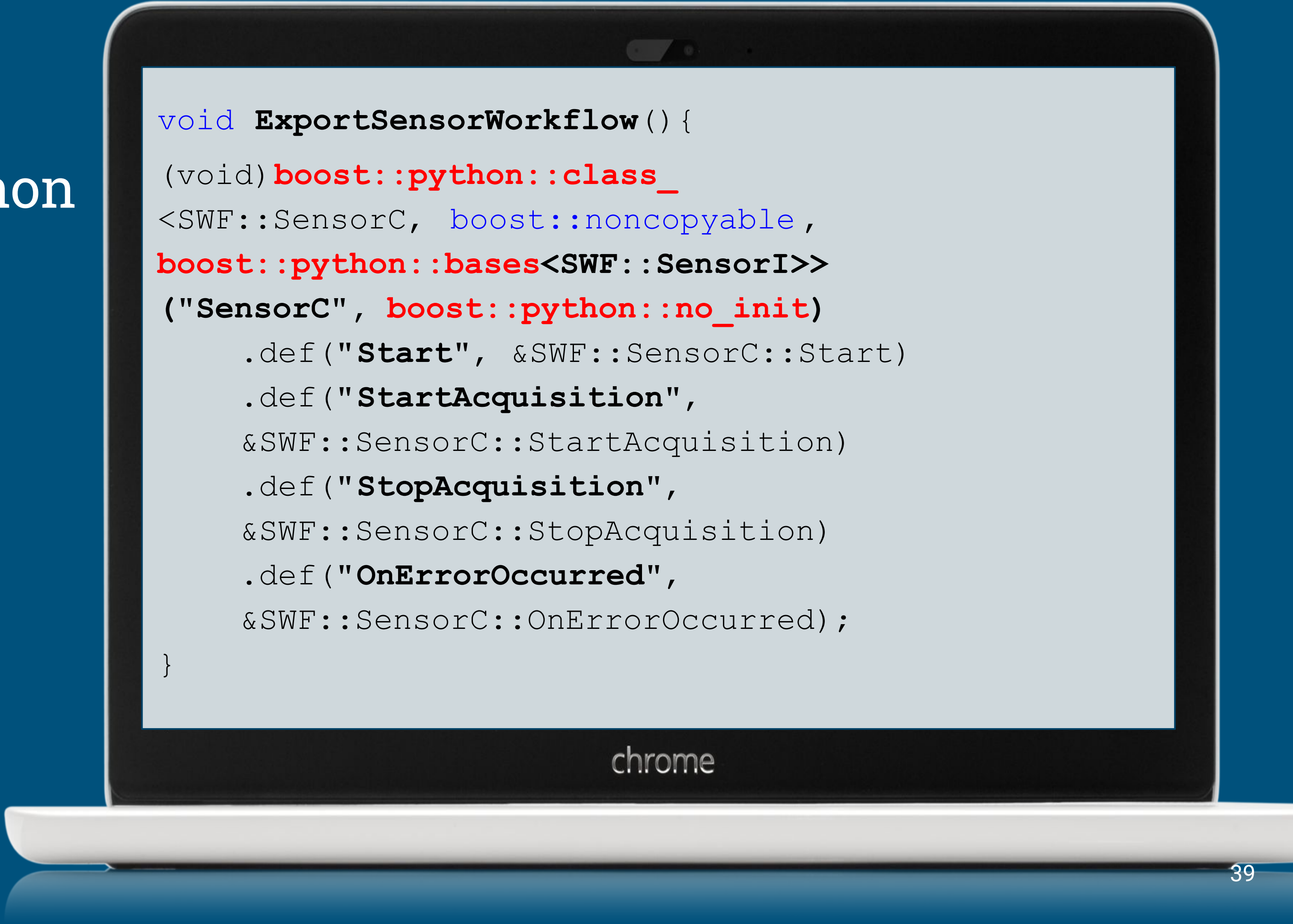
```
void ExportSensorWorkflow()
{
boost::python::enum_<SWF::StateE>
("SensorWorkflowStateE")
     .value("Idle", SWF::IDLE)
     .value("ReadyToStart", SWF::READYTOSTART)
     .value("AcquiringData", SWF::ACQUIRINGDATA)
     .value("ProvidingData", SWF::PROVIDINGDATA);
}
```

Boost python



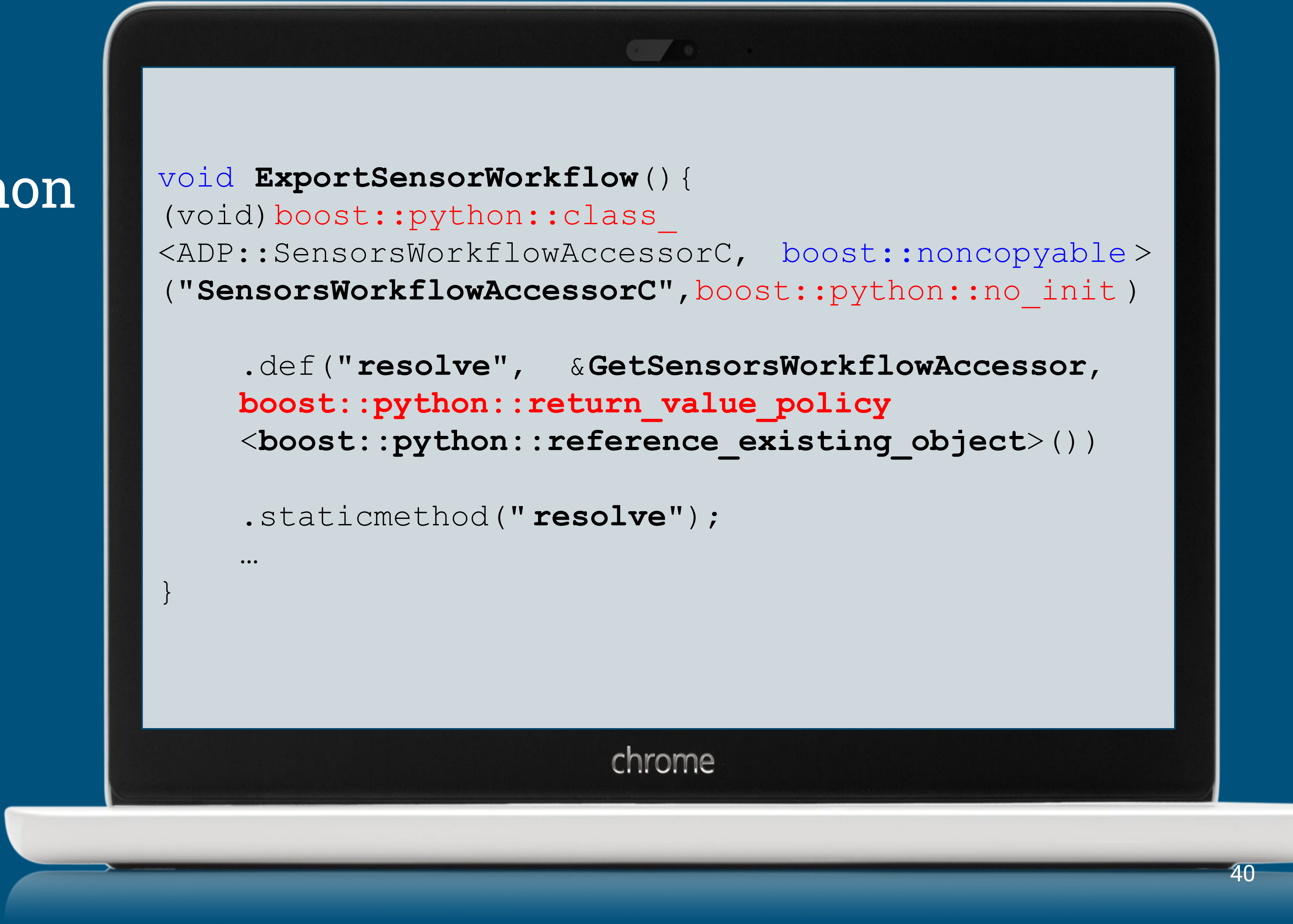
```
void ExportSensorWorkflow(){
(void)boost::python::class_
<SWF::SensorI,boost::noncopyable>
     ("SensorI", boost::python::no_init)
          .def("Start",
          boost::python::pure_virtual
          &SWF::SensorI::Start)
          .def("StartAcquisition",
          boost::python::pure_virtual
          &SWF::SensorI::StartAcquisition)
     ;
}
```

Boost python



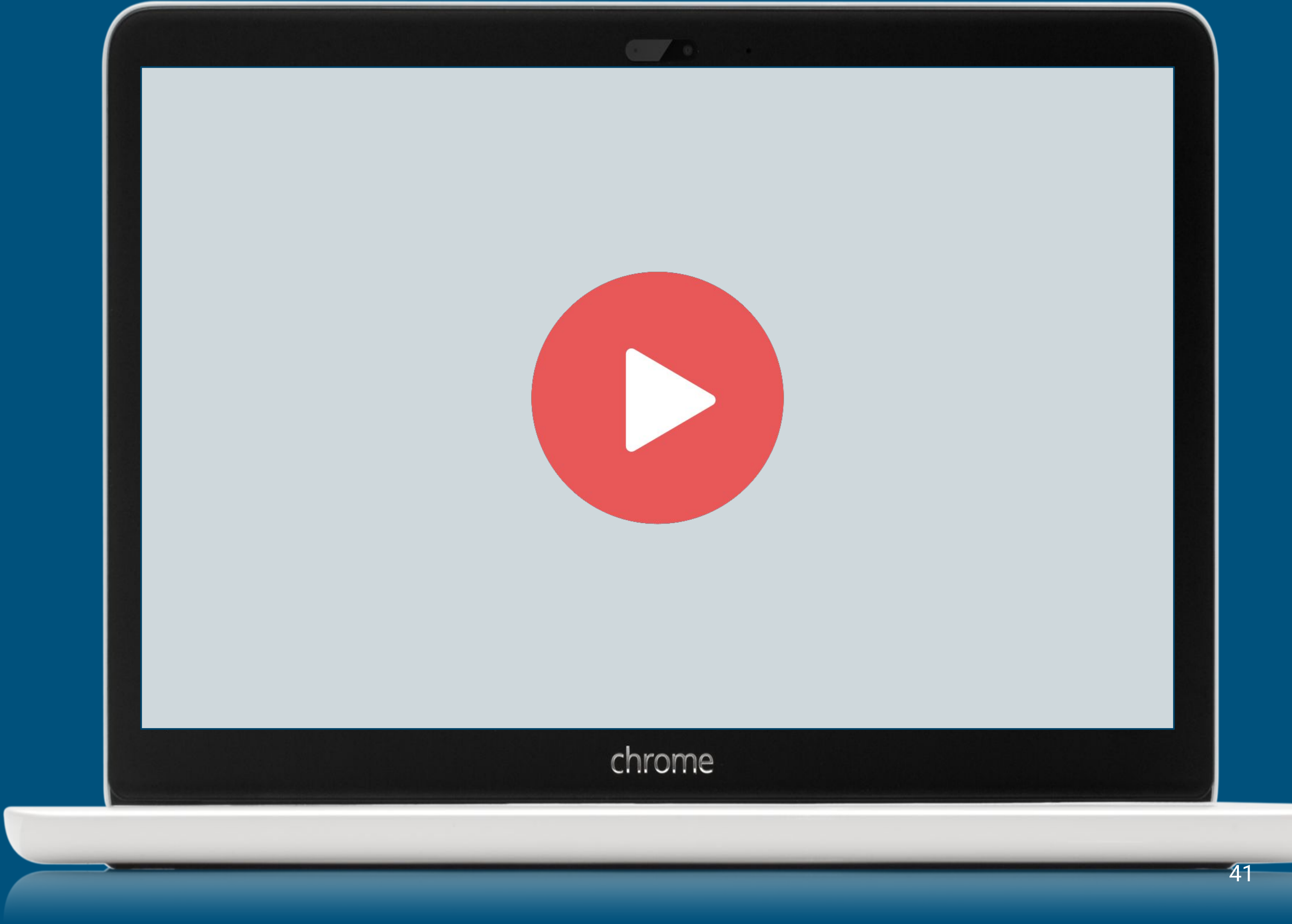
```
void ExportSensorWorkflow(){
(void)boost::python::class_
<SWF::SensorC, boost::noncopyable,
boost::python::bases<SWF::SensorI>>
("SensorC", boost::python::no_init)
    .def("Start", &SWF::SensorC::Start)
    .def("StartAcquisition",
    &SWF::SensorC::StartAcquisition)
    .def("StopAcquisition",
    &SWF::SensorC::StopAcquisition)
    .def("OnErrorOccurred",
    &SWF::SensorC::OnErrorOccurred);
}
```

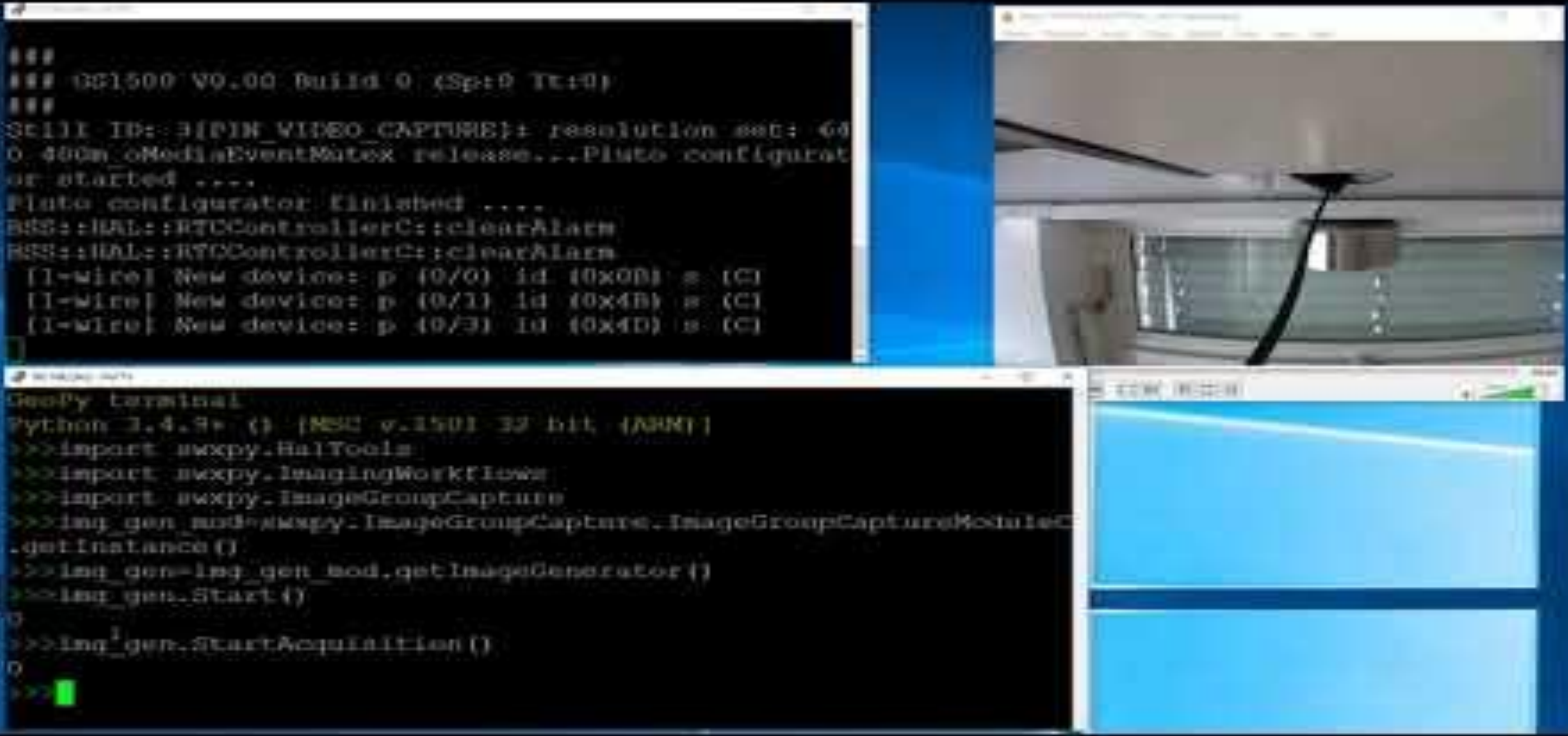
Boost python



```
void ExportSensorWorkflow(){
(void)boost::python::class_
<ADP::SensorsWorkflowAccessorC, boost::noncopyable>
("SensorsWorkflowAccessorC",boost::python::no_init)

     .def("resolve",  &GetSensorsWorkflowAccessor,
     boost::python::return_value_policy
     <boost::python::reference_existing_object>())

     .staticmethod("resolve");
     …
}
```

Video





```
###
### GS1500 V0.00 Build 0 (Sp:0 Tt:0)
###
Still ID: 3[PIN_VIDEO_CAPTURE]: resolution set: 64
0 480m_oMediaEventMutex release...Pluto configurat
or started ....
Pluto configurator finished ....
RSS::HAL::RTCControllerC::clearAlarm
RSS::HAL::RTCControllerC::clearAlarm
 [1-wire] New device: p (0/0) id (0x0B) s (C)
 [1-wire] New device: p (0/1) id (0x4B) s (C)
 [1-wire] New device: p (0/3) id (0x4D) s (C)
```

```
GeoPy terminal
Python 3.4.9+ () [MSC v.1501 32 bit (ARM)]
>>>import swxpy.HalTools
>>>import swxpy.ImagingWorkflows
>>>import swxpy.ImageGroupCapture
>>>img_gen_mod=swxpy.ImageGroupCapture.ImageGroupCaptureModuleC
.getInstance()
>>>img_gen=img_gen_mod.getImageGenerator()
>>>img_gen.Start()
0
>>>img_gen.StartAcquisition()
0
>>>
```

Summary

- Python bindings can help us to **add scripts to our C++ code**.
- Python bindings are **easy to start with** (if python.dll is available)
- Python interpreter can be included in a project and **accessed from remote**
- Several **possibilities of usage**: sensor checks, calibrations, automatic test etc.
- Add vertical opportunities for an **agile development** of the features

Question time

Rule: before each question tell me one thing (positive or negative) that you will take with you.

Author

Andrea Pallini

Contact me on Linkedin

Or write me an email:
andrea.pallini@leica-geosystems.com

Thank you