

You can call
your C++
code from
Python

Jack Grahl

Making a
package
setuptools
python setup.py

PyObjects
Python.h
PyObject *args
PyBool_FromLong

Building a
Python
module
PyMethodDef
PyModuleDef
PyMODINIT_FUNC

Putting
together the
Python
package
calendar.py
setuptools.Extension
setup

You can call your C++ code from Python

Jack Grahl

PrismFP Analytics

July 9, 2017

Table of Contents

You can call
your C++
code from
Python

Jack Grahl

Making a package

setuptools
python setup.py

PyObjects

Python.h
PyObject *args
PyBool_FromLong

Building a Python module

PyMethodDef
PyModuleDef
PyMODINIT_FUNC

Putting together the Python package

calendar.py
setuptools.Extension
setup

1 Making a package

- setuptools
- python setup.py

2 PyObjects

- Python.h
- PyObject *args
- PyBool_FromLong

3 Building a Python module

- PyMethodDef
- PyModuleDef
- PyMODINIT_FUNC

4 Putting together the Python package

- calendar.py
- setuptools.Extension
- setup

setuptools

You can call
your C++
code from
Python

Jack Grahl

Making a
package
setuptools
python setup.py
PyObjects
Python.h
PyObject *args
PyBool_FromLong

Building a
Python
module
PyMethodDef
PyModuleDef
PyMODINIT_FUNC

Putting
together the
Python
package
calendar.py
setuptools.Extension
setup

```
from setuptools import setup
```

```
setup(  
    name='calexicon',  
    version='0.1.2',  
    description='Calendar stuff',  
    url='http://github.com/jwg4/calexicon',  
    author='Jack Grahl',  
    author_email='jack.grahl@gmail.com',  
    license='Apache License 2.0',  
    packages=['calexicon'],  
    test_suite='nose.collector',  
    tests_require=['nose', 'hypothesis']  
)
```

python setup.py

You can call
your C++
code from
Python

Jack Grahl

Making a
package
setuptools
python setup.py

PyObjects
Python.h
PyObject *args
PyBool_FromLong

Building a
Python
module
PyMethodDef
PyModuleDef
PyMODINIT_FUNC

Putting
together the
Python
package
calendar.py
setuptools.Extension
setup

env :

- TASK=test
- TASK=sdist
- TASK=build
- TASK=bdist_egg

script :

- cd pyQuantuccia
- python setup.py \$TASK

Table of Contents

You can call
your C++
code from
Python

Jack Grahl

Making a
package
setuptools
python setup.py

PyObjects
Python.h
PyObject *args
PyBool_FromLong

Building a
Python
module
PyMethodDef
PyModuleDef
PyMODINIT_FUNC

Putting
together the
Python
package
calendar.py
setuptools.Extension
setup

- 1 Making a package
 - setuptools
 - python setup.py
- 2 PyObjects
 - Python.h
 - PyObject *args
 - PyBool_FromLong
- 3 Building a Python module
 - PyMethodDef
 - PyModuleDef
 - PyMODINIT_FUNC
- 4 Putting together the Python package
 - calendar.py
 - setuptools.Extension
 - setup

Python.h

You can call
your C++
code from
Python

Jack Grahl

Making a
package
setuptools
python setup.py

PyObjects
Python.h
PyObject *args
PyBool_FromLong

Building a
Python
module
PyMethodDef
PyModuleDef
PyMODINIT_FUNC

Putting
together the
Python
package
calendar.py
setuptools.Extension
setup

```
#include <Python.h>
#include "Quantuccia/ql/time/calendar.hpp"
#include "Quantuccia/ql/time/date.hpp"
#include "Quantuccia/ql/time/calendars/  
unitedkingdom.hpp"
```

PyObject *args

You can call
your C++
code from
Python

Jack Grahl

Making a
package

setuptools
python setup.py

PyObjects

Python.h
PyObject *args
PyBool_FromLong

Building a
Python
module

PyMethodDef
PyModuleDef
PyMODINIT_FUNC

Putting
together the
Python
package

calendar.py
setuptools.Extension
setup

```
static PyObject*
united_kingdom_is_business_day(PyObject *
    self, PyObject *args)
{
    int year;
    int month;
    int day;
    if (!PyArg_ParseTuple(args, "iii|", \&
        year, \&month, \&day))
        return NULL;
    QuantLib::Day d(day);
    QuantLib::Month m = static_cast<QuantLib
        ::Month>(month);
    QuantLib::Year y(year);
```

PyBool_FromLong

You can call
your C++
code from
Python

Jack Grahl

Making a
package
setuptools
python setup.py

PyObjects
Python.h
PyObject *args
PyBool_FromLong

Building a
Python
module
PyMethodDef
PyModuleDef
PyMODINIT_FUNC

Putting
together the
Python
package
calendar.py
setuptools.Extension
setup

```
static PyObject*
united_kingdom_is_business_day(PyObject *
    self, PyObject *args)
{
    // ...
    QuantLib::Date date(d, m, y);
    QuantLib::UnitedKingdom calendar(
        QuantLib::UnitedKingdom::Exchange);
    bool result = calendar.isBusinessDay(
        date);
    return PyBool_FromLong(result);
}
```


Table of Contents

You can call
your C++
code from
Python

Jack Grahl

Making a
package
setuptools
python setup.py

PyObjects
Python.h
PyObject *args
PyBool_FromLong

Building a
Python
module
PyMethodDef
PyModuleDef
PyMODINIT_FUNC

Putting
together the
Python
package
calendar.py
setuptools.Extension
setup

- 1 Making a package
 - setuptools
 - python setup.py
- 2 PyObjects
 - Python.h
 - PyObject *args
 - PyBool_FromLong
- 3 Building a Python module
 - PyMethodDef
 - PyModuleDef
 - PyMODINIT_FUNC
- 4 Putting together the Python package
 - calendar.py
 - setuptools.Extension
 - setup

PyMethodDef

You can call
your C++
code from
Python

Jack Grahl

Making a
package
setuptools
python setup.py

PyObjects
Python.h
PyObject *args
PyBool_FromLong

Building a
Python
module
PyMethodDef
PyModuleDef
PyMODINIT_FUNC

Putting
together the
Python
package
calendar.py
setuptools.Extension
setup

```
static PyMethodDef QuantucciaMethods[] = {  
    {  
        "united_kingdom_is_business_day",  
        (PyCFunction)  
            united_kingdom_is_business_day ,  
        METH_VARARGS,  
        NULL  
    },  
    {NULL, NULL, 0, NULL}  
};
```

PyModuleDef

You can call
your C++
code from
Python

Jack Grahl

Making a
package
setuptools
python setup.py

PyObjects
Python.h
PyObject *args
PyBool_FromLong

Building a
Python
module
PyMethodDef
PyModuleDef
PyMODINIT_FUNC

Putting
together the
Python
package
calendar.py
setuptools.Extension
setup

```
static struct PyModuleDef
quantuccia_module_def = {
    PyModuleDef_HEAD_INIT,
    "quantuccia",
    NULL,
    -1,
    QuantucciaMethods,
    NULL,
    NULL,
    NULL,
    NULL
};
```

PyMODINIT_FUNC

You can call
your C++
code from
Python

Jack Grahl

Making a
package
setuptools
python setup.py

PyObjects
Python.h
PyObject *args
PyBool_FromLong

Building a
Python
module
PyMethodDef
PyModuleDef
PyMODINIT_FUNC

Putting
together the
Python
package
calendar.py
setuptools.Extension
setup

```
PyMODINIT_FUNC PyInit_quantuccia(void){  
    PyObject *m;  
    m = PyModule_Create(&  
        quantuccia_module_def);  
    return m;  
}
```

Table of Contents

You can call
your C++
code from
Python

Jack Grahl

Making a
package
setuptools
python setup.py

PyObjects
Python.h
PyObject *args
PyBool_FromLong

Building a
Python
module
PyMethodDef
PyModuleDef
PyMODINIT_FUNC

Putting
together the
Python
package
calendar.py
setuptools.Extension
setup

- 1 Making a package
 - setuptools
 - python setup.py
- 2 PyObjects
 - Python.h
 - PyObject *args
 - PyBool_FromLong
- 3 Building a Python module
 - PyMethodDef
 - PyModuleDef
 - PyMODINIT_FUNC
- 4 Putting together the Python package
 - calendar.py
 - setuptools.Extension
 - setup

calendar.py

You can call
your C++
code from
Python

Jack Grahl

Making a
package
setuptools
python setup.py

PyObjects
Python.h
PyObject *args
PyBool_FromLong

Building a
Python
module
PyMethodDef
PyModuleDef
PyMODINIT_FUNC

Putting
together the
Python
package
calendar.py
setuptools.Extension
setup

```
from quantuccia import  
    united_kingdom_is_business_day as  
    c_function
```

```
def united_kingdom_is_business_day(dt):  
    y = dt.year  
    m = dt.month  
    d = dt.day  
    return c_function(y, m, d)
```

setuptools.Extension

You can call
your C++
code from
Python

Jack Grahl

Making a
package

setuptools
python setup.py

PyObjects

Python.h
PyObject *args
PyBool_FromLong

Building a
Python
module

PyMethodDef
PyModuleDef
PyMODINIT_FUNC

Putting
together the
Python
package

calendar.py
setuptools.Extension
setup

```
import setuptools

qu_ext = setuptools.Extension(
    'quantuccia',
    include_dirs=['src/Quantuccia'] +
        extra_dirs,
    sources=['src/pyQuantuccia.cpp'],
    headers=[]
)
```

setup

You can call
your C++
code from
Python

Jack Grahl

Making a
package

setuptools
python setup.py

PyObjects

Python.h
PyObject *args
PyBool_FromLong

Building a
Python
module

PyMethodDef
PyModuleDef
PyMODINIT_FUNC

Putting
together the
Python
package

calendar.py
setuptools.Extension
setup

```
setuptools.setup(  
    name='pyQuantuccia',  
    author='Jack Grahl',  
    author_email='jack.grahl@gmail.com',  
    version='0.2.0',  
    packages=['pyQuantuccia'],  
    package_dir={'': 'src'},  
    setup_requires=['pytest-runner'],  
    tests_require=['pytest'],  
    test_suite='tests',  
    ext_modules=[qu_ext]  
)
```


Thank you.

You can call
your C++
code from
Python

Jack Grahl

Making a
package

setuptools
python setup.py

PyObjects

Python.h
PyObject *args
PyBool_FromLong

Building a
Python
module

PyMethodDef
PyModuleDef
PyMODINIT_FUNC

Putting
together the
Python
package

calendar.py
setuptools.Extension
setup

github.com/jwg4
www.prismfp.com
fullstackquants.org

test_calendar.py

You can call
your C++
code from
Python

Jack Grahl

Making a
package
setuptools
python setup.py

PyObjects
Python.h
PyObject *args
PyBool_FromLong

Building a
Python
module
PyMethodDef
PyModuleDef
PyMODINIT_FUNC

Putting
together the
Python
package
calendar.py
setuptools.Extension
setup

```
from datetime import date
```

```
from pyQuantuccia import calendar
```

```
def
```

```
test_united_kingdom_is_business_day_identif  
():
```

```
    """ Check a single day which isn't a  
        holiday.
```

```
    """
```

```
    assert(calendar.  
           united_kingdom_is_business_day(date  
           (2017, 4, 17)) is False)
```

extra_dirs

You can call
your C++
code from
Python

Jack Grahl

Making a
package

setuptools
python setup.py

PyObjects

Python.h
PyObject *args
PyBool_FromLong

Building a
Python
module

PyMethodDef
PyModuleDef
PyMODINIT_FUNC

Putting
together the
Python
package

calendar.py
setuptools.Extension
setup

```
import platform
import setuptools
```

```
if platform.system() == 'Windows':
    extra_dirs = [
        "C:\\Program Files (x86)\\Windows
        Kits\\10\\include\\10.0.10240.0\\
        ucrt"
    ]
else:
    extra_dirs = []
```