Turbodbc:
From C to Python

Dr. Michael König, Blue Yonder GmbH

michael.koenig@blue-yonder.com
@turbodbc

BlueYonder
Best decisions, delivered daily

turbodbc

turbocharged database access for data scientists.

```
from turbodbc import connect

connection = connect("My database")
cursor = connection.cursor()
cursor.execute_many("INSERT INTO my_table VALUES (?, ?)",
                    [["Hello", 42],
                     ["C++ user group", 23]])
```

BlueYonder

Best decisions, delivered daily

Step 0: ODBC API

- ~80 very low-level C functions
- Functions have up to 10 parameters

```
SQLRETURN SQLBindParameter(SQLHSTMT StatementHandle,
                           SQLUSMALLINT ParameterNumber,
                           SQLSMALLINT InputOutputType,
                           SQLSMALLINT ValueType,
                           SQLSMALLINT ParameterType,
                           SQLULEN ColumnSize,
                           SQLSMALLINT DecimalDigits,
                           SQLPOINTER ParameterValuePtr,
                           SQLLEN BufferLength,
                           SQLLEN * StrLen_or_IndPtr);
```

Step 1: C++ abstractions

- Get C++ side similar to Python interface
  - Interfaces / mocks → stateless unit testing
  - Sensible types / RAII
  - Exception handling
  - High-level logic

```
class connection {
public:
    void commit();
    void rollback();
    cursor make_cursor();
};
```

BlueYonder
Best decisions, delivered daily

Step 2: Translation

- Expose C++ code to Python with Pybind11

```
#include <pybind11/pybind11.h>

PYBIND11_PLUGIN(turbodbc_intern)
{
    pybind11::module module("turbodbc_intern",
                            "Some documentation");

    pybind11::class_<connection>(module, "Connection")
        .def("commit", &connection::commit)
        .def("rollback", &connection::rollback)
        .def("cursor", &connection::make_cursor);
}
```

BlueYonder
Best decisions, delivered daily

Step 3: Python sugarcoating

- When performance and bits do not matter

```
class Connection(object):
    def __init__(self, cpp_connection):
        self.cpp_connection = cpp_connection
        self.cursors = []

    @translate_exceptions
    def cursor(self):
        c = Cursor(self.cpp_connection.cursor())
        self.cursors.append(c)
        return c
```

BlueYonder
Best decisions, delivered daily

Open source learnings

- Open source *early*
  - Excellent tools: GitHub, Travis CI, AppVeyor
  - Public software only
- Be portable
  - Prefer the standard library
  - CMake!

BlueYonder
Best decisions, delivered daily

Links

- Turbodbc @ GitHub
- Pybind11 @ GitHub

- Making of turbodbc @ tech.blue-yonder.com

BlueYonder
Best decisions, delivered daily