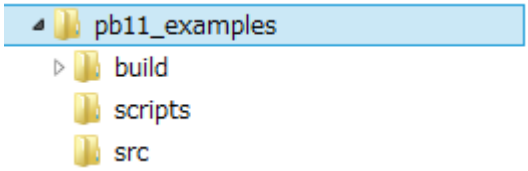
pybind11のインストール、使用手順（Windowsの場合）

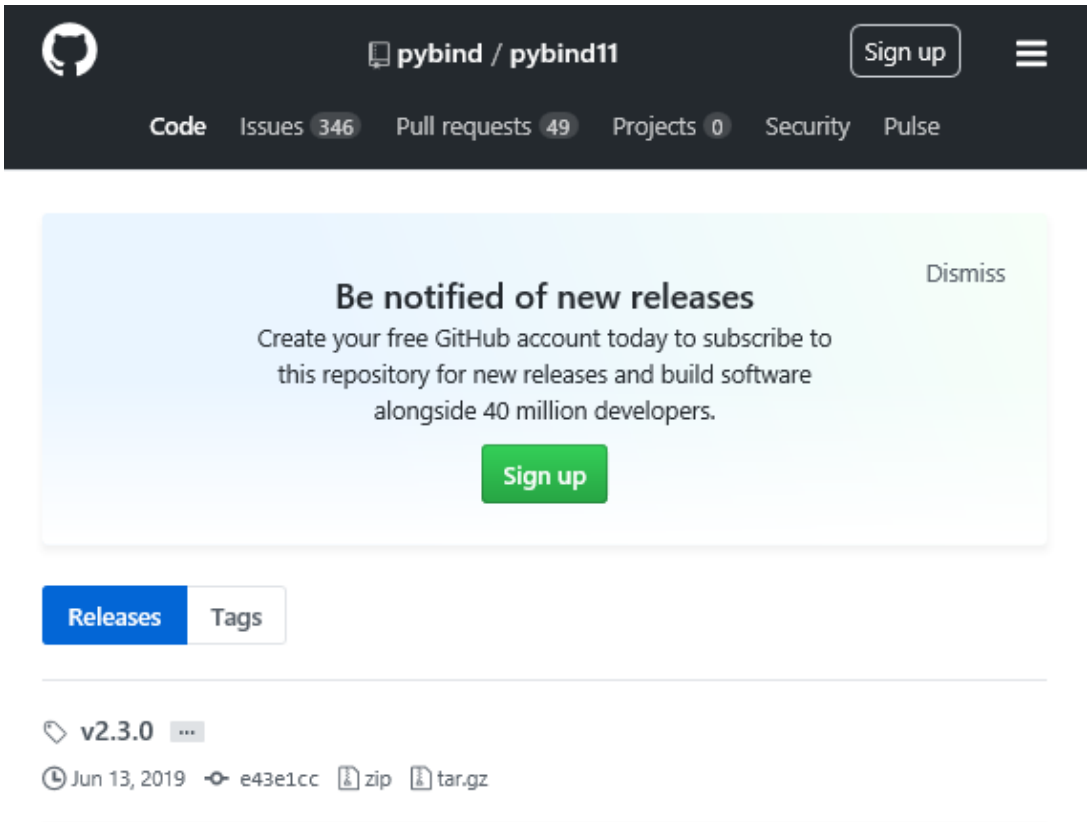
本文書ではpybind11 というライブラリを用いて Python スクリプトからC++ で実装したモジュールにある関数を呼び出す手順を示す。ビルドには Visual Studio Community 2019、CMake、NMake を使用する。（CMake の効用により、以下の手順のうち、NMake を GNU Make （make）で置換えれば、Linuxの場合もほぼ同様の手順でビルド、実行できる。）

1. 次のディレクトリ構造を作成する。



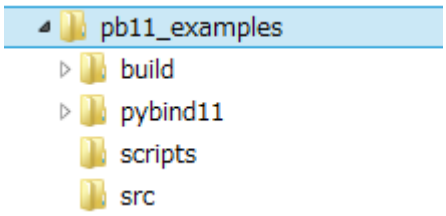
2. 下記の pybind11 の GitHubリポジトリに ブラウザでアクセスする

https://github.com/pybind/pybind11/releases



3. 上記Webページのリリースタグ名 'v2.3.0' の直下にあるリンク [zip] のZIPファイルをダウンロードする。以下、このZIPファイルを pybind11-2.3.0.zip とする。

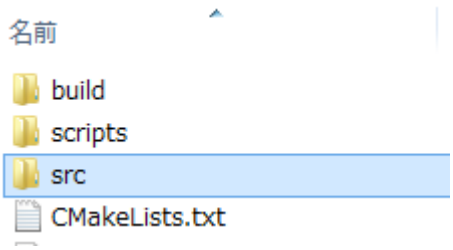
4. pybind11-2.3.0.zip を ディレクトリ pb11_examples に展開する。



5. 展開されてできた サブディレクトリ pybind11-2.3.0 の名称を pybind11 に変更する。これによりサブディレクトリ pb11_examples/pybind11/include/ 等が存在する状態となる。

6. pb11_examples/ の直下に以下の内容のテキストファイル CMakeLists.txt を作成する。

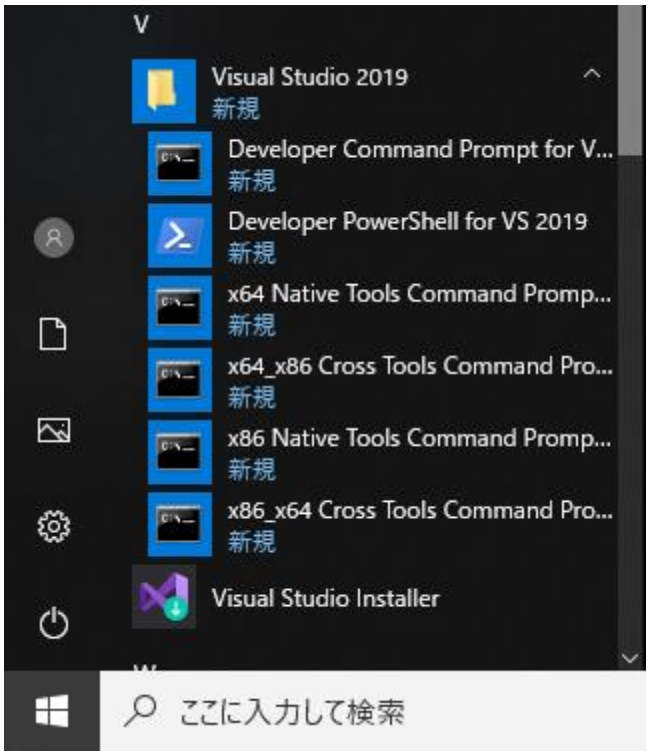
```
cmake_minimum_required(VERSION 2.8.12)
project(pb11_examples)

add_subdirectory(pybind11)
pybind11_add_module(ex_simple src/ex_simple.cpp)
```



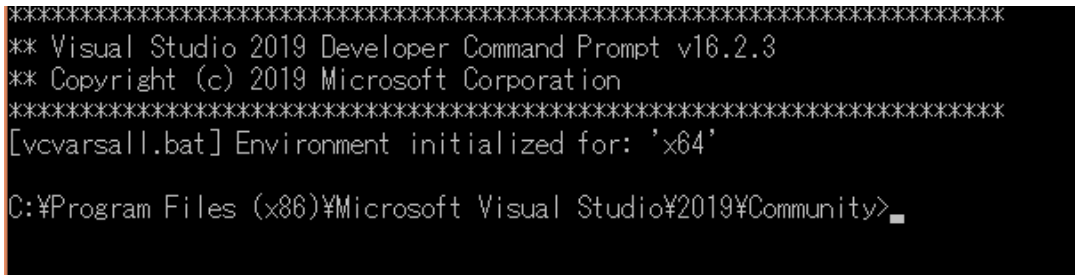
7. サブディレクトリsrc/ にソースファイル ex_simple.cpp を作成し、次の内容を記述する。ソースファイルを保存するときには、文字コードは utf-8 (byte-order-mark（BOM）付き)に設定して保存する。

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

void say_hello()
{
  std::cout << "Hello, from C++" << std::endl;
}

PYBIND11_MODULE(ex_simple, m)
{
  m.def("hello",    // Python で扱うときの メソッド名
        say_hello, // 上記に対応付ける関数オブジェクト
        u8"ターミナル(標準出力)に'Hello'を表示する"); //docstring
}
```

8. タスクバーの検索で"command" （または "tools"）を検索する。これにより [x64 Native Tools Command Prompt for VS 2019]という項目が見つかる。



9. 項目 [x64 Native Tools Command Prompt for VS 2019] をクリックしてコマンドプロンプトを開く。このコマンドプロンプトでは NMake 等を利用するための設定が読み込まれている。

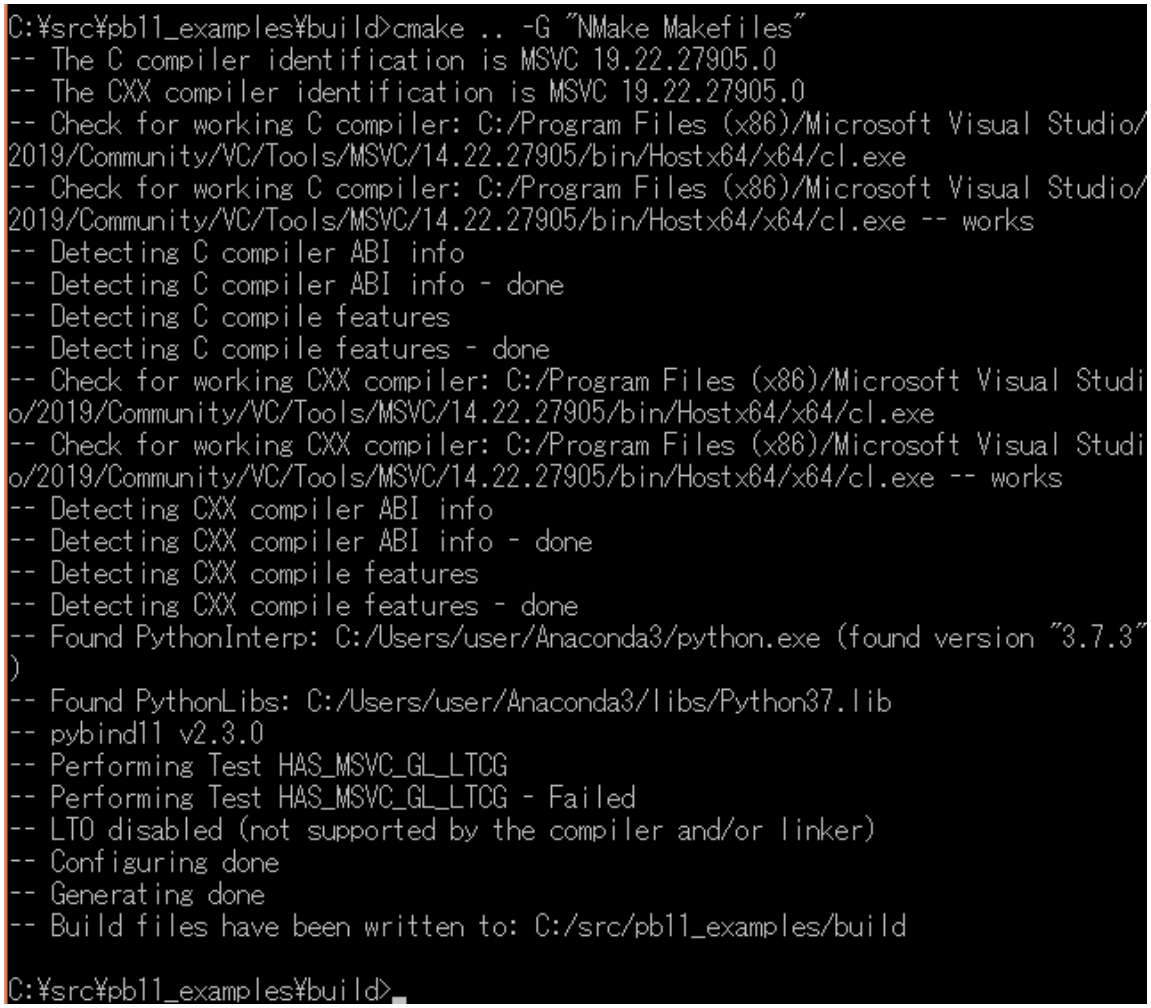


10. 次のコマンドにより、カレントディレクトリをbuildに変更する

```
cd C:¥src¥pb11_examples¥build
```

11. 次のコマンドにより、CMakeLists.txt から NMake 用のMakefileを生成する。

```
cmake .. -G "NMake Makefiles"
```



12. 次のコマンドによりソースファイルのビルドを行う。

```
nmake
```

13. 以上によってモジュールex_simple.cp37-win_amd64.pyd （動的リンクライブラリ（DLL））が生成された。なお、”37”は、Python 3.7.* に同梱のヘッダファイルを利用してビルドしたことを意味し、異なるバージョンの Python インタプリタでは、このDLLを正しくロードできない。次に、このDLLを利用するPython スクリプトを作成する。

14. サブディレクトリ scripts の直下に以下の内容のPython スクリプト test_ex_simple.py を作成する。

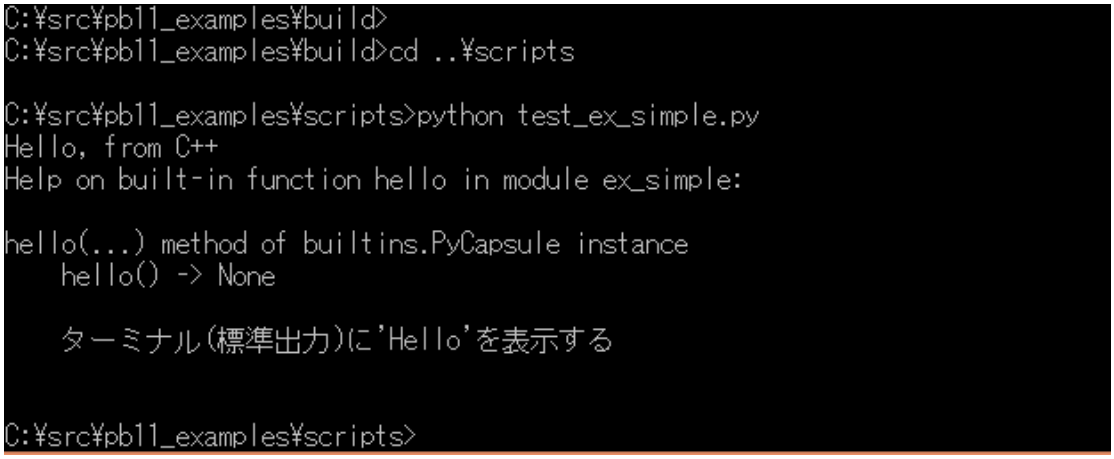
```
#!/usr/bin/env python
# -*- coding: utf-8 -*-
import sys
sys.path.append('../build') # モジュールの置き場所に応じて指定する
import ex_simple

ex_simple.hello() # モジュール内の関数を呼び出す
help(ex_simple.hello) # hello の説明文を表示する
```

15. 次のコマンドにより、カレントディレクトリをscriptsに変更する

```
cd ..¥scripts
```

16. 次のコマンドにより test_ex_simple.py を実行する。

```
python test_ex_simple.py
```



以上