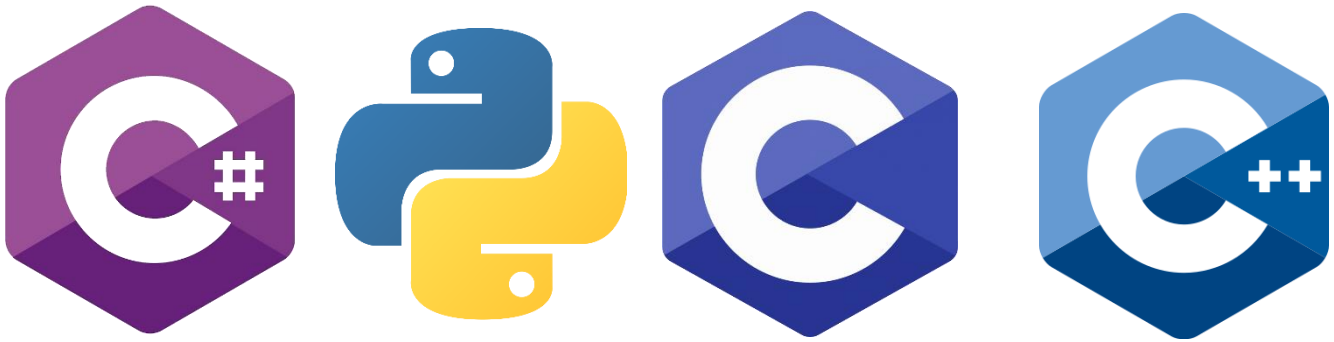
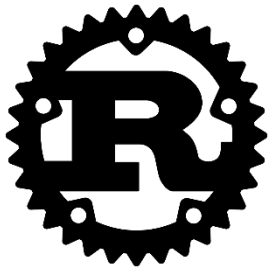
Interfaçage du C#, Python, C, C++, Go et Rust avec LabVIEW





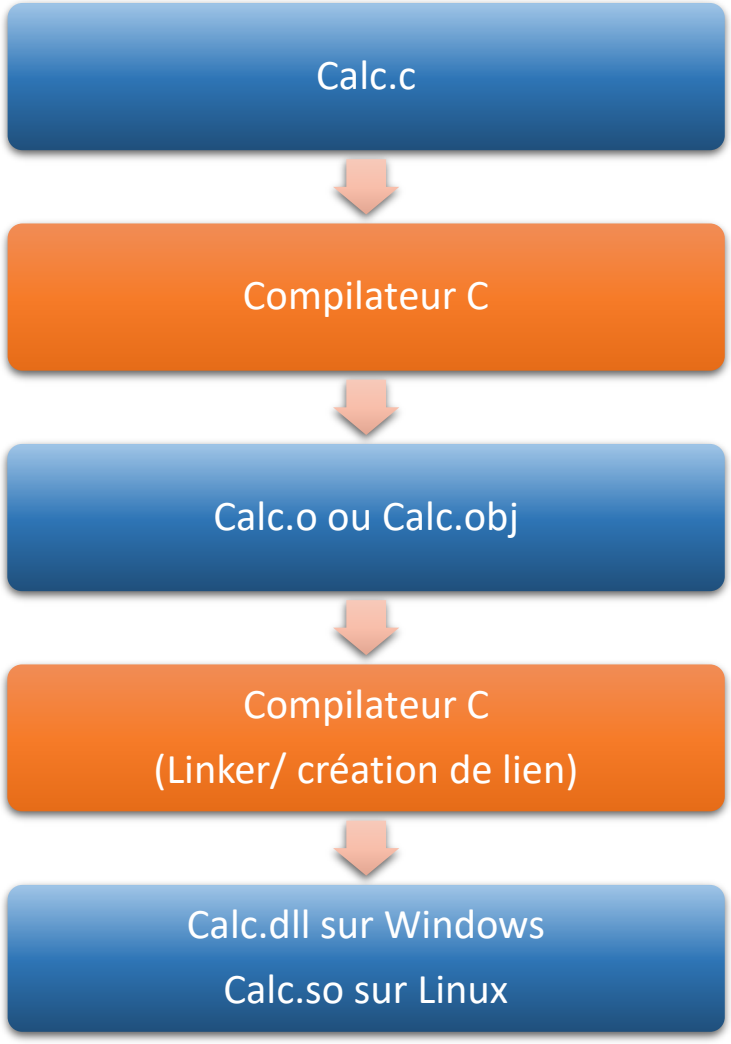
Sommaire

I. Interfaçage de fonction

1. C
2. C++
3. Python
4. Go
5. Rust

Des fonctions C dans LabVIEW

Fonctions C (1/2)



Ouvrir — Calc.c — ~/Documents/Python-so/cc — Enregistrer

```
double addition(double a, double b)
{
        return a+b;
}

double multiplication(double a, double b)
{
        return a*b;
}
```

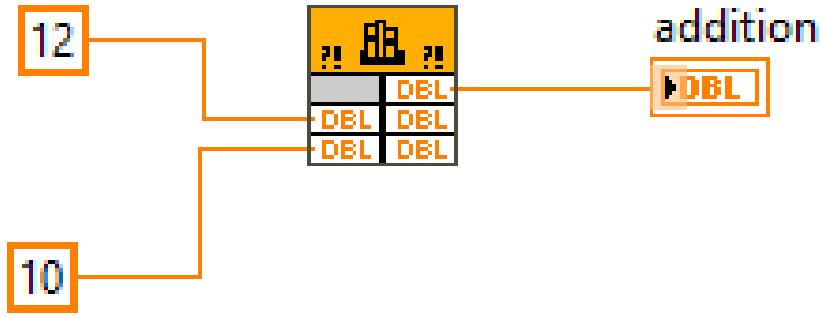
Lig 1, Col 1 INS

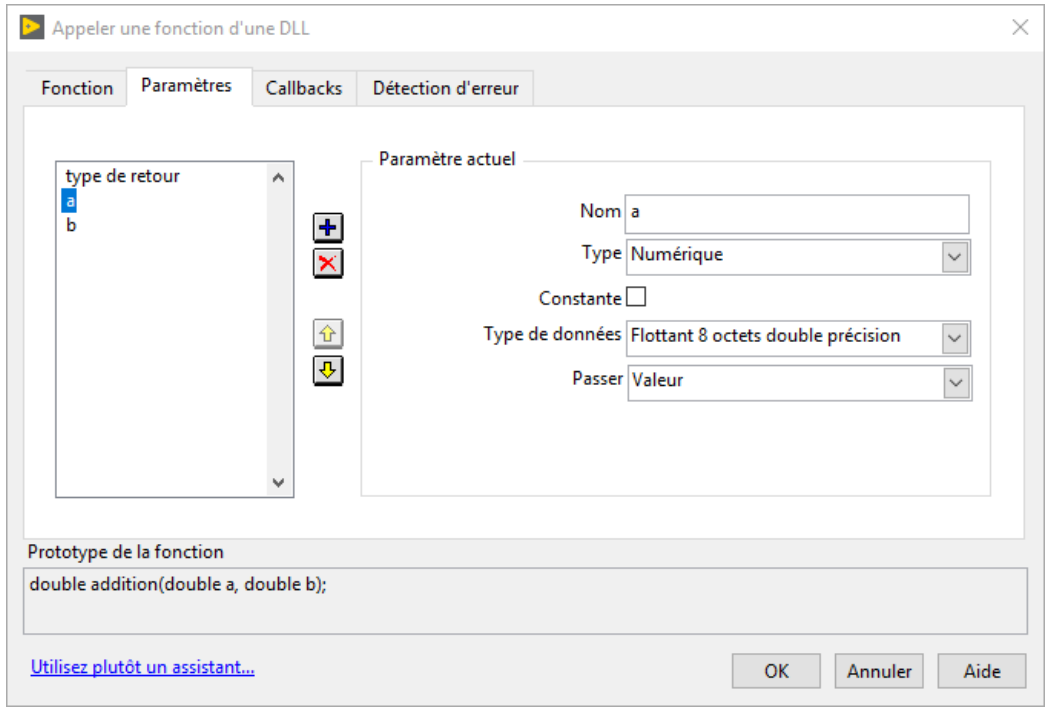
mjulien@localhost:~/Documents/Python-so/cc

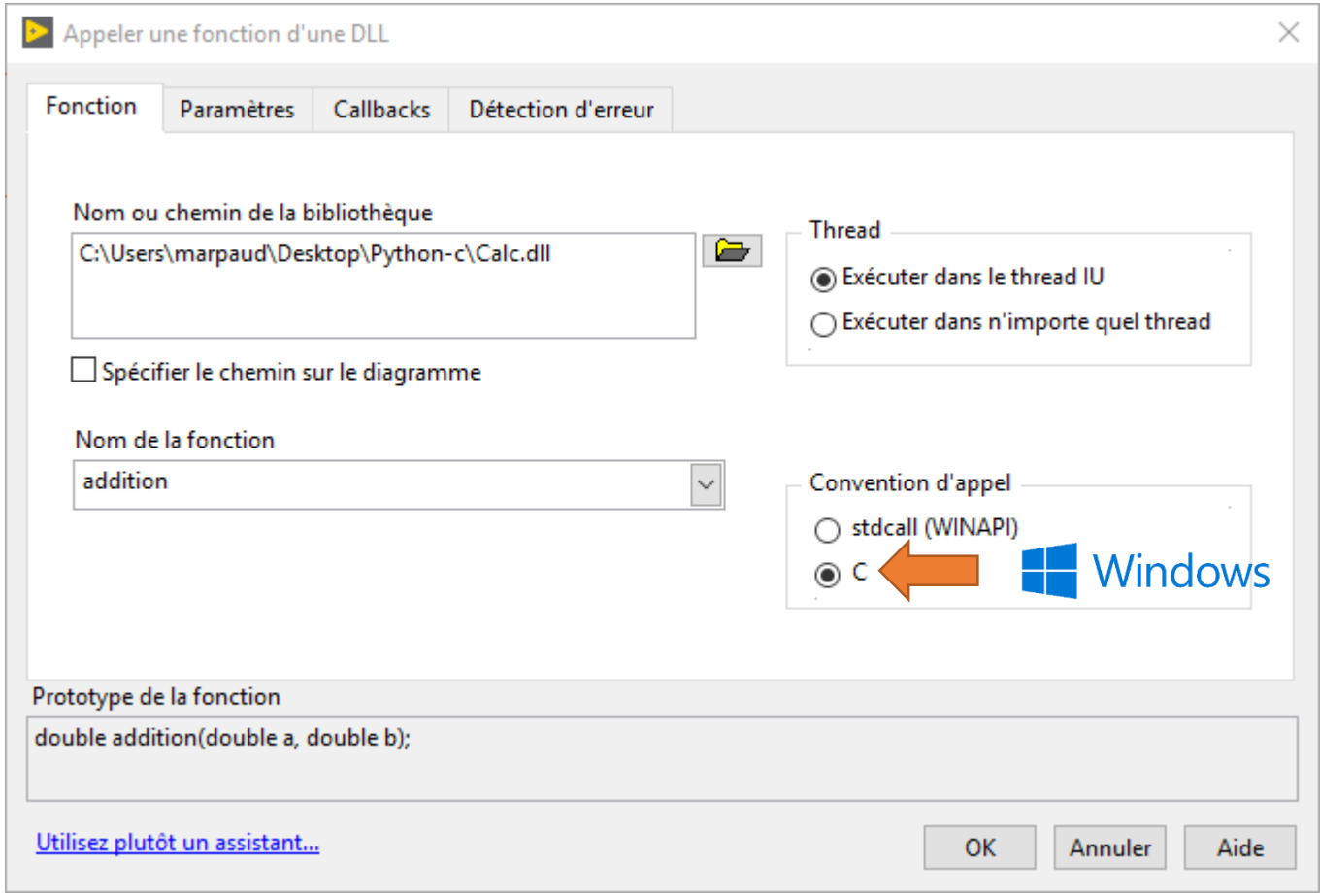
Fichier Édition Affichage Rechercher Terminal Aide

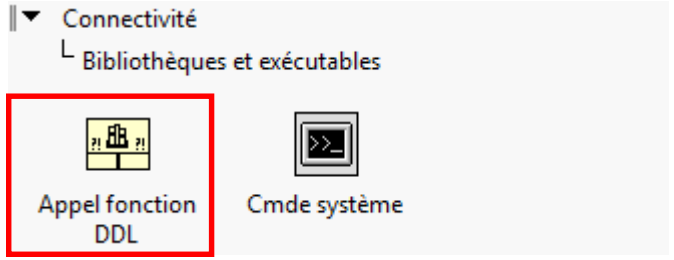
```
[mjulien@localhost cc]$ gcc -c -fpic Calc.c
[mjulien@localhost cc]$ gcc -shared *.o -o Calc.so
[mjulien@localhost cc]$
```

Fonctions C (2/2)

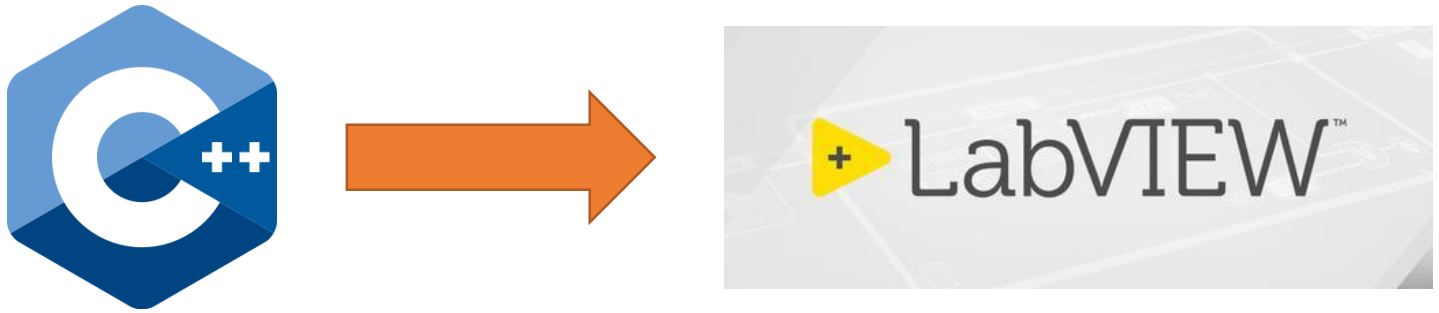








Des fonctions C++ dans LabVIEW



Fonctions C++ (1/3)

Calc.cpp — ~/Documents/Python-so/cpp-func...

```
#include "Calc.h"

double addition(double a, double b)
{
        return a+b;
}

double multiplication(double a, double b)
{
        return a*b;
}
```

C++ ▾ Largeur des tabulations : 8 ▾ Lig 1, Col 18 INS

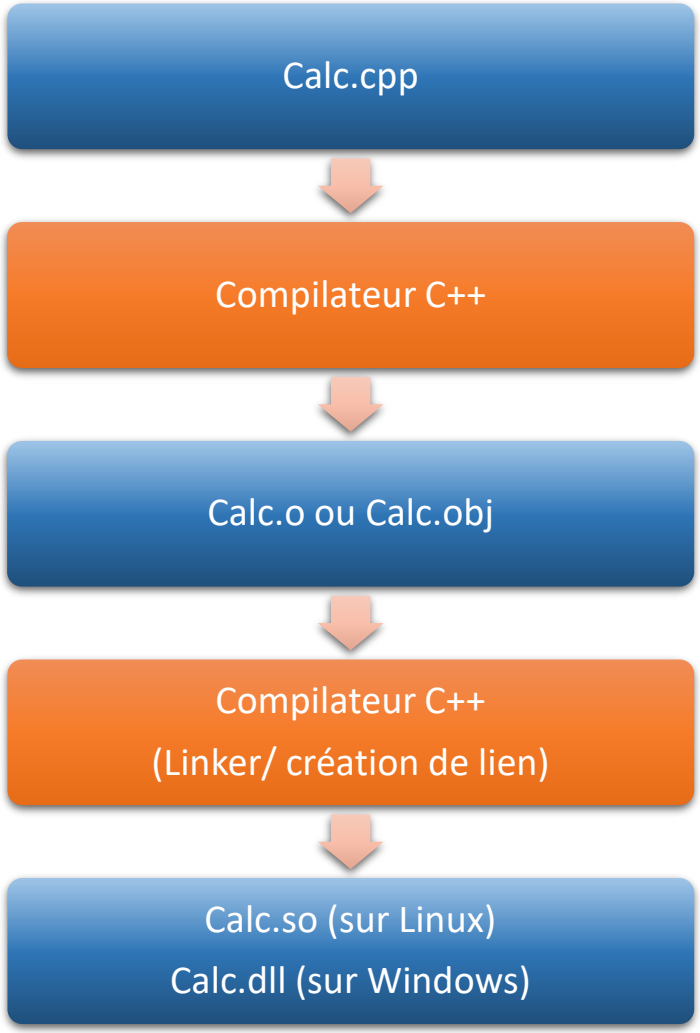
Calc.h — ~/Documents/Python-so/cpp-func...

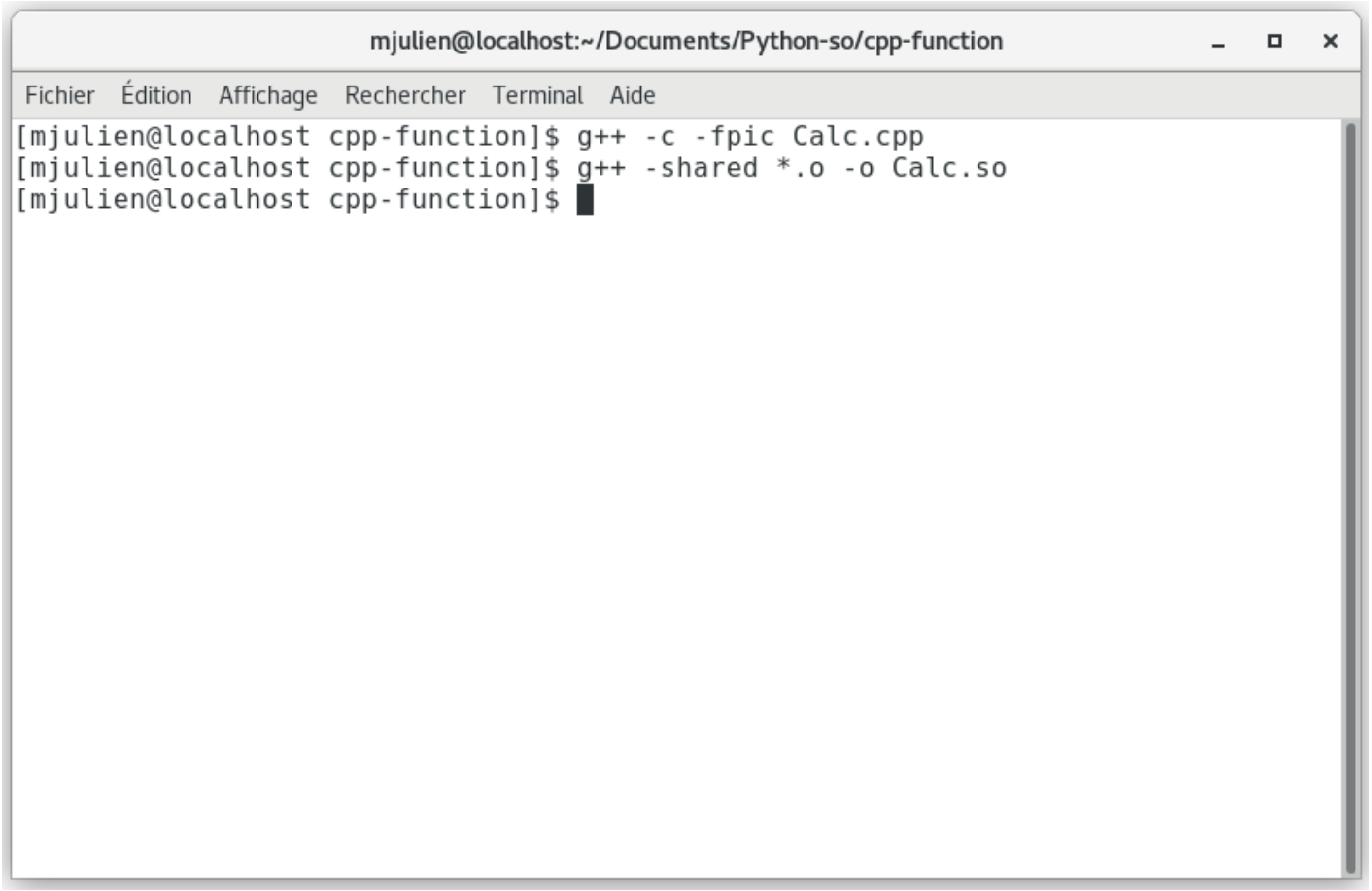
```
extern "C"
{
        double addition(double a, double b);
        double multiplication(double a, double b);
}
```

En-tête C/ObjC ▾ Largeur des tabulations : 8 ▾ Lig 5, Col 2 INS

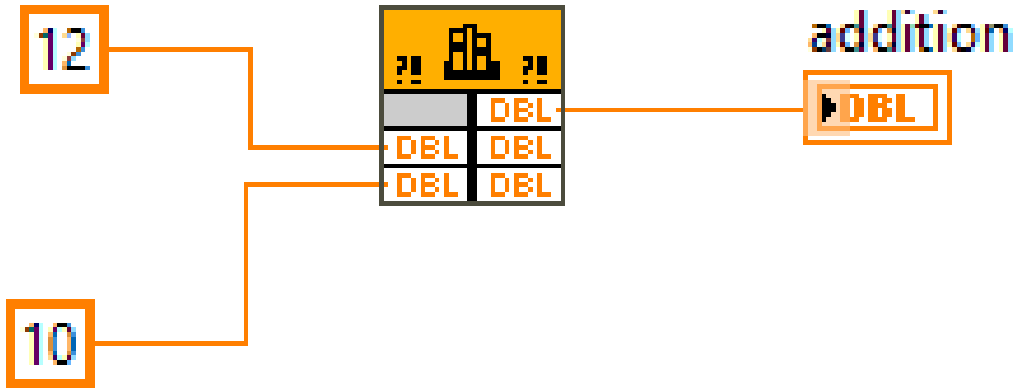
Ajout de l'instruction « extern C »

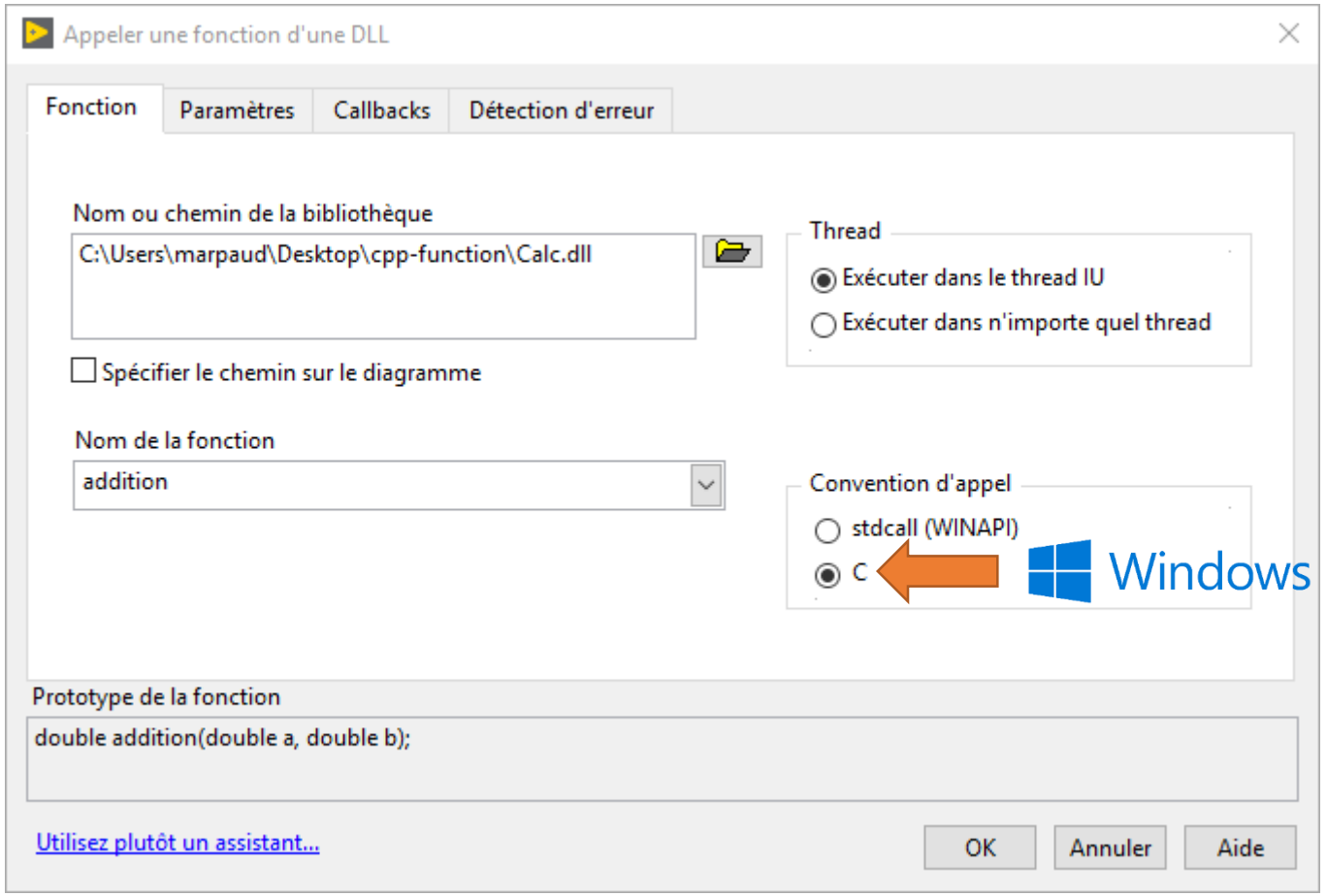
Fonctions C++ (2/3)





Fonctions C++ (3/3)

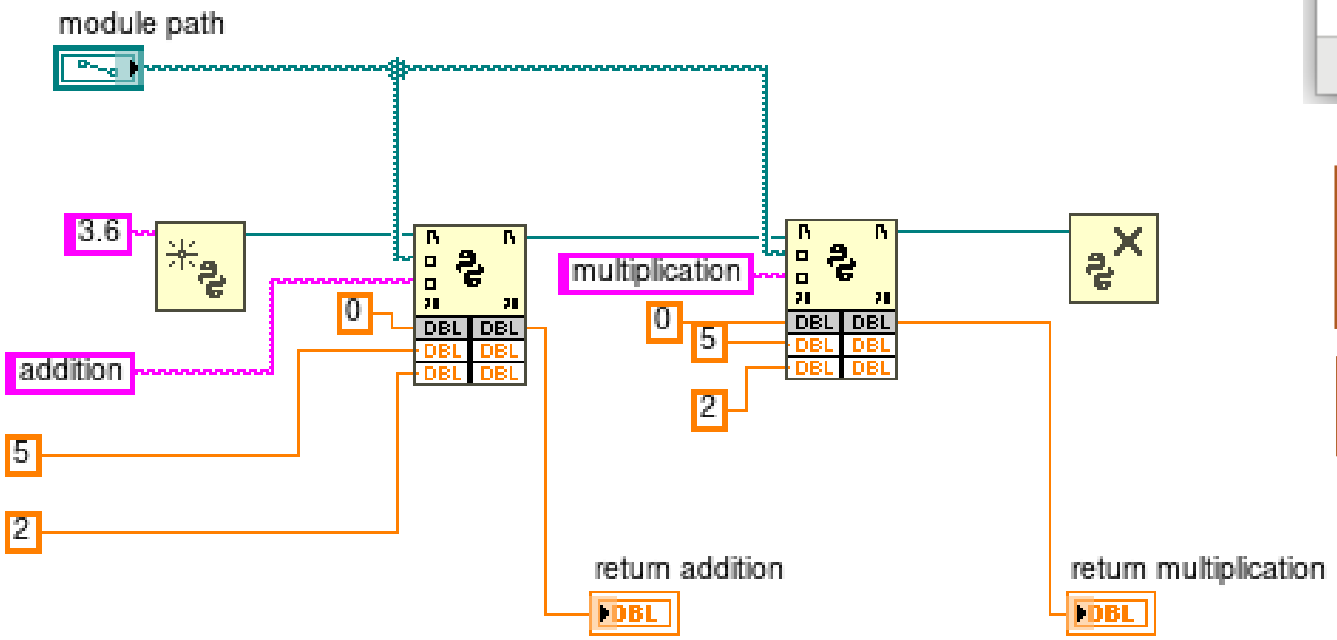




Fonctions Python dans LabVIEW

Rq: à partir de LabVIEW 2018

Fonctions Python (1/1)

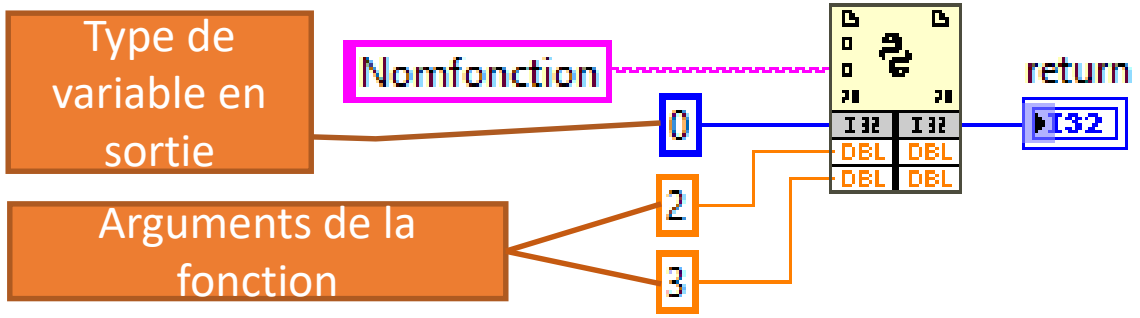


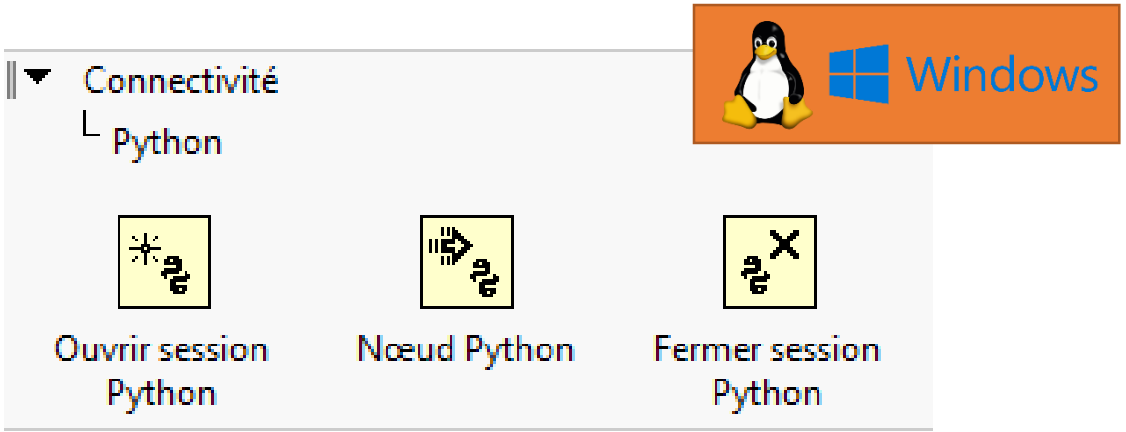
fonction.py
~/Documents/Python-Lab...

Ouvrir | Enregistrer

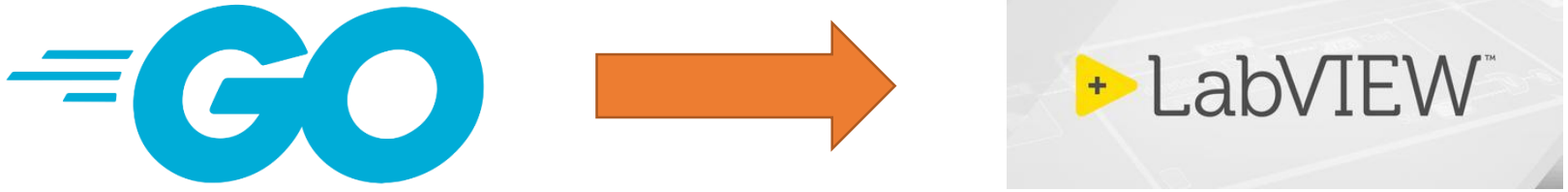
```
def addition(a, b):
        return a+b

def multiplication(a, b):
        return a*b
```

Python | Largeur des tabulations : 8 | Lig 5, Col 19 | INS





Go dans LabVIEW



Go dans LabVIEW (1/1)

mjulien@localhost:~/Documents/go

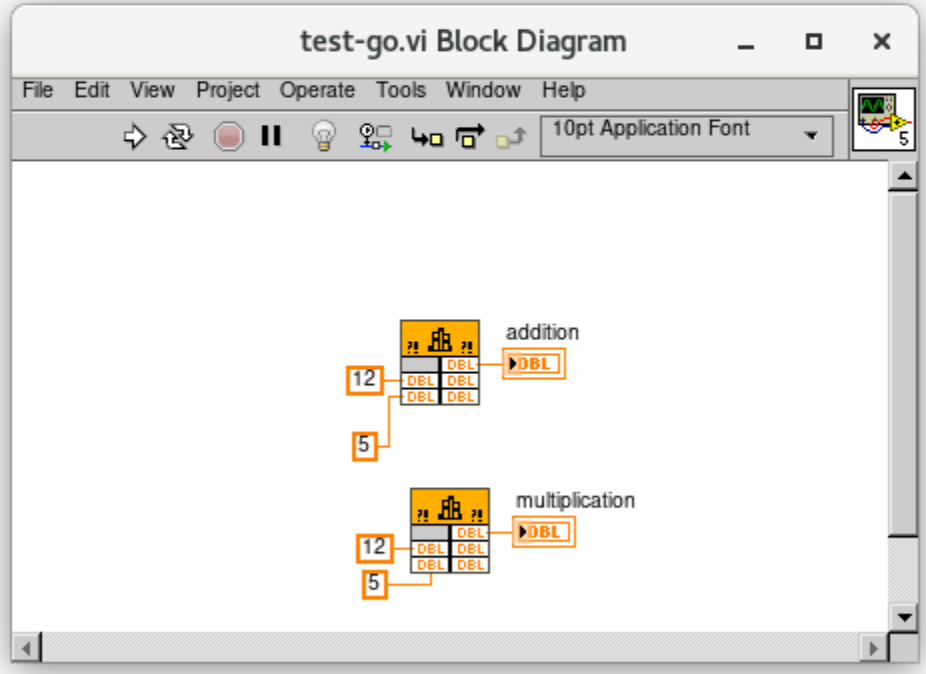
Fichier Édition Affichage Rechercher Terminal Aide

```
[mjulien@localhost go]$ go build -buildmode=c-shared -o test_go.so test.go
[mjulien@localhost go]$
```

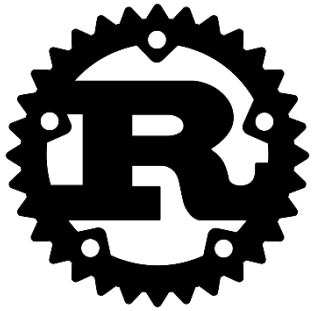
Ouvrir test.go ~/Documents... Enregistrer

```
package main

import "C"

//export addition
func addition(a, b float64) float64 {return a+b}

//export multiplication
func multiplication(a, b float64) float64 {return a*b}

func main() {}
```

Go Largeur des tabulations : 8 Lig 9, Col 42 INS



Rust dans LabVIEW



Rust dans LabVIEW (1/1)

Windows

lib.rs — ~/Documents/rust/Ca... — Ouvrir / Enregistrer

```
#[no_mangle]
pub extern fn addition(nb1: f64, nb2: f64) -> f64 {
        return nb1 + nb2;
}

#[no_mangle]
pub extern fn multiplication(nb1: f64, nb2: f64) -> f64 {
        return nb1 * nb2;
}
```

Cargo.toml — ~/Documents/rust/Calc — Ouvrir / Enregistrer

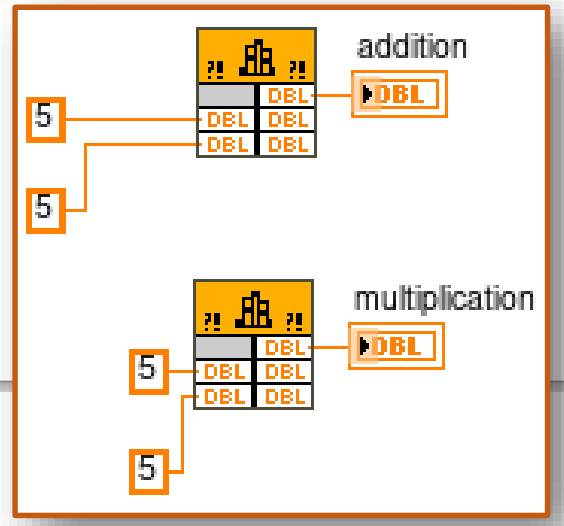
```
[package]
name = "Calc"
version = "0.1.0"
authors = ["mjulien"]
edition = "2018"

# See more keys and their definitions at https://doc.rust-lang.org/cargo/reference/
manifest.html

[dependencies]

[lib]
name = "Calc"
crate-type = ["dylib"]
```

TOML ▾ Largeur des tabulations : 8 ▾

mjulien@localhost:~/Documents/rust/Calc

Fichier Édition Affichage Rechercher Terminal Aide

```
[mjulien@localhost Calc]$ cargo build --release
   Compiling Calc v0.1.0 (/home/mjulien/Documents/rust/Calc)
warning: crate `Calc` should have a snake case name
  |
  = note: `#[warn(non_snake_case)]` on by default
  = help: convert the identifier to snake case: `calc`

    Finished release [optimized] target(s) in 0.60s
[mjulien@localhost Calc]$
```



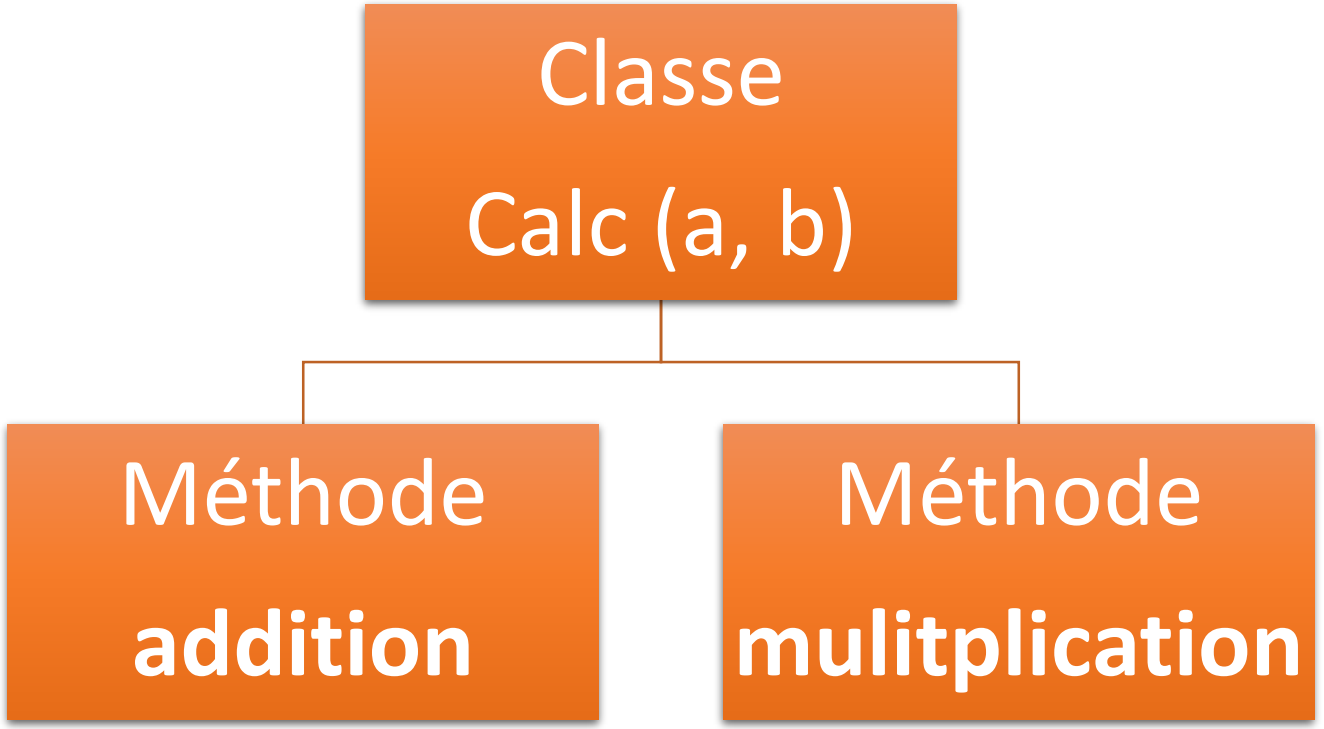
Voir les annexes pour Cargo et Rust

II. Interfaçage d'objet

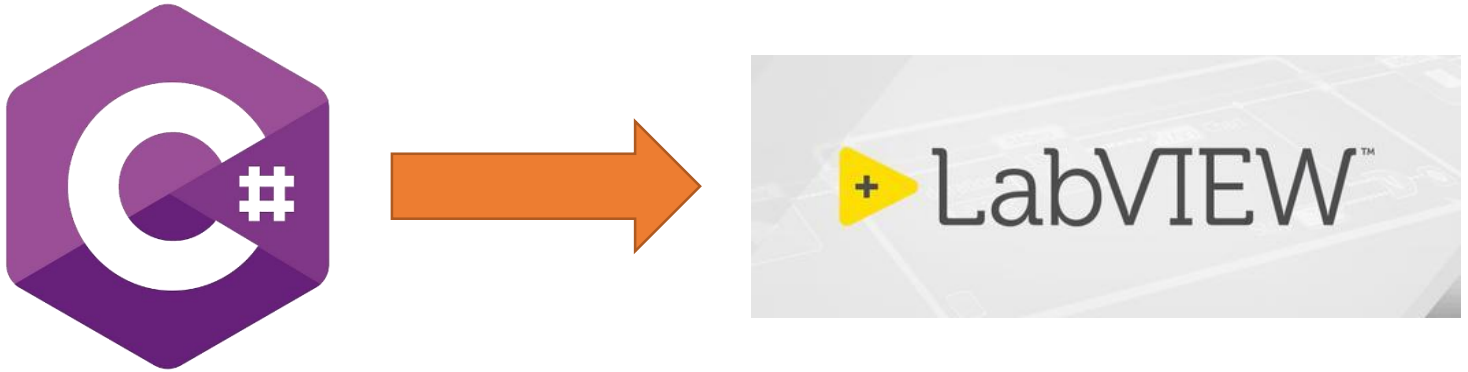
1. Objet
2. C#
3. Python
4. C++ vers Python
5. C++

Objet (1/1)

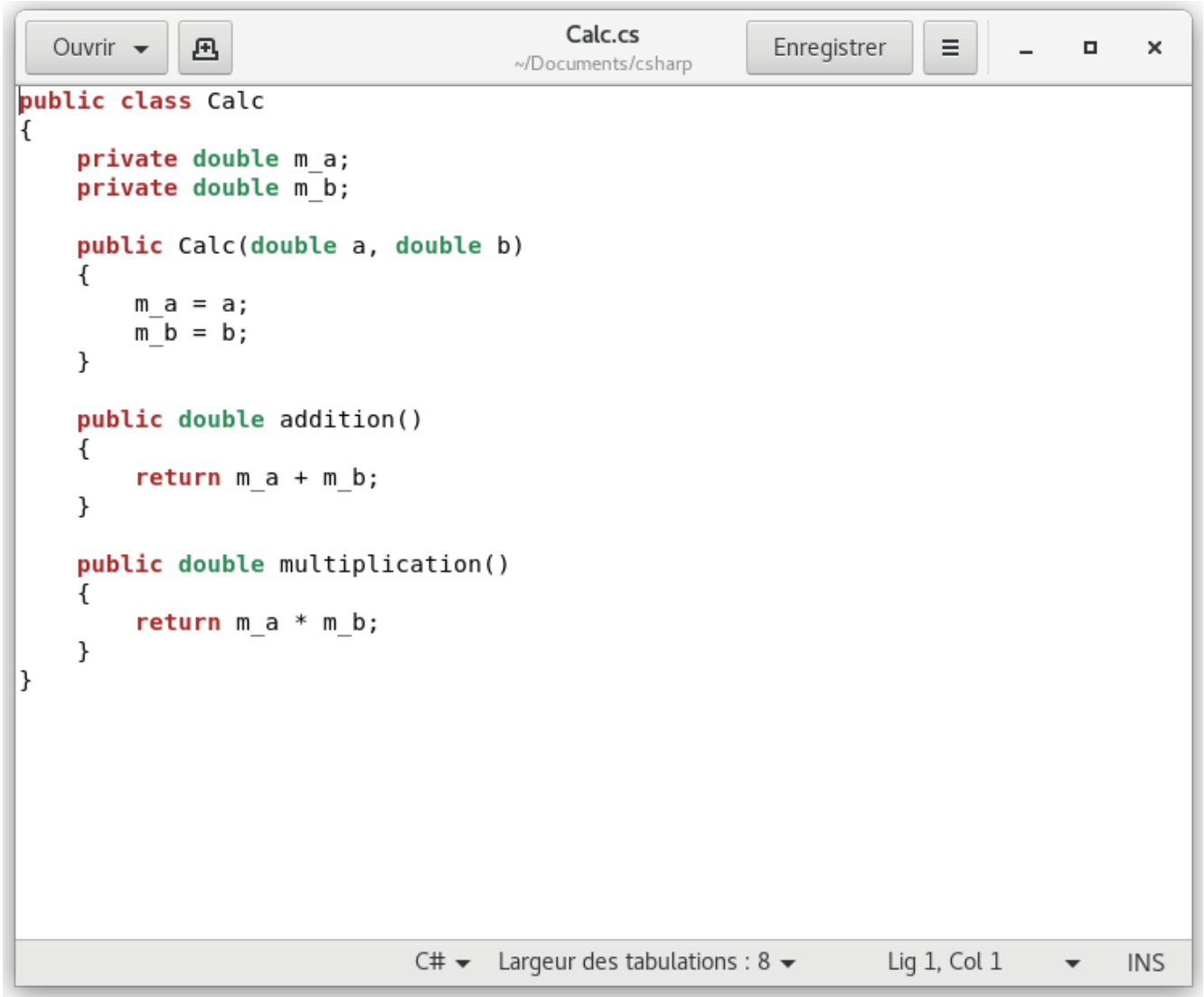
- Une classe **Calc** avec:
  - Une méthode **addition**
  - Une méthode **multiplication**



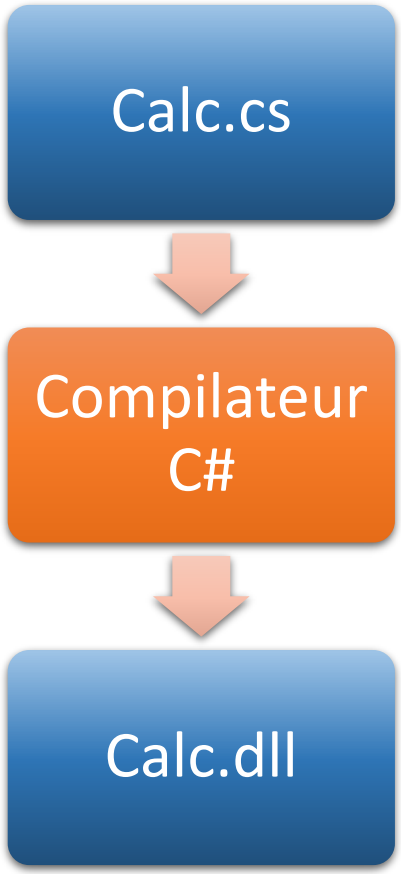
Objet C# dans LabVIEW

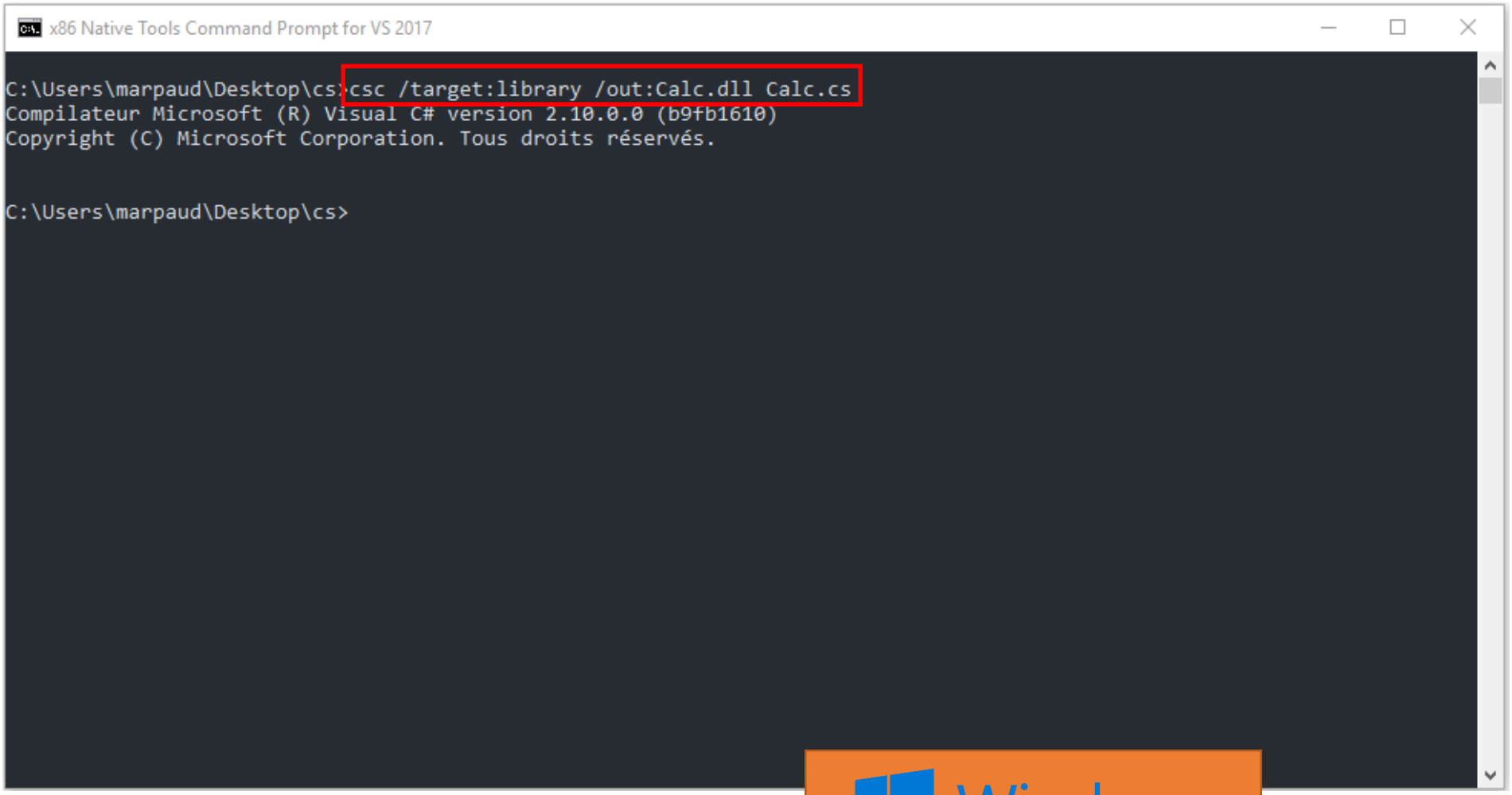


Objet C# dans LabVIEW (1/3)



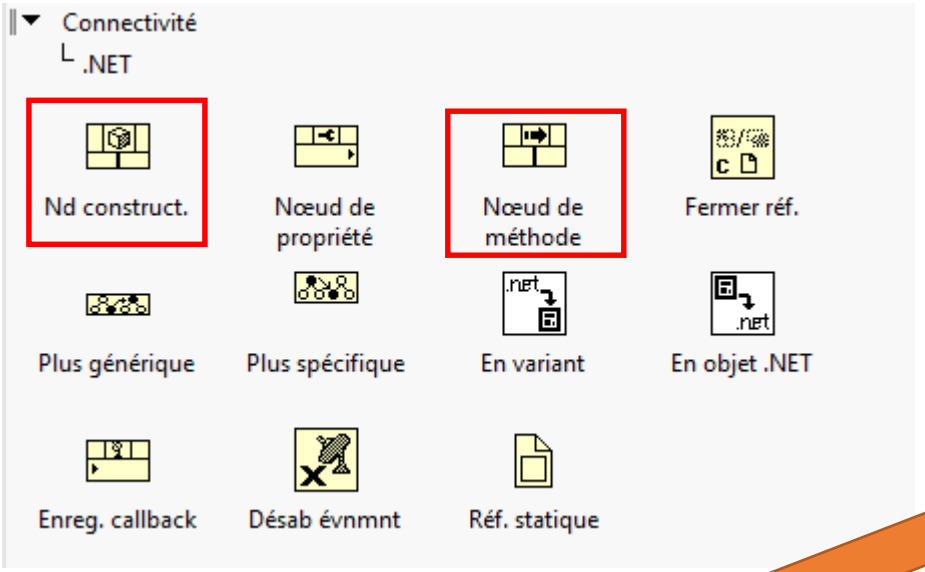
Objet C# dans LabVIEW (2/3)

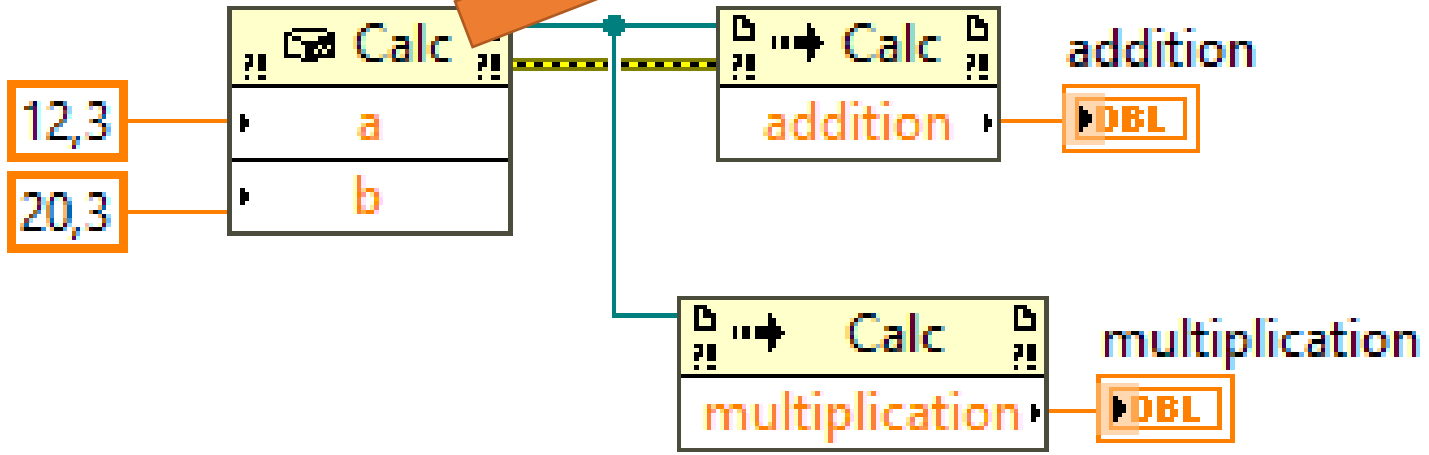


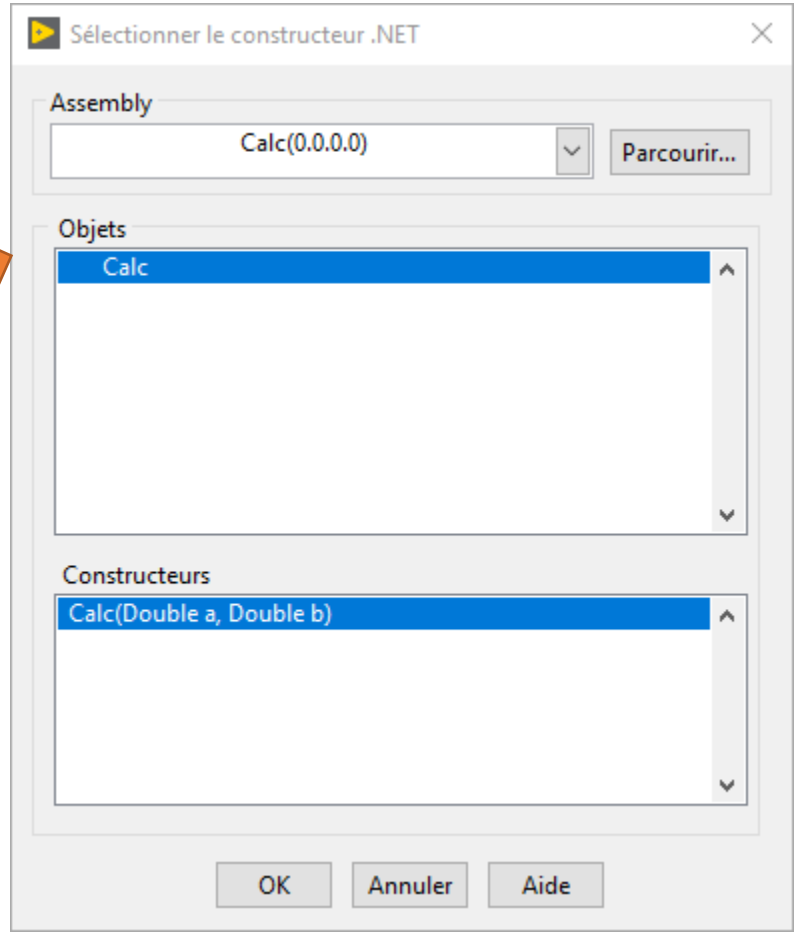


Objet **C#** dans LabVIEW (3/3)

Windows



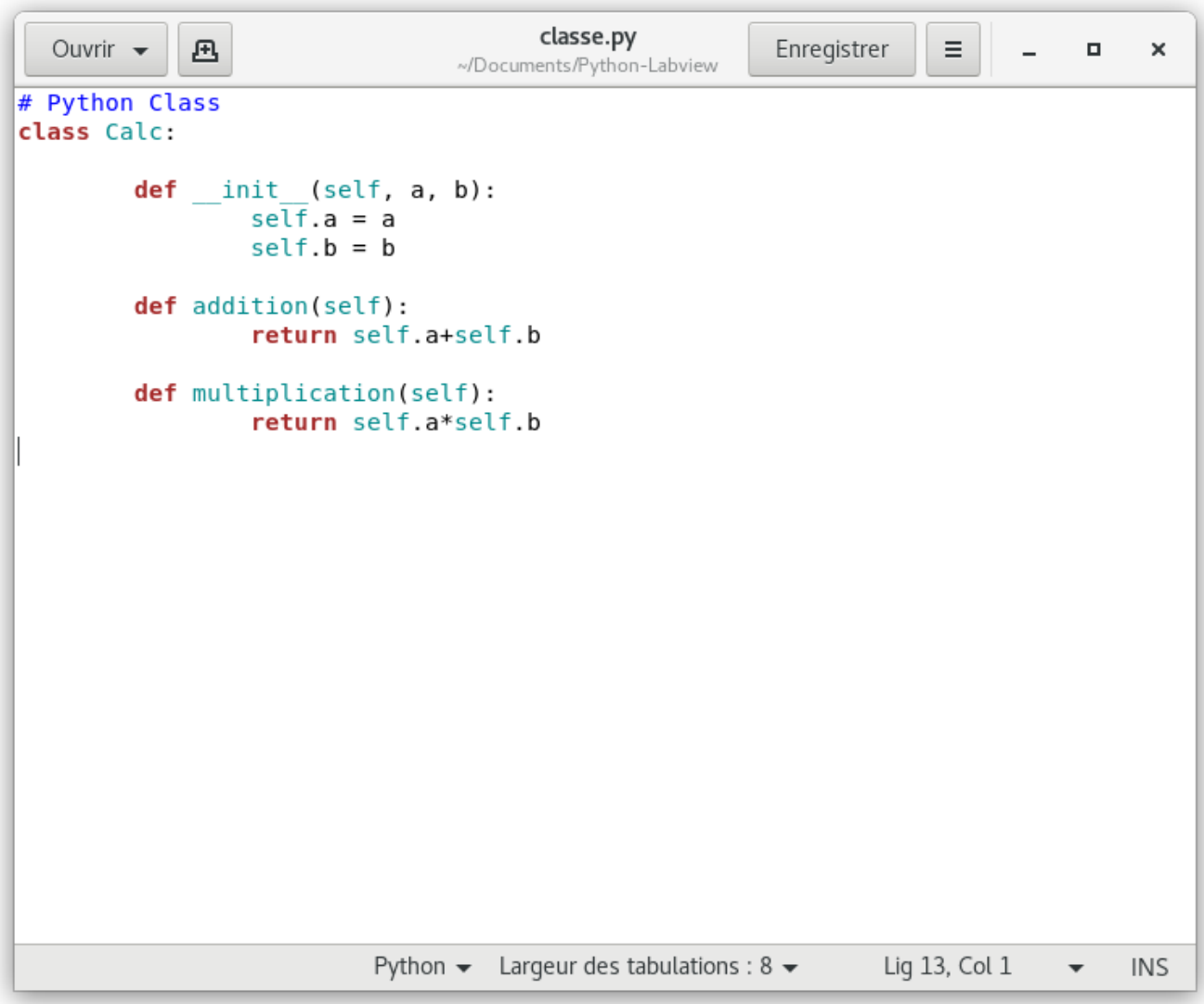




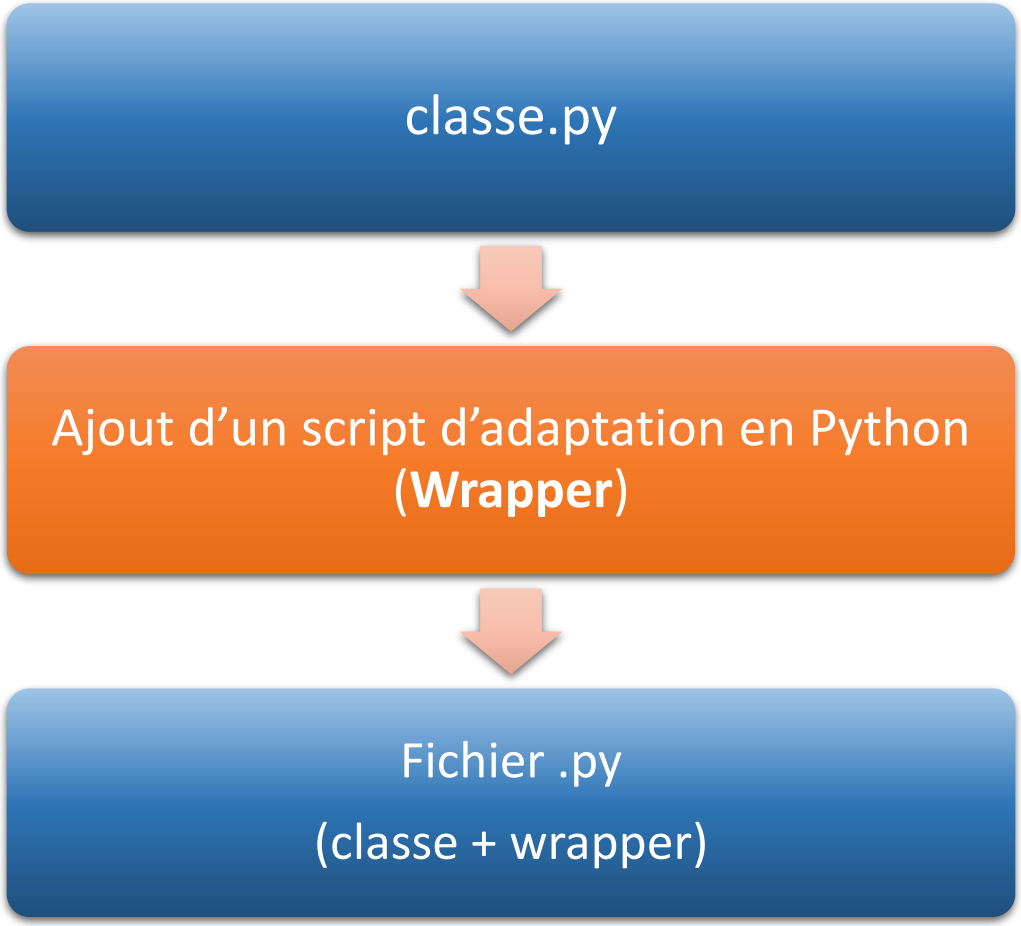
Objet Python dans LabVIEW

Rq: à partir de LabVIEW 2018

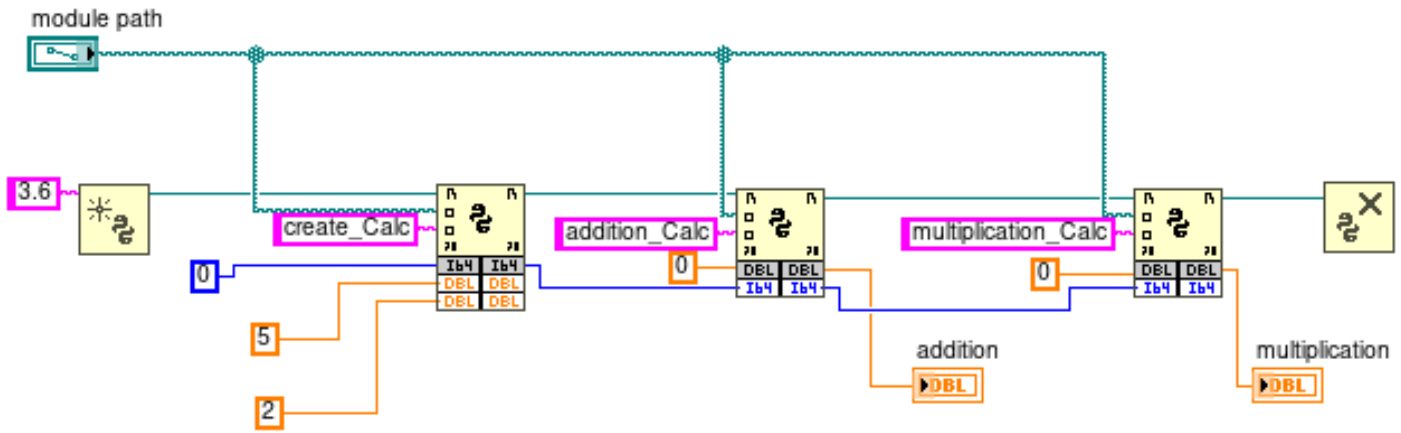
Objet Python (1/3)



Objet Python (2/3)



Objet Python (3/3)



Windows

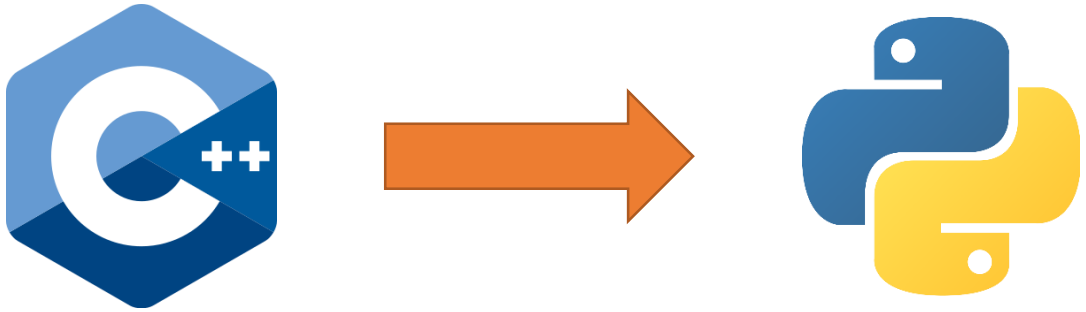
```
classe.py
~/Documents/Python-Labview
```

```python
# Python Class
class Calc:

        def __init__(self, a, b):
                self.a = a
                self.b = b

        def addition(self):
                return self.a+self.b

        def multiplication(self):
                return self.a*self.b
```

```python
# Python pour Labview
tabObject = {}
i = 0

def create_Calc(a, b):
        obj = Calc(a,b)
        global i
        global tabObject
        tabObject[i] = obj
        i = i + 1
        return i-1

def addition_Calc(index):
        global tabObject
        return tabObject[index].addition()

def multiplication_Calc(index):
        global tabObject
        return tabObject[index].multiplication()

def delete_Calc(index):
        global tabObject
        if index in tabObject:
                del tabObject[index]
                return True
        else:
                return False
```

Python — Largeur des tabulations : 8 — Lig 39, Col 29 — INS

C++ dans Python
(avec SWIG ou PyBind11)



C++ dans Python

- Utilisation de **SWIG** qui permet d'interfacer du C++ ou du C avec:
  - Python
  - Perl
  - C#
  - Java
  - Ruby
  - …
- Utilisation de **PyBind11** pour interfacer du C++ vers Python

C++ dans Python

Calc.cpp
~/Documents/Python-so/cpp

```cpp
#include "Calc.h"

Calc::Calc(double c, double d)
{
    a = c;
    b = d;
}

double Calc::addition()
{
    return a+b;
}

double Calc::multiplication()
{
    return a*b;
}
```

Calc.h
~/Documents/Python-so/cpp

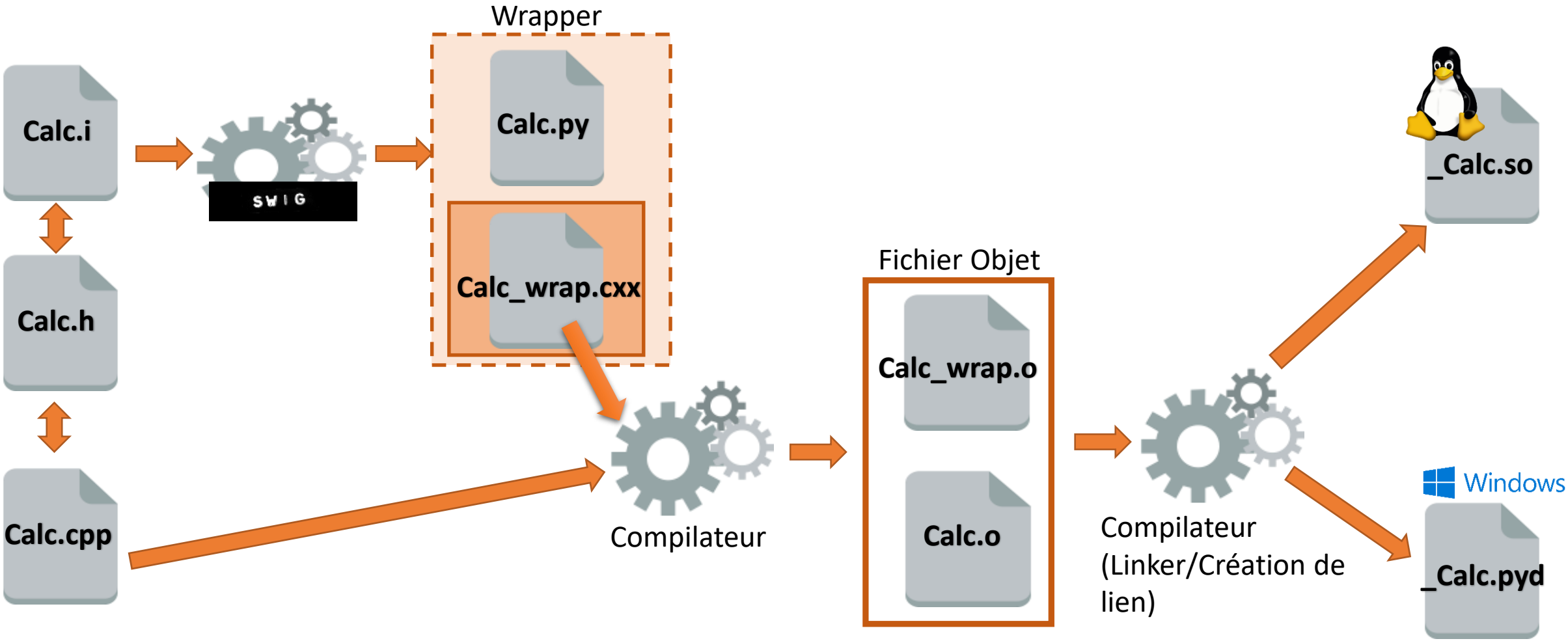
```cpp
class Calc
{
    public:
    Calc(double c, double d);
    double addition();
    double multiplication();

    private:
    double a;
    double b;
};
```

C++ dans Python

SWIG

(**S**implified **W**rapper and **I**nterface **G**enerator)

C++ dans Python (1/4)
Simplified Wrapper and Interface Generator



Rq: *.pyd = *.dll compatible avec la fonction import de Python

C++ dans Python (2/4)
Simplified Wrapper and Interface Generator

Ouvrir | Calc.i ~/Documents/Python-so/code | Enregistrer

```
%module Calc

%{
    #include "Calc.h"
%}

%include "Calc.h"
```

Le fichier .i utilisé par SWIG pour générer le wrapper
(SWIG générera le fichier **Calc_wrap.cxx** et **Calc.py**)

C++ dans Python (3/4)
Simplified Wrapper and Interface Generator

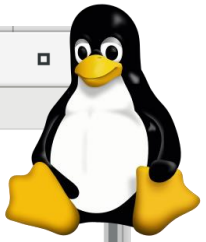
```
C:\Users\marpaud\Desktop\python-swig
λ C:\Users\marpaud\Desktop\cpp-cs\swig\swig.exe -python -c++ Calc.i

C:\Users\marpaud\Desktop\python-swig
λ g++ -c -fpic Calc_wrap.cxx Calc.cpp -I C:\Python27\include

C:\Users\marpaud\Desktop\python-swig
λ g++ -shared *.o -o _Calc.pyd -LC:\Python27\libs -lpython27
```

Windows

Utilisation de MinGW pour la compilation

```
IPython 5.7.0 -- An enhanced Interactive Python.
?         -> Introduction and overview of IPython's features.
%quickref -> Quick reference.
help      -> Python's own help system.
object?   -> Details about 'object', use 'object??' for extra details.

In [1]: import Calc

In [2]: obj = Calc.Calc(12.3,2.5)

In [3]: obj.addition()
Out[3]: 14.8

In [4]: obj.multiplication()
Out[4]: 30.75
```

C++ dans Python (4/4)
Simplified Wrapper and Interface Generator



IPython: Python-so/code

Fichier Édition Affichage Rechercher Terminal Aide

```
[mjulien@localhost code]$ swig -python -c++ Calc.i
[mjulien@localhost code]$ g++ -c -fpic Calc_wrap.cxx Calc.cpp -I/usr/include/python3.6m
[mjulien@localhost code]$ g++ -shared *.o -o _Calc.so
[mjulien@localhost code]$ ipython
Python 3.6.8 (default, Aug  7 2019, 17:28:10)
Type 'copyright', 'credits' or 'license' for more information
IPython 7.8.0 -- An enhanced Interactive Python. Type '?' for help.

In [1]: import Calc

In [2]: obj = Calc.Calc(12.3,15.6)

In [3]: obj.addition()
Out[3]: 27.9

In [4]: obj.multiplication()
Out[4]: 191.88

In [5]:
```

Rq: Les chemins de python sont pour CentOS 7

C++ dans Python

Avec PyBind11

C++ dans Python avec PyBind11 (1/2)

Calc.cpp

```cpp
#include <pybind11/pybind11.h>
#include "Calc.h"

namespace py = pybind11;
using namespace std;

Calc::Calc(double c, double d)
{
    a = c;
    b = d;
}

double Calc::addition()
{
    return a+b;
}

double Calc::multiplication()
{
    return a*b;
}

PYBIND11_MODULE(Calc, m) {
    py::class_<Calc>(m, "Calc")
        .def(py::init<double, double>())
        .def("addition", &Calc::addition)
        .def("multiplication", &Calc::multiplication);
}
```

Calc.h

```cpp
class Calc
{
    public:
    Calc(double c, double d);
    double addition();
    double multiplication();

    private:
    double a;
    double b;
};
```

C++ dans Python avec PyBind11 (2/2)

Cmder

```
Python 2.7.15 (v2.7.15:ca079a3ea3, Apr 30 2018, 16:22:17) [MSC v.1500 32 bit (Intel)]
Type "copyright", "credits" or "license" for more information.

IPython 5.7.0 -- An enhanced Interactive Python.
?         -> Introduction and overview of IPython's features.
%quickref -> Quick reference.
help      -> Python's own help system.
object?   -> Details about 'object', use 'object??' for extra details.

In [1]: import Calc

In [2]: obj = Calc.Calc(12.3,2.5)

In [3]: obj.multiplication()
Out[3]: 30.75

In [4]: obj.addition()
Out[4]: 14.8
```

Rq: Installer Pybind11 en utilisant:
pip install pybind11

```
g++ -O3 -Wall -shared -std=c++11 -fpic 'python3 -m pybind11 --includes' Calc.cpp -o Calc'python3-config --extension-suffix'
```

```
g++ -c -fpic Calc.cpp -I C:\Python27\include -D_hypot=hypot
g++ -shared *.o -o Calc.pyd -LC:\Python27\libs -lpython27
```

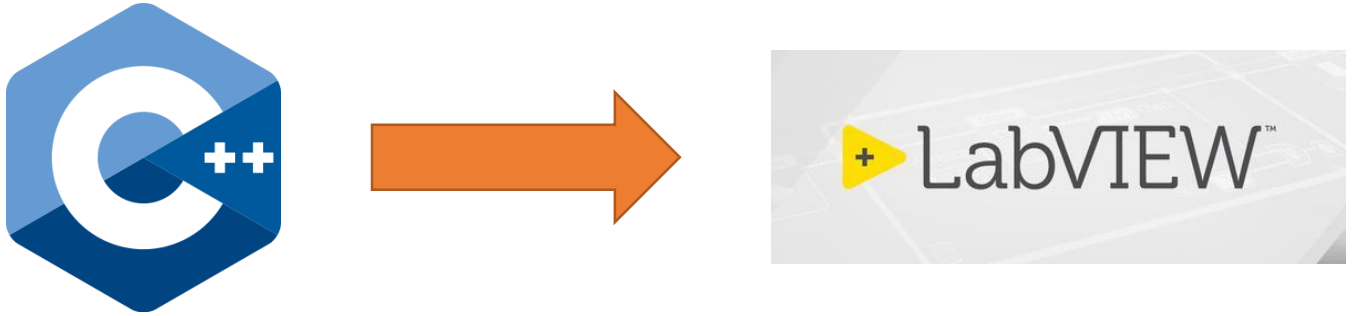
Windows — Avec g++

```
cl Calc.cpp /EHsc /I C:\Python27\include /LD C:\Python27\libs\python27.lib
```

Rq: renommer Calc.dll en Calc.pyd

Windows — Avec Microsoft Visual Studio

Objet C++ dans LabVIEW



Objet C++

Calc.cpp
~/Documents/Python-so/cpp

```cpp
#include "Calc.h"

Calc::Calc(double c, double d)
{
    a = c;
    b = d;
}

double Calc::addition()
{
    return a+b;
}

double Calc::multiplication()
{
    return a*b;
}
```

Calc.h
~/Documents/Python-so/cpp

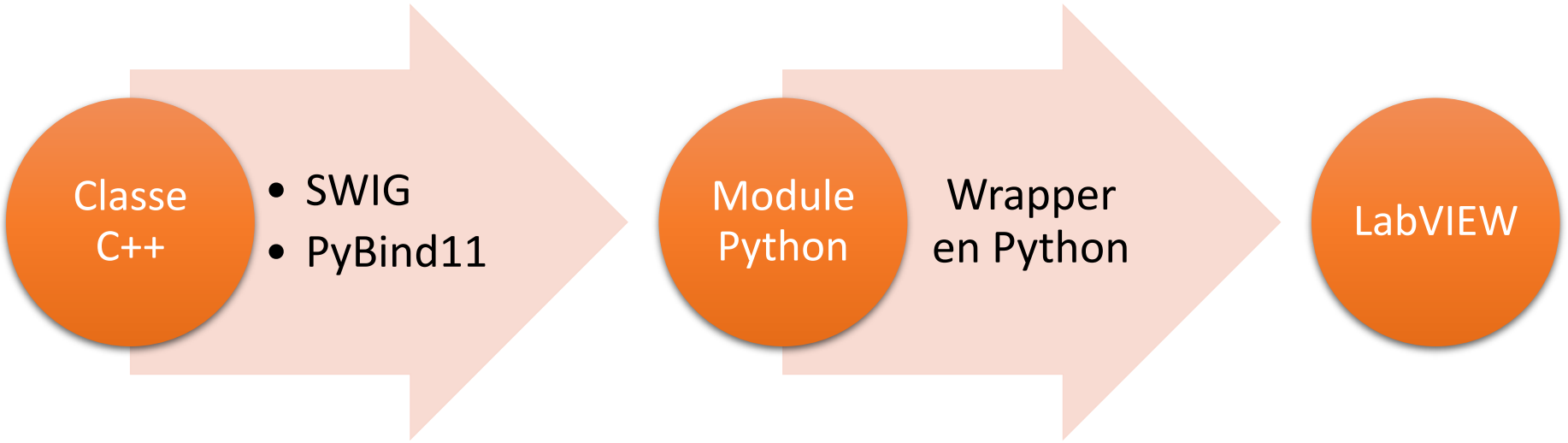
```cpp
class Calc
{
    public:
    Calc(double c, double d);
    double addition();
    double multiplication();

    private:
    double a;
    double b;
};
```

Objet C++

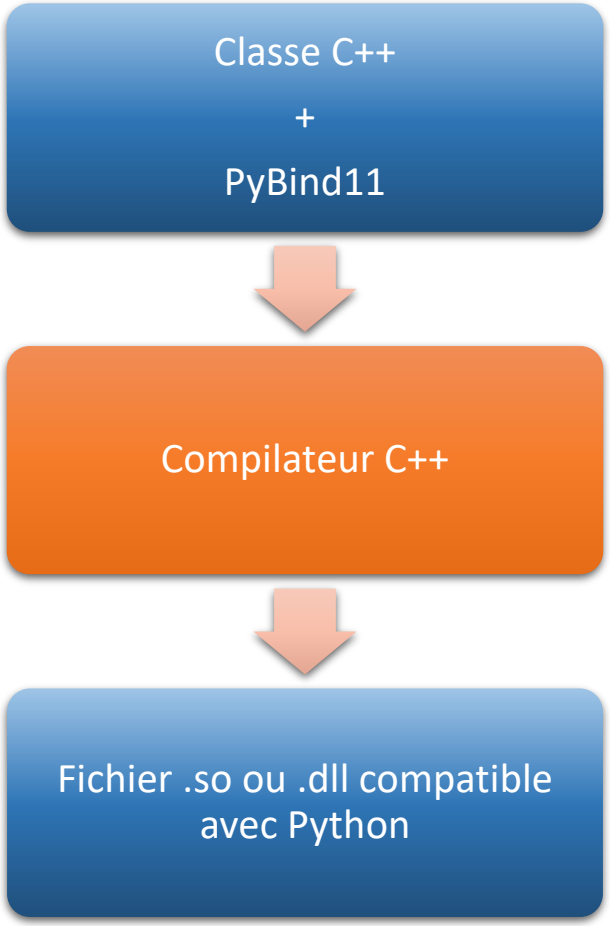
- Objet C++ via Python (avec PyBind11)
- Objet C++ via C# (avec SWIG)
- Création d’un wrapper pour interfacer un objet C++ avec LabVIEW

Objet C++

Objet C++ via Python (1/3)



Objet C++ via Python (2/3)

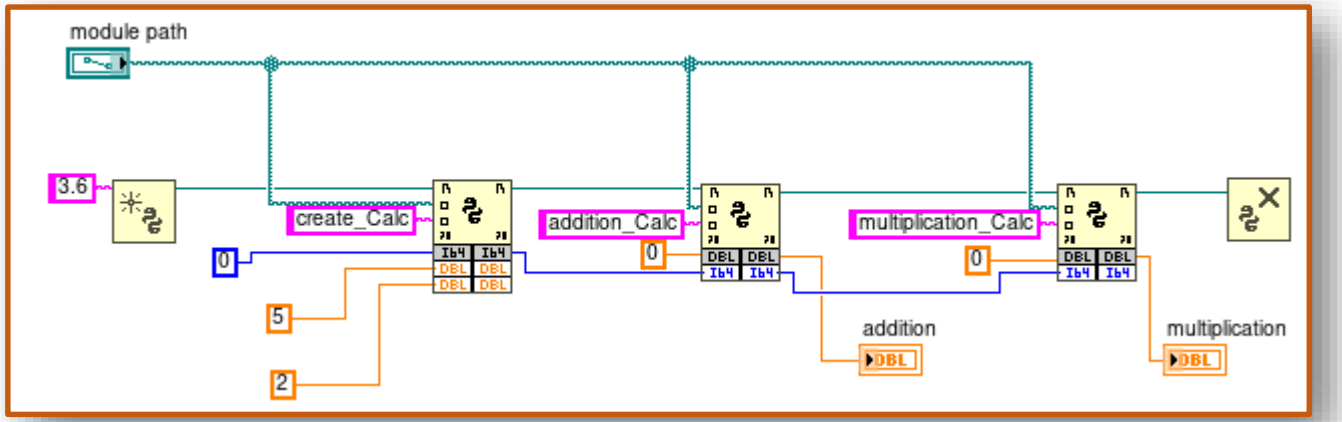


Objet C++ via Python (3/3)

Labview

Wrapper en Python (.py)

Fichier .so/.dll généré par Pybind11



Appelle de la librairie généré par PyBind11 (.dll ou .so)

Ouvrir — cpp-python-labview.py — ~/Documents/cpp-python-Labview — Enregistrer

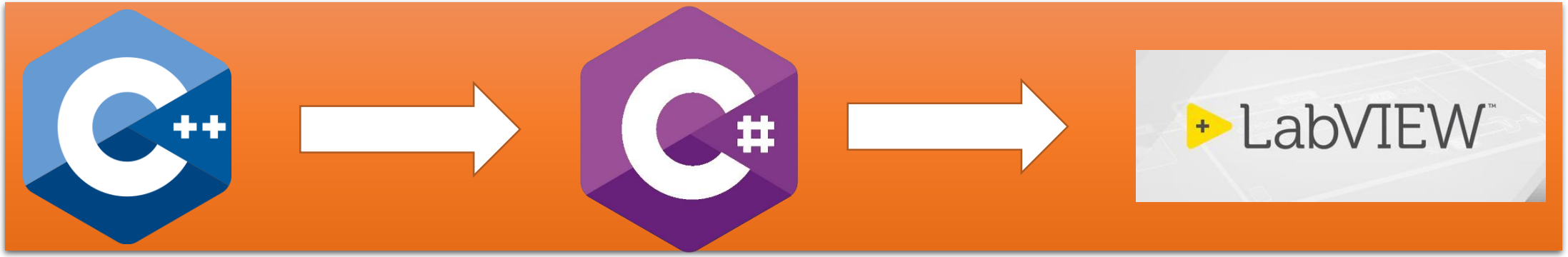
```
import Calc

# Python pour Labview
tabObject = {}
i = 0

def create_Calc(a, b):
        obj = Calc.Calc(a,b)
        global i
        global tabObject
        tabObject[i] = obj
        i = i + 1
        return i-1

def addition_Calc(index):
        global tabObject
        return tabObject[index].addition()

def multiplication_Calc(index):
        global tabObject
        return tabObject[index].multiplication()

def delete_Calc(index):
        global tabObject
        if index in tabObject:
                del tabObject[index]
                return True
        else:
                return False
```

Wrapper

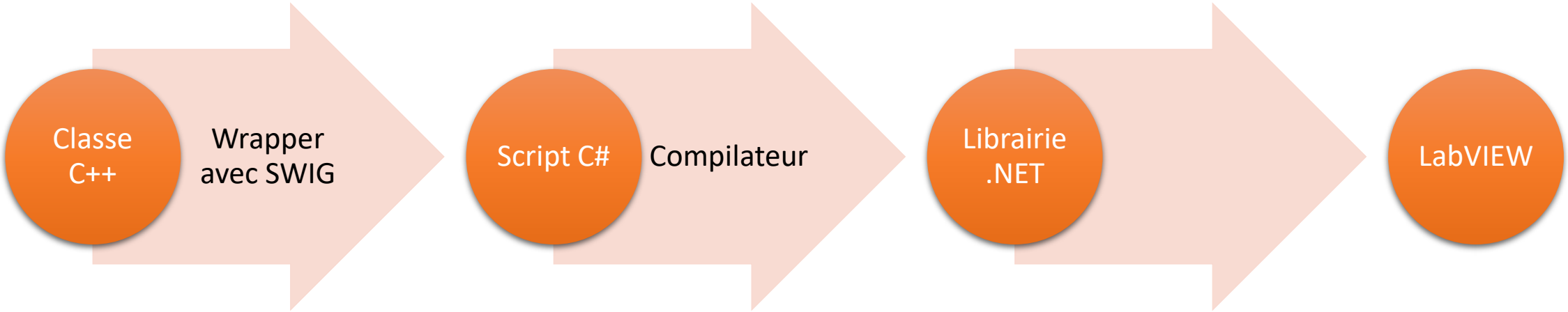
Python — Largeur — ol 20 — INS

Windows

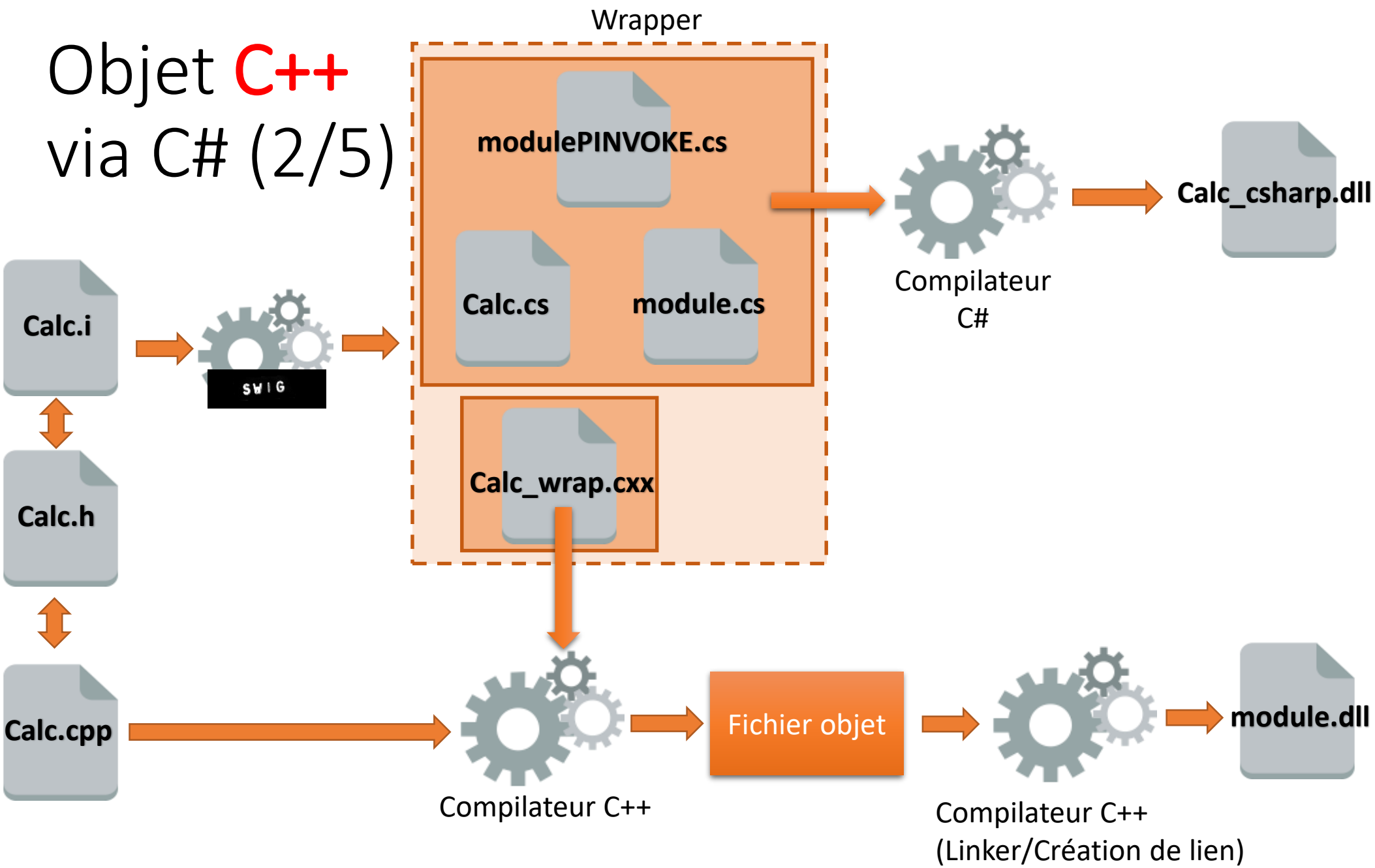
Objet C++



Objet C++ via C# (1/5)



Objet C++ via C# (2/5)



Objet C++ via C# (3/5)

Calc.i pour générer le wrapper avec SWIG


```
1  %module example
2
3  %{
4      #include "Calc.h"
5  %}
6
7  %include "Calc.h"
```

SWIG n'autorise pas que le module est le même nom que la classe en C#

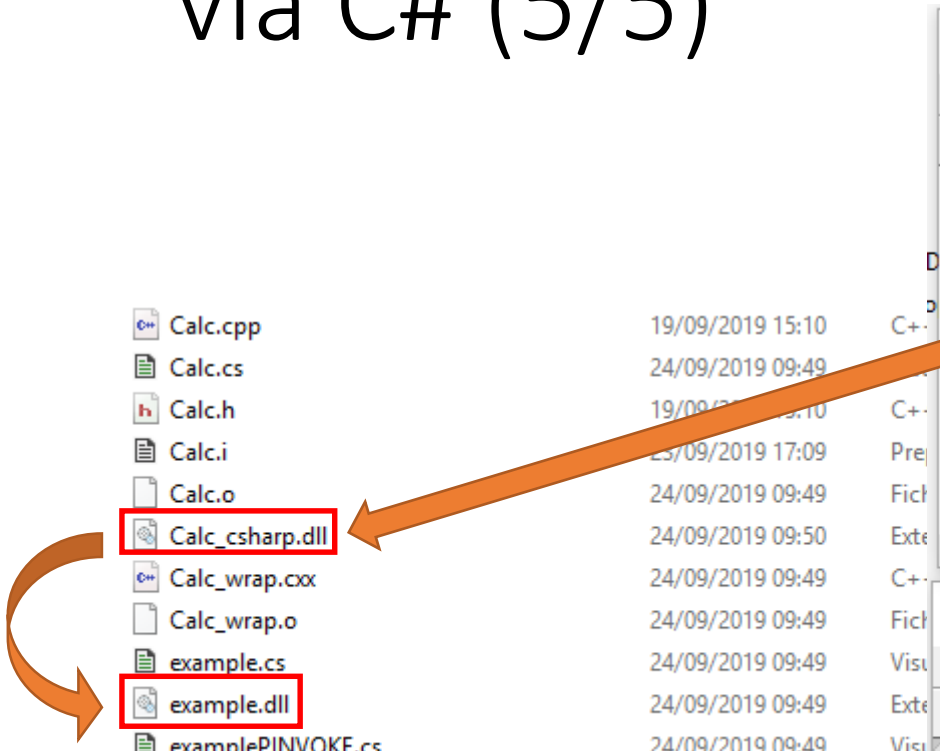
Objet C++ via C# (4/5)

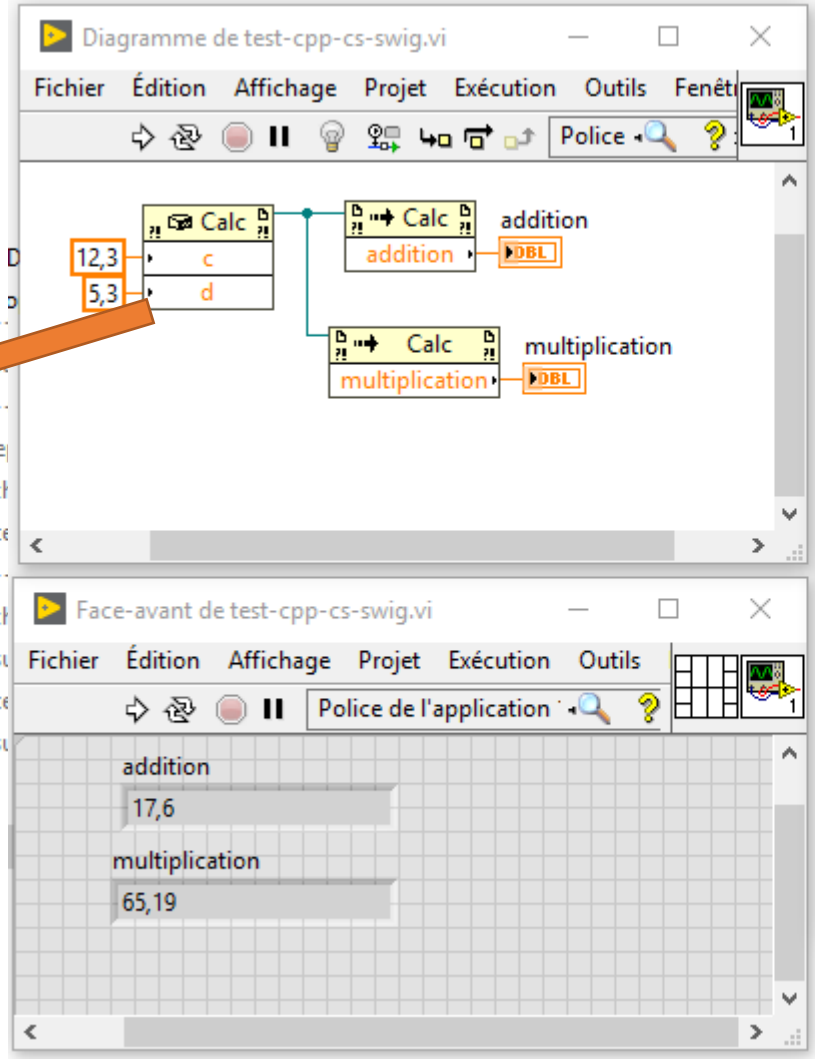
```
C:\Users\marpaud\Desktop\cpp-cs
λ C:\Users\marpaud\Desktop\cpp-cs\swig\swig.exe -csharp -c++ Calc.i

C:\Users\marpaud\Desktop\cpp-cs
λ g++ -c -fpic Calc_wrap.cxx Calc.cpp

C:\Users\marpaud\Desktop\cpp-cs
λ g++ -shared -Wl,--add-stdcall-alias *.o -o example.dll

C:\Users\marpaud\Desktop\cpp-cs
λ csc /target:library /out:Calc_csharp.dll *.cs
Compilateur Microsoft (R) Visual C# version 2.10.0.0 (b9fb1610)
Copyright (C) Microsoft Corporation. Tous droits réservés.


C:\Users\marpaud\Desktop\cpp-cs
```

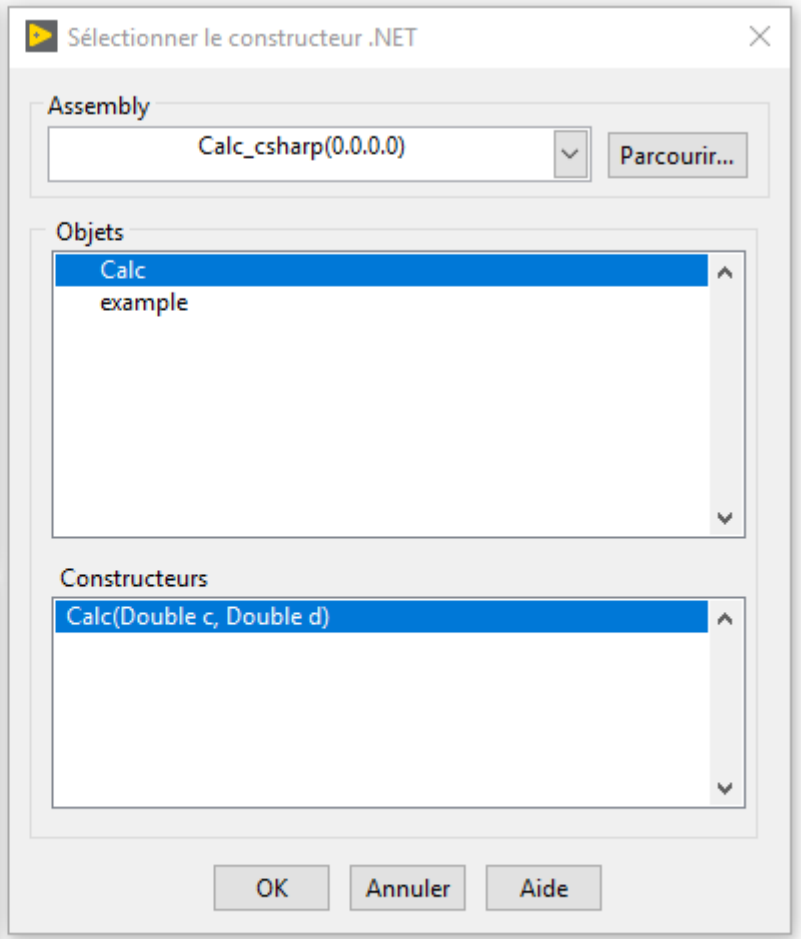
Génération du wrapper avec SWIG

Objet C++ via C# (5/5)

Windows



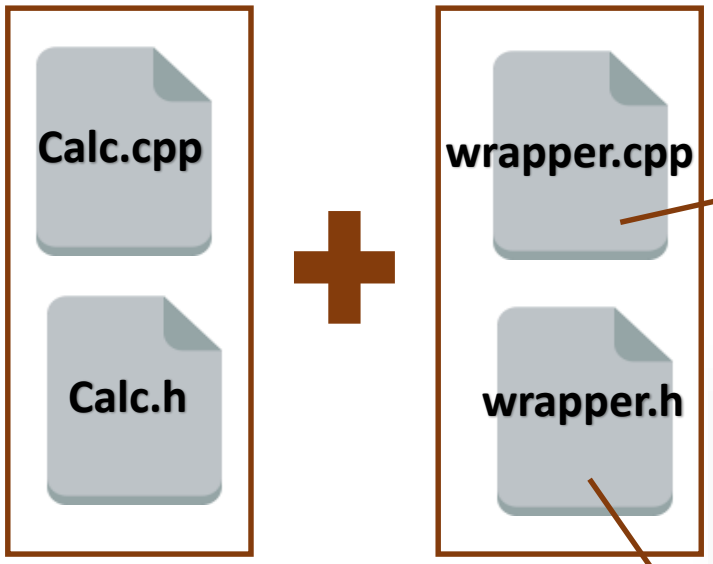




Objet C++

Création d’un wrapper pour interfacer un objet C++ avec LabVIEW

Objet C++
C++ avec un wrapper (1/3)

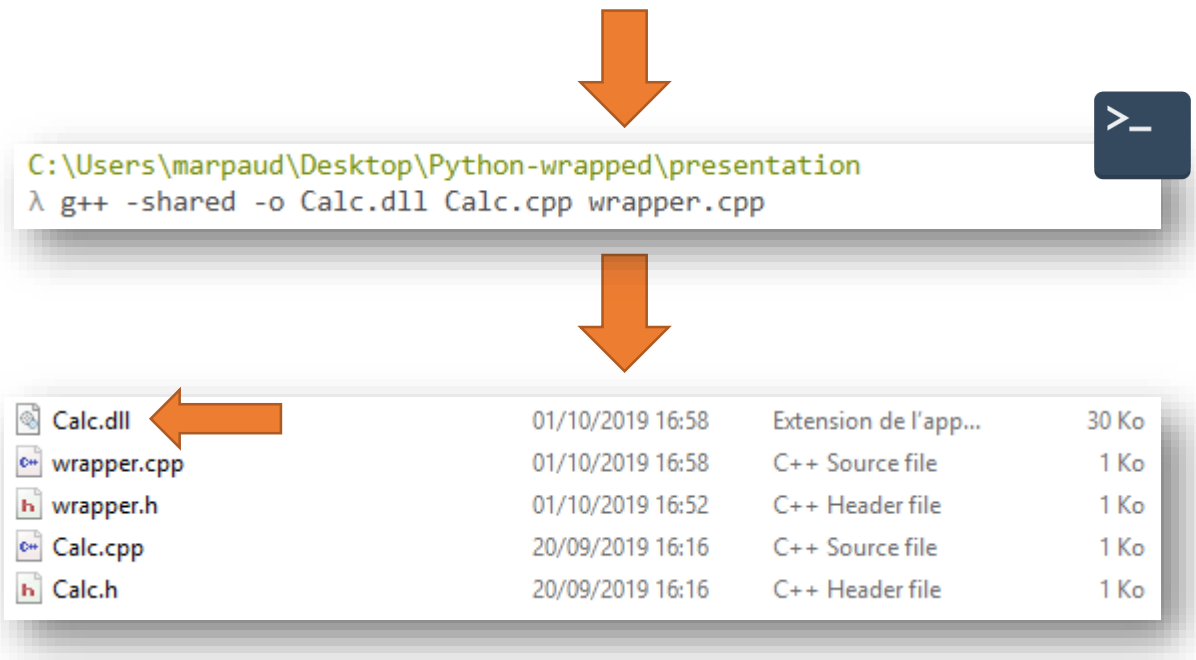



```h
#ifndef __WRAPPER_H__
#define __WRAPPER_H__

#ifdef __cplusplus
extern "C" {
#endif

struct wrapper;
typedef struct wrapper wrapper_t;

wrapper_t *Calc_create(double a, double b);
void Calc_destroy(wrapper_t *m);

double Calc_addition(wrapper_t *m);
double Calc_multiplication(wrapper_t *m);

#ifdef __cplusplus
}
#endif

#endif /* __WRAPPER_H__ */
```

```cpp
#include <stdlib.h>
#include "wrapper.h"
#include "Calc.h"

struct wrapper {
    void *obj;
};

wrapper_t *Calc_create(double a, double b)
{
    wrapper_t *m;
    Calc *obj;

    m      = (typeof(m))malloc(sizeof(*m));
    obj    = new Calc(a, b);
    m->obj = obj;

    return m;
}

void Calc_destroy(wrapper_t *m)
{
    if (m == NULL)
        return;
    delete static_cast<Calc *>(m->obj);
    free(m);
}

double Calc_addition(wrapper_t *m)
{
    Calc *obj;

    if (m == NULL)
        return 0;

    obj = static_cast<Calc *>(m->obj);
    return obj->addition();
}

double Calc_multiplication(wrapper_t *m)
{
    Calc *obj;

    if (m == NULL)
        return 0;

    obj = static_cast<Calc *>(m->obj);
    return obj->multiplication();
}
```

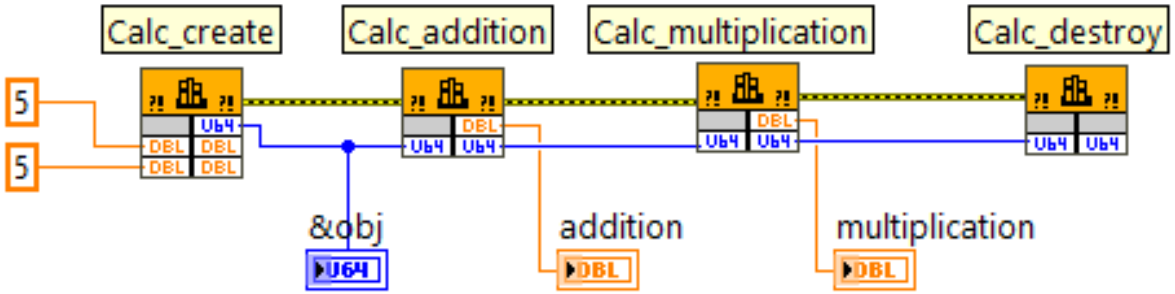
Objet C++
C++ avec un wrapper (2/3)

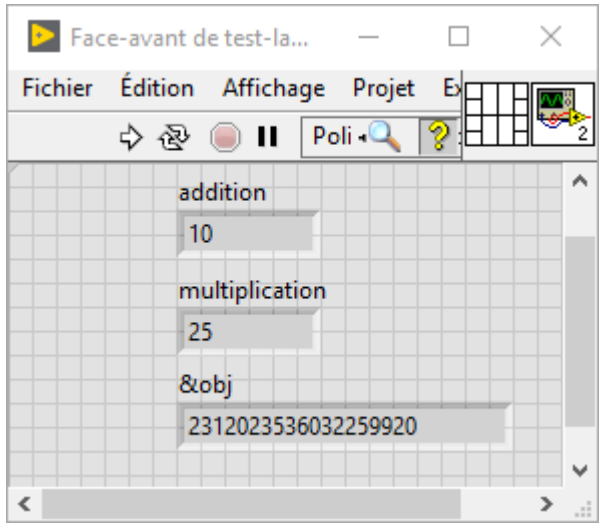
- Calc.cpp
- wrapper.cpp

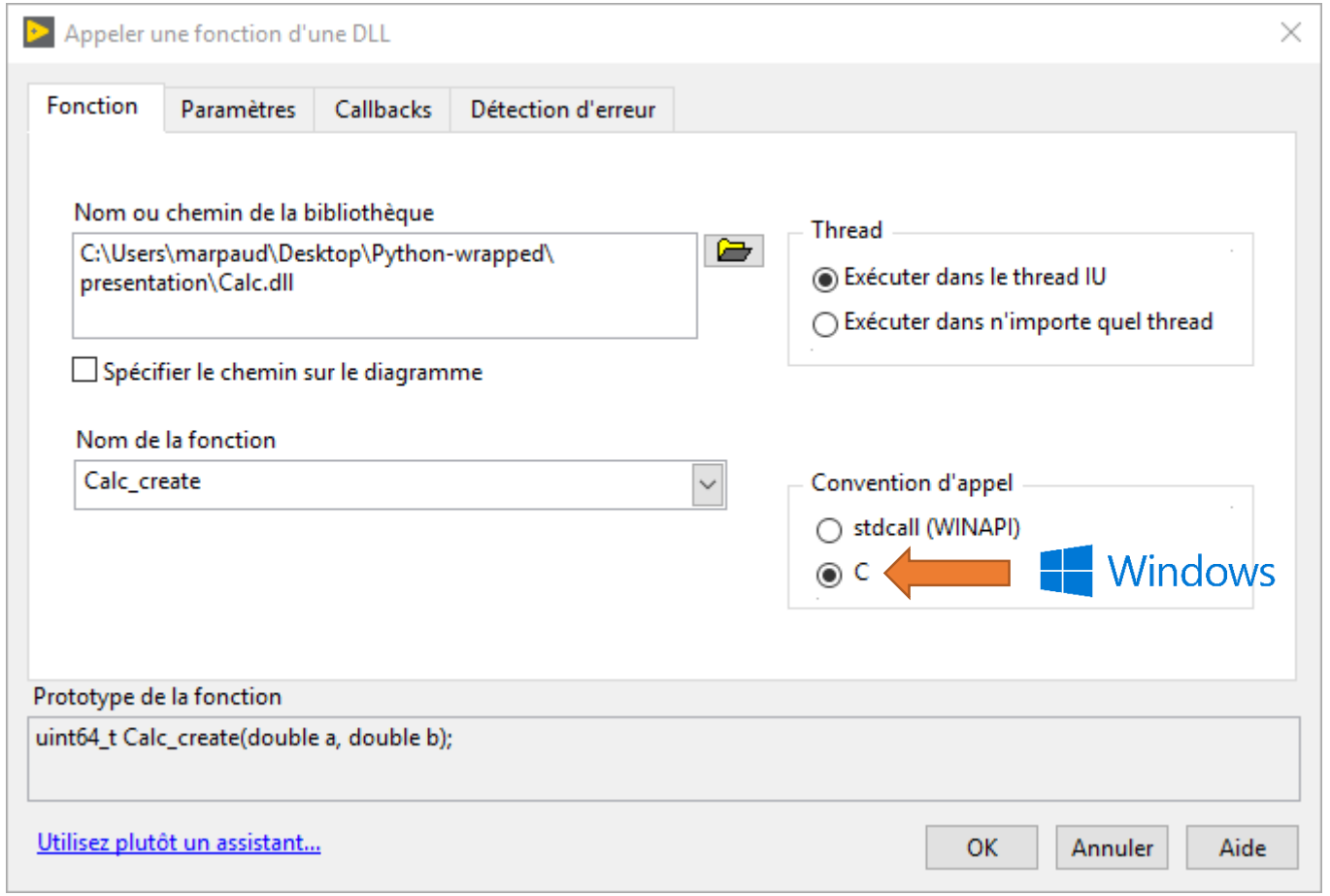


Rq: Sur Linux remplacer « Calc.dll » par « Calc.so »

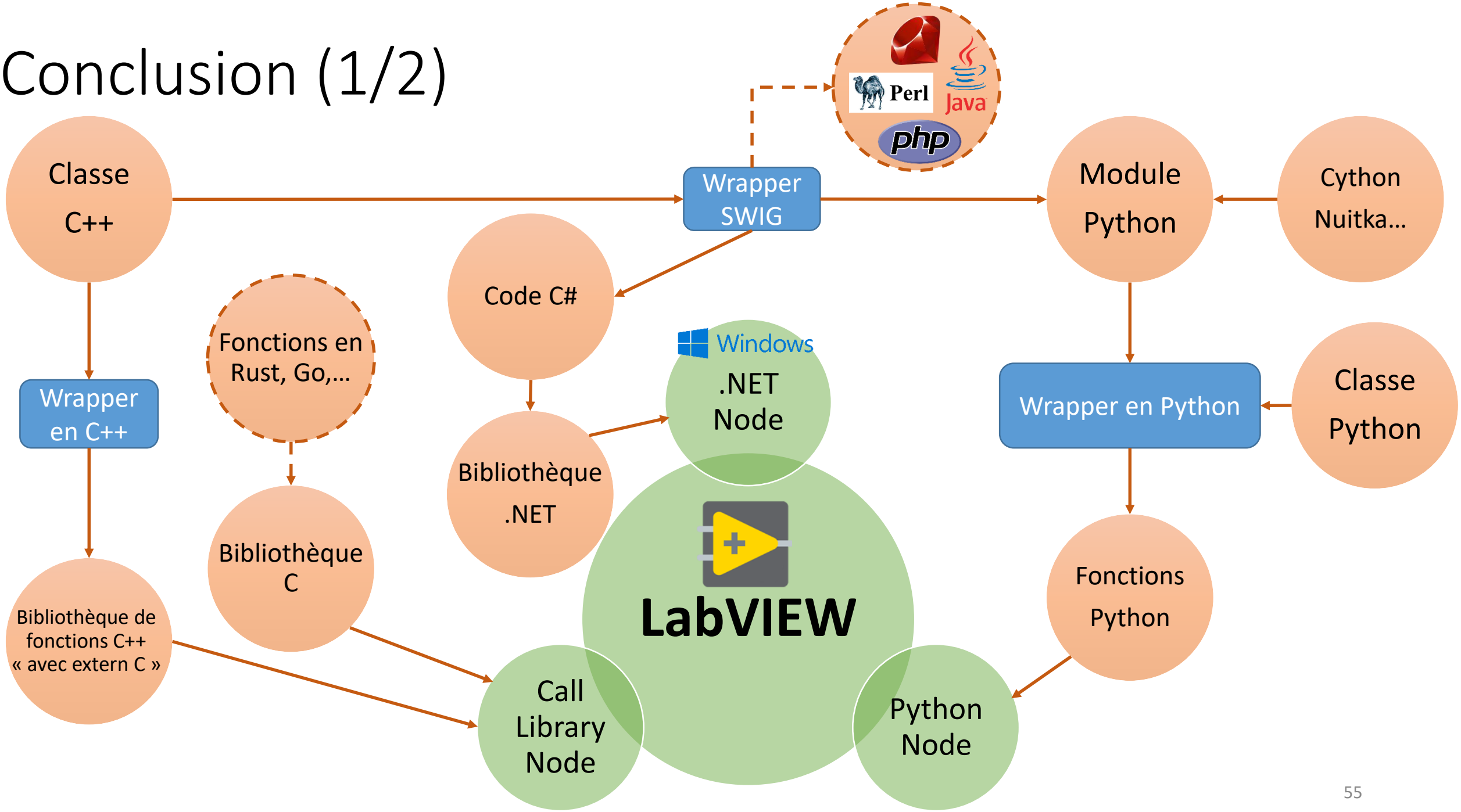
Objet C++
C++ avec un wrapper (3/3)



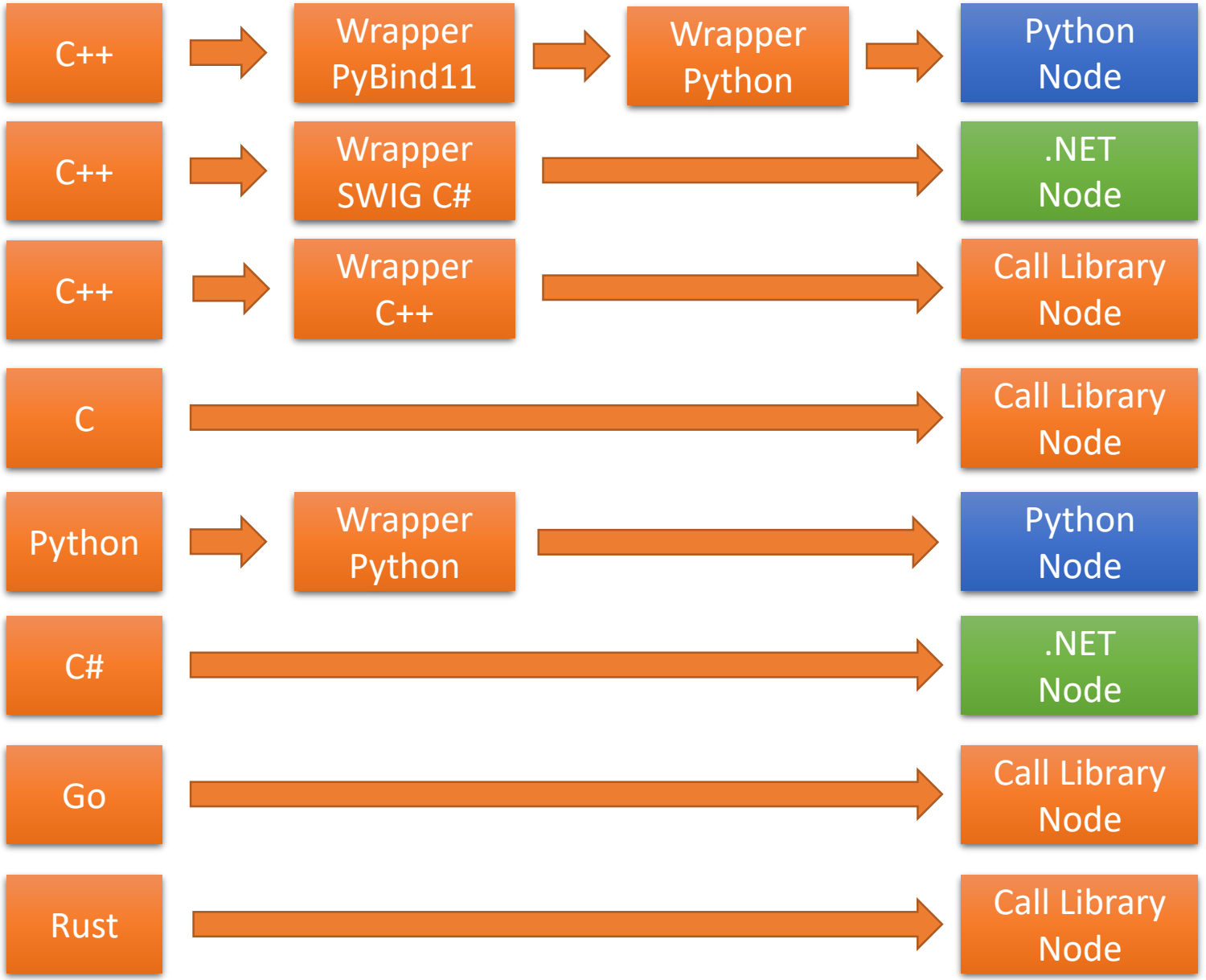




Conclusion (1/2)



Conclusion (2/2)



Référence

- SWIG: http://www.swig.org/
- PyBind11:
  - https://github.com/pybind/pybind11
  - https://pybind11.readthedocs.io/en/master/
- Wrapper C/C++: https://nachtimwald.com/2017/08/18/wrapping-c-objects-in-c/
- Python avec C/C++: https://docs.python.org/fr/3/extending/building.html
- Python et ctypes: https://deptinfo-ensip.univ-poitiers.fr/ENS/doku/doku.php/stu:progc:ctypes
- Go:
  - https://medium.com/learning-the-go-programming-language/calling-go-functions-from-other-languages-4c7d8bcc69bf
  - https://golang.org/
- Rust: https://doc.rust-lang.org/1.2.0/book/rust-inside-other-languages.html
- Microsoft Visual Studio:
  - https://docs.microsoft.com/en-us/cpp/build/reference/compiler-options-listed-by-category?view=vs-2019
  - https://docs.microsoft.com/fr-fr/cpp/build/walkthrough-compiling-a-native-cpp-program-on-the-command-line?view=vs-2019
  - https://docs.microsoft.com/fr-fr/cpp/build/walkthrough-compile-a-c-program-on-the-command-line?view=vs-2019

Annexe
Installer g++ et gcc sur Windows (1/2)

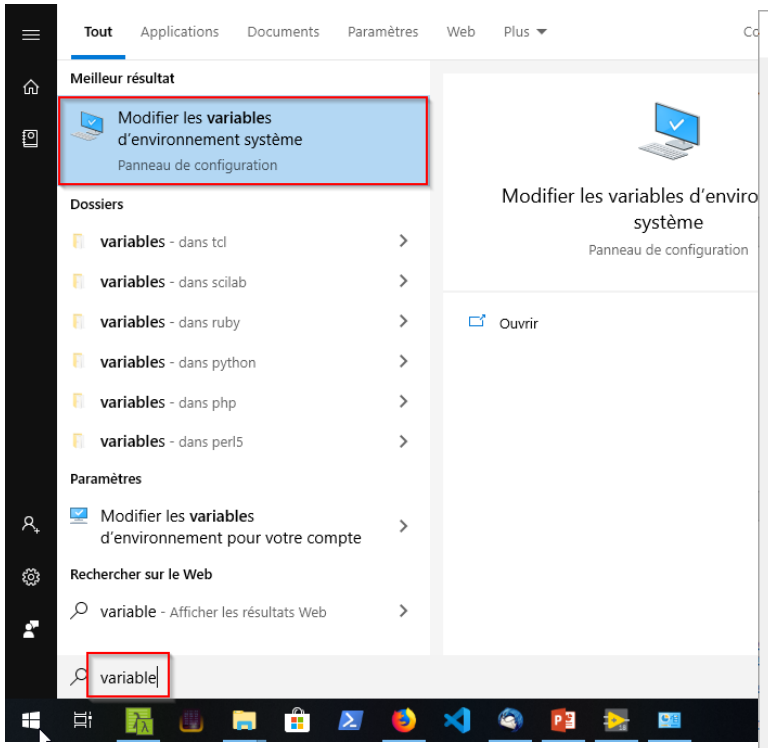
Aller sur http://mingw-w64.org/doku.php/download, télécharger et installer la version MingW-W64-builds

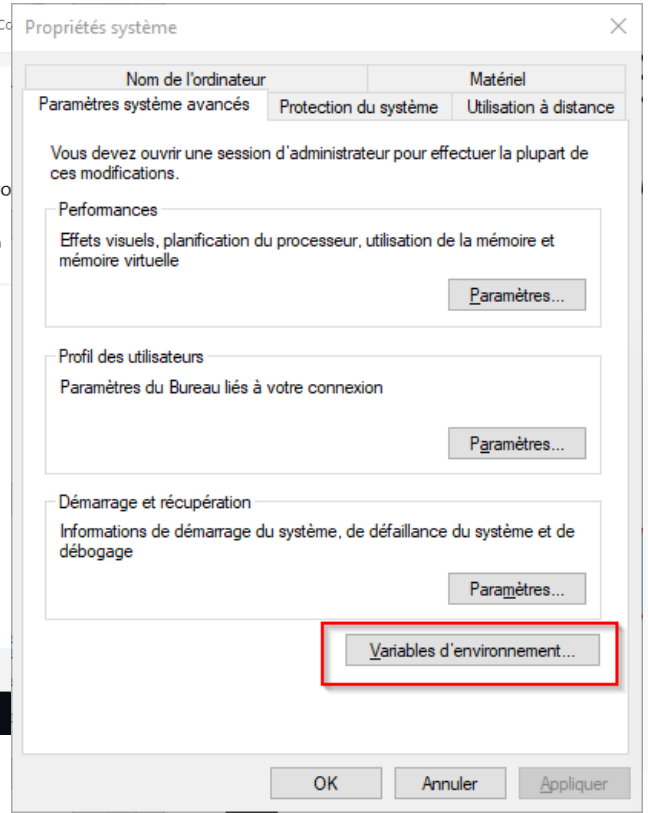
Pre-built toolchains and packages

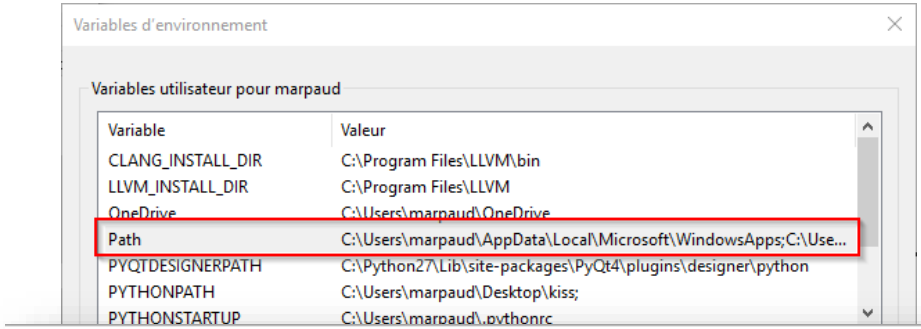
	Version	Host	GCC / Mingw-w64 Version	Languages	Additional Software in Package Manager
Arch Linux	Arch Linux		8.2.0/5.0.4	Ada, C, C++, Fortran, Obj-C, Obj-C++	305, full list: Show
Cygwin	Rolling	Windows	5.4.0/5.0.2	Ada, C, C++, Fortran, Obj-C	5 (bzip2, libgcrypt, libgpg-error, minizip, xz, zlib)
Debian	Debian 7 (Wheezy)		4.6.3/2.0.3	Ada, C, C++, Fortran, Obj-C, Obj-C++, OCaml	2 (gdb, nsis)
	Debian 8 (Jessie)		4.9.1/3.2.0		
	Debian 9 (Stretch)		6.3.0/5.0.0		9 (gdb, libassuan, libgcrypt, libgpg-error, libksba, libnpth, nsis, win-iconv, zlib)
	Debian 10 (Buster)		8.3.0/6.0.0		
Fedora	Fedora 19		4.8.1/?	Ada, C, C++, Fortran, Obj-C, Obj-C++	149, full list: Show
MacPorts	Rolling	macOS **macOS**	8.2.0/5.0.4	C, C++, Fortran, Obj-C, Obj-C++	1 (nsis)
MingW-W64-builds	Rolling	Windows	7.2.0/5.0.3	C, C++, Fortran	4 (gdb, libiconf, python, zlib)
Msys2	Rolling	Windows	8.2.0/trunk	Ada, C, C++, Fortran, Obj-C, Obj-C++, OCaml	many
Ubuntu	12.04 Precise Pangolin		4.6.3/2.0.1	Ada, C, C++, Fortran, Obj-C, Obj-C++, OCaml	2 (nsis, gdb)
	14.04 Trusty Tahr		4.8.2/3.1.0		
	14.10 Utopic Unicorn		4.9.1/3.1.0		
	15.04 Vivid Vervet		4.9.2/3.2.0		
	15.10 Wily Werewolf		4.9.2/4.0.2		
	16.04 Xenial Xerus		5.3.1/4.0.4		3 (nsis, gdb, zlib)
Win-Builds	1.5	Windows Linux	4.8.3/3.3.0	C, C++	91, full list: Show

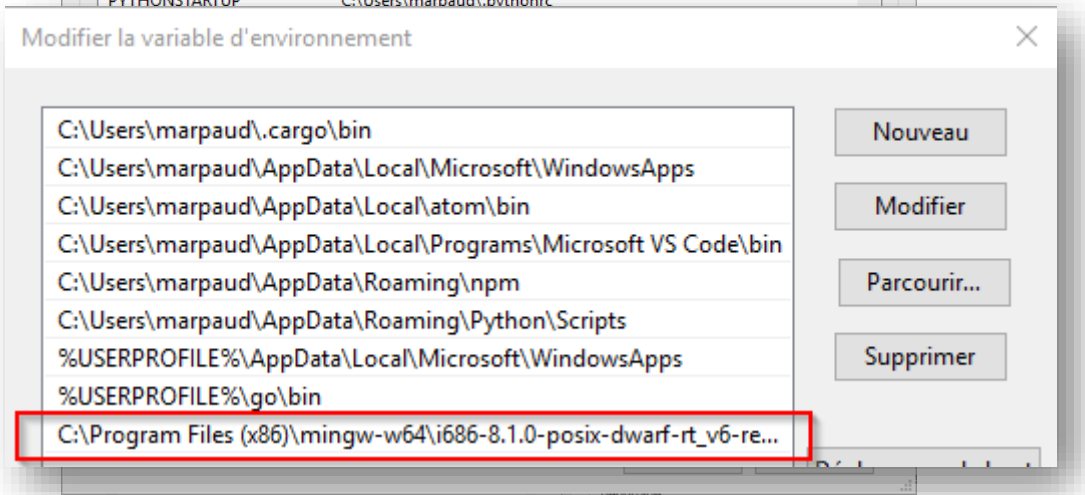
Annexe
Installer g++ et gcc sur Windows (2/2)

Modifier la variable d'environnement « **Path** » pour pointer vers les dossiers « **bin** » de MinGW: *C:\Program Files (x86)\mingw-w64\i686-8.1.0-posix-dwarf-rt_v6-rev0\mingw32\bin* (par exemple pour moi)



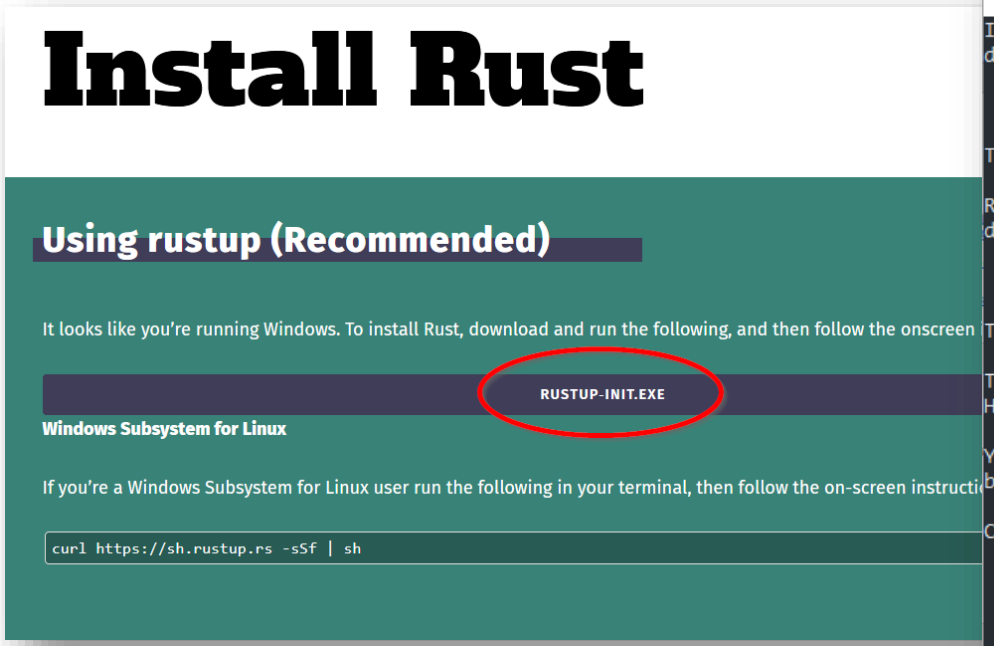


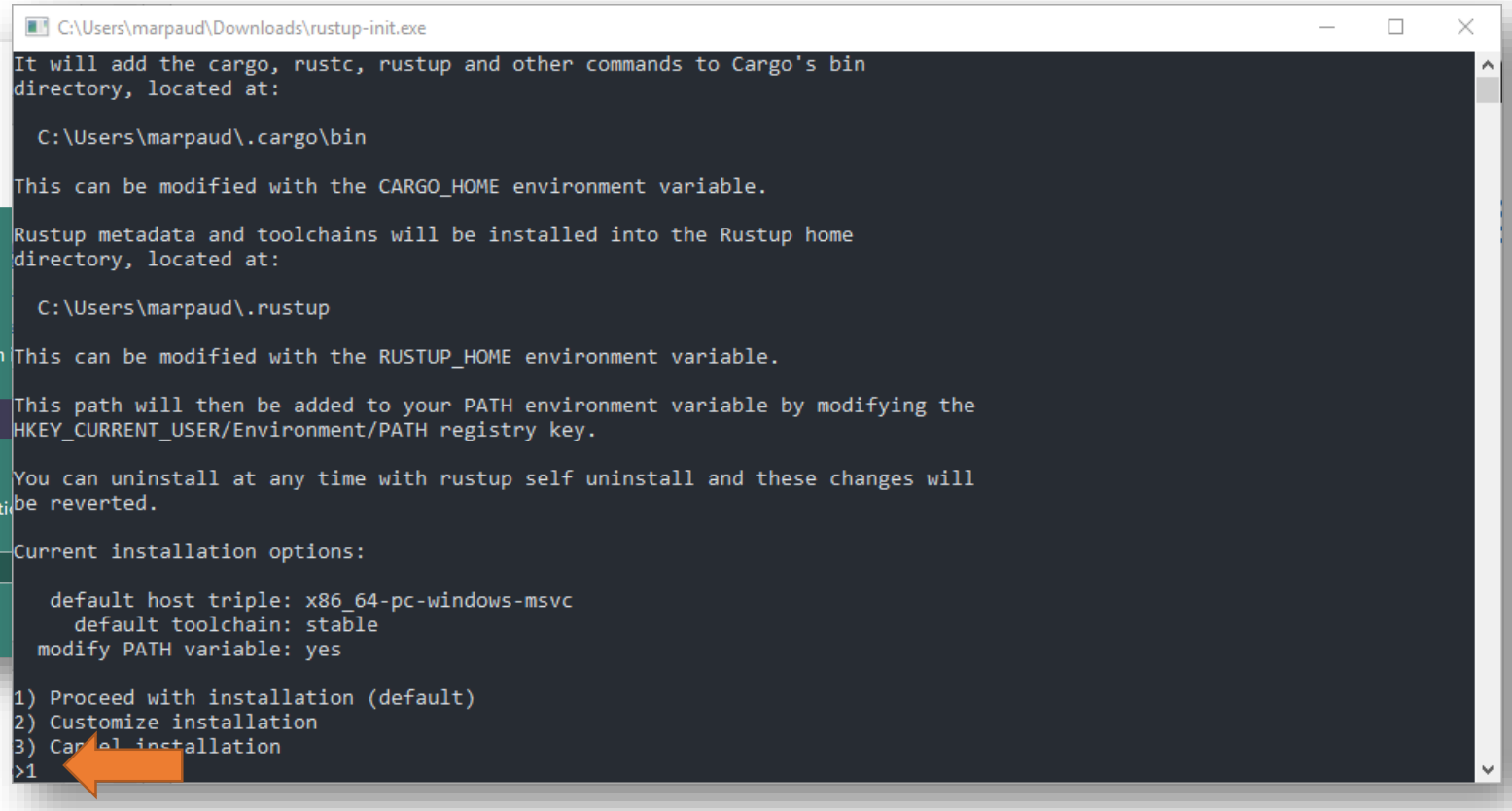




Annexe
Installation de Rust sur Windows (1/1)

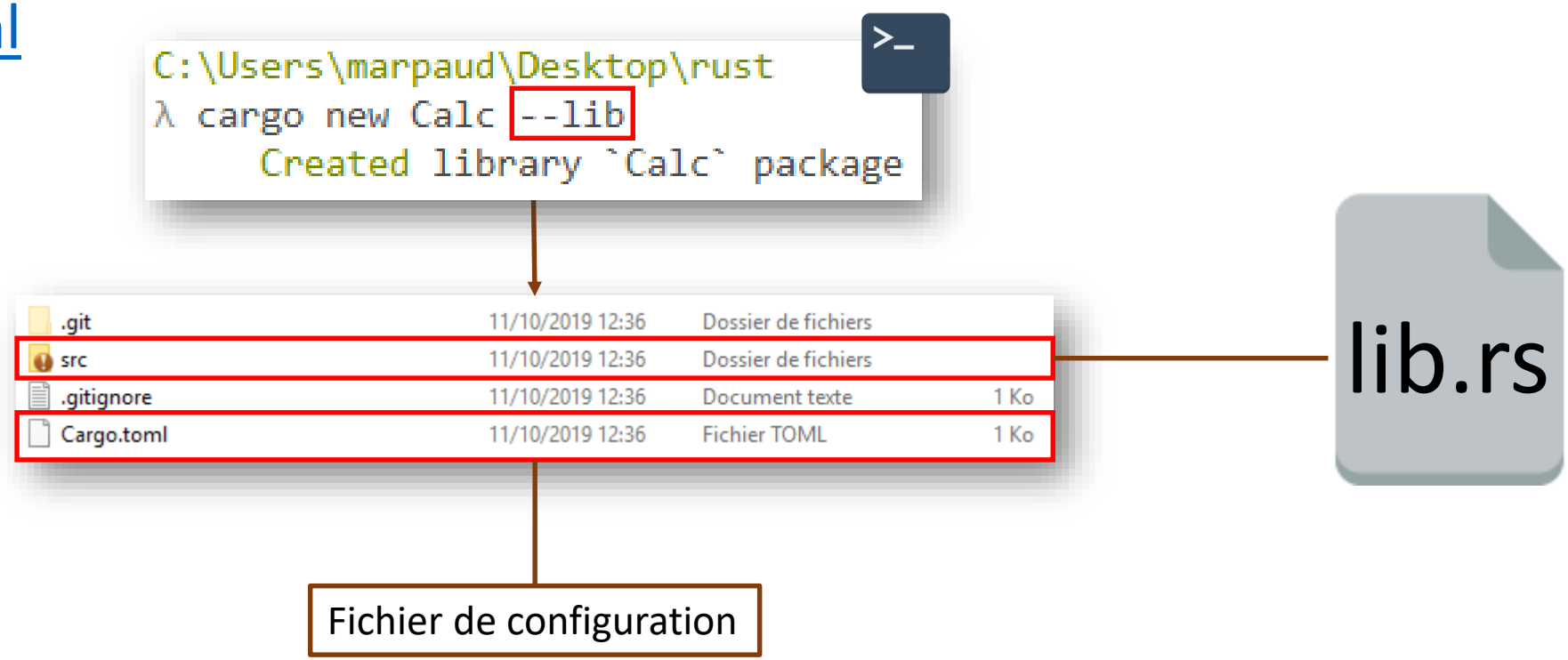
Aller sur https://www.rust-lang.org/tools/install





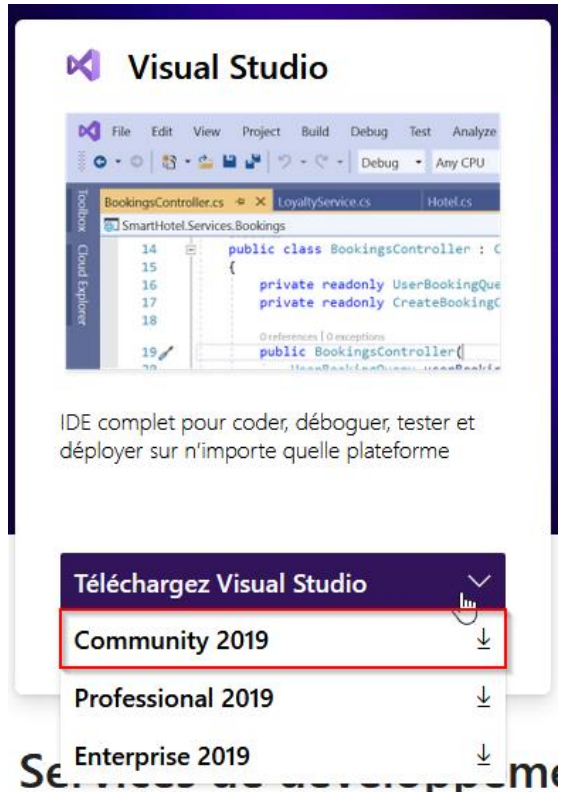
Annexe
Rust, Cargo et librairie (1/1)

Créer un projet **Cargo** → https://doc.rust-lang.org/book/ch01-03-hello-cargo.html et https://doc.rust-lang.org/cargo/commands/cargo-new.html



Annexe
C# sur Windows (1/2)

- Pour pouvoir compiler du C#, il faut installer **Visual Studio** : https://visualstudio.microsoft.com/fr/ , ici on a utilisé la version « community »
- Une fois installé, ouvrir **Visual Studio Installer**, et installer « Développement .NET Desktop »



Annexe
C# sur Windows (2/2)

Si vous voulez utiliser les lignes de commande pour compiler du C#, il faut chercher les « prompt » dans le menu démarrer, et choisir la version qui vous convient (32bits ou 64 bits)