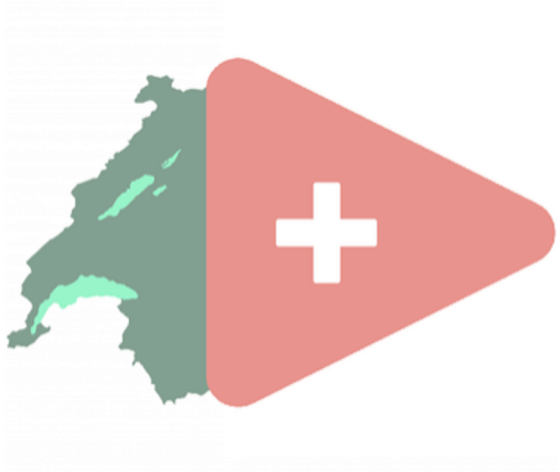


LabVIEW - Python - OpenCV

UG LabVIEW, July 9 2024

Dr. Christophe Salzmann

EPFL

Laboratoire
d' Automatique

LabVIEW

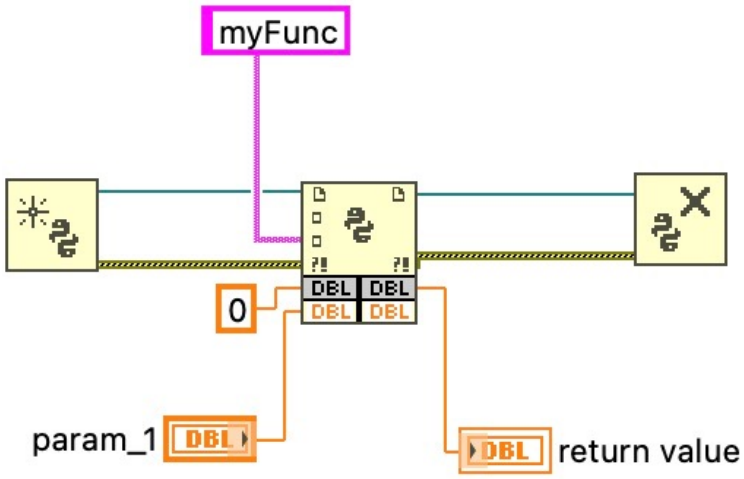
…

LabVIEW can access external code via

- System Exec
- DLL/CLN
- Shared Memory
- Network primitive (TCP/UDP/RPC/REST/WS)
- Formula/Matlab node
- External node, ex: **Python**, .Net, ActiveX, AppleScript

LabVIEW

Python node example



Non-working example

Python

- Interpreted textual language with compiled modules and components
- Cross platform and open source

- Zillions of libraries/modules written in python or compiled

- Python can call external modules
- Python can be embedded in other programs

Python example

Python function example

```
def myFunc(Param_1):

    if Param_1 > 0:
        return Param_1*Param_1

    elif Param_1 == 0:
        return 0

    else:
        return Param_1*Param_1*Param_1
```

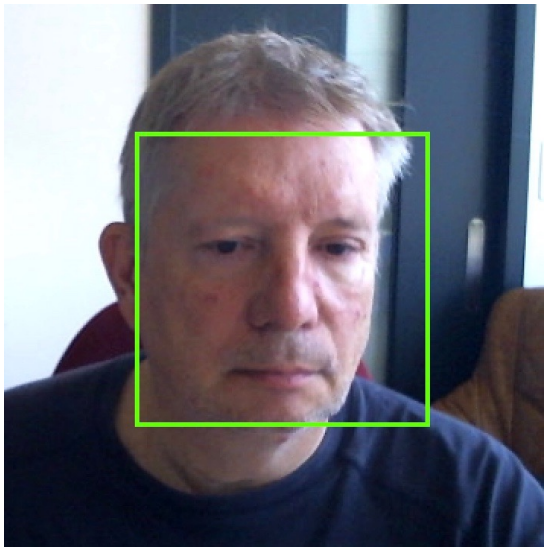
OpenCV

- Open source computer vision and machine learning software library
- Initially developed by Intel, written in c++
- Lots of machine vision algorithms, etc.
- Permits real-time processing, take advantage of existing hardware
- Can be interfaced via C/C++, Python, Java and MATLAB

OpenCV examples

Blur

Sobel

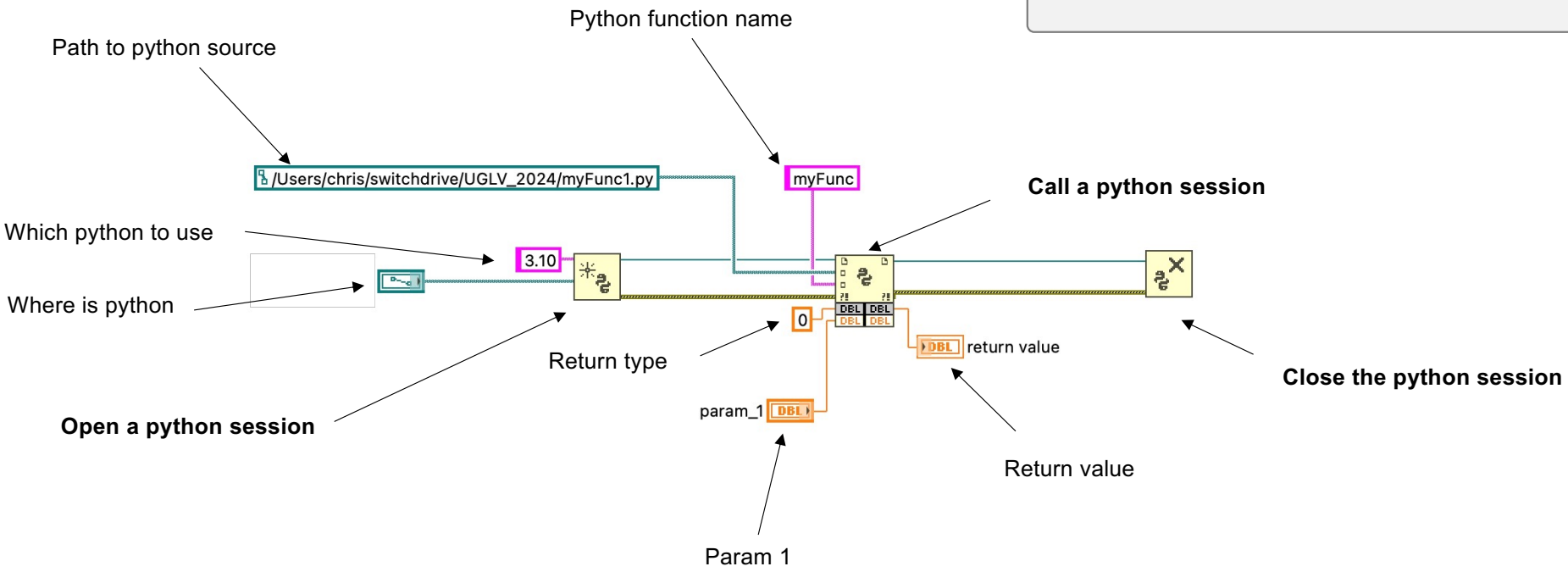


Face detection

LabVIEW + Python

myFunc1.py

```python
def myFunc(Param_1):

    if Param_1 > 0:
        return Param_1*Param_1

    elif Param_1 == 0:
        return 0

    else:
        return Param_1*Param_1*Param_1
```



LabVIEW + Python - parameters

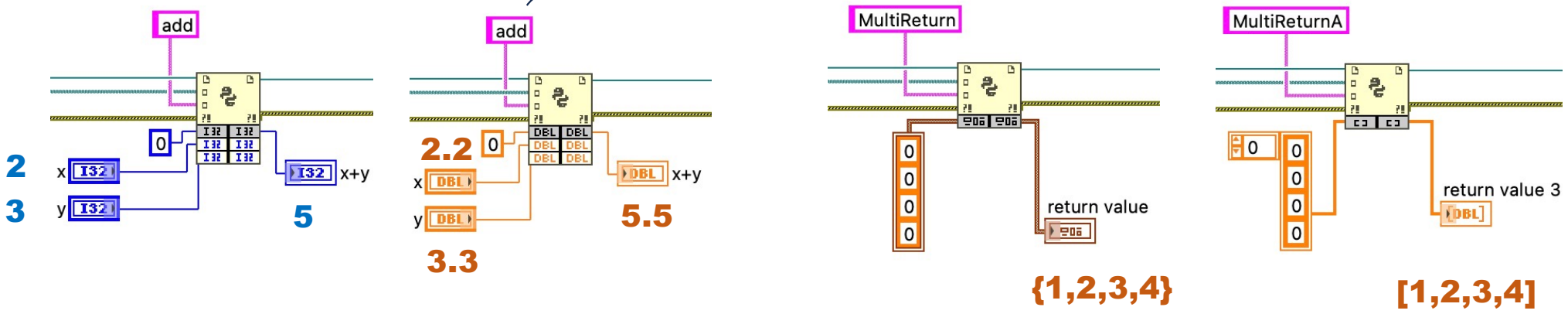
LabVIEW and python parameters **must match** !

Python has only 1 return parameter, but more than 1 value can be returned, (by position or by name)

Python is dynamically typed (no need to declare a var and type can change)

myFunc1.py

```
def add(x,y):
    return x+y


def MultiReturn():
    return (1,2,3,4)

def MultiReturnA():
    return [1,2,3,4]
```

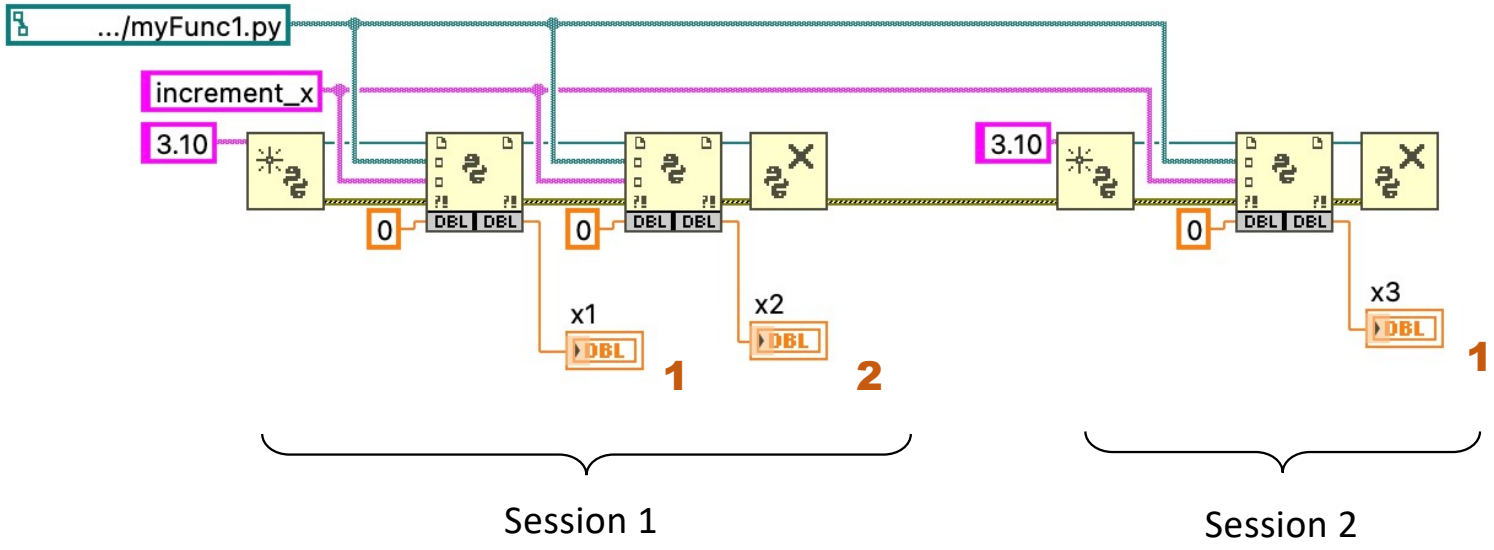


LabVIEW + Python - session

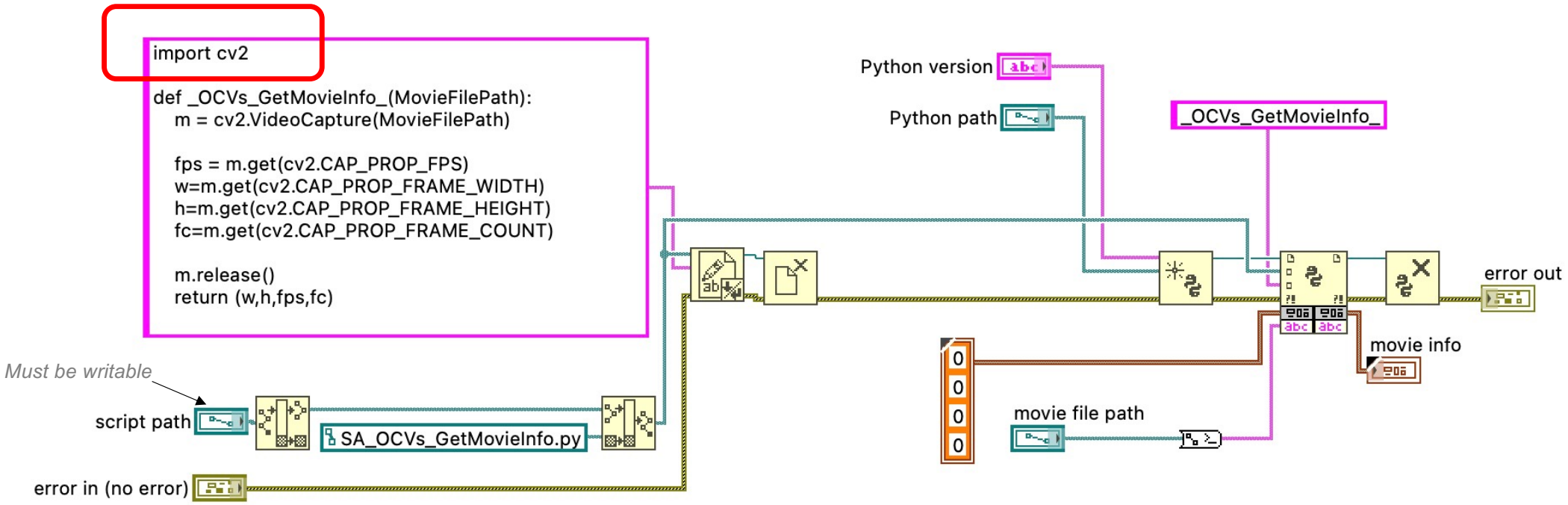
Python data can be *stored* between calls in global var

myFunc1.py

```
global x
x = 0

def increment_x():
    global x
    x += 1
    return x
```



LabVIEW + Python + openCV

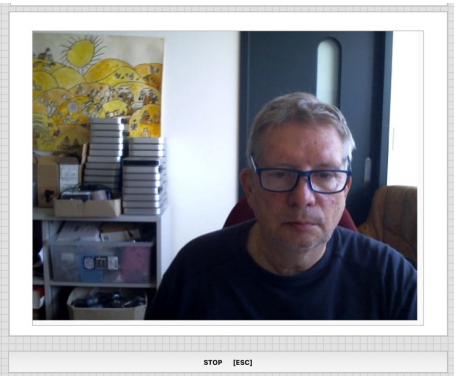


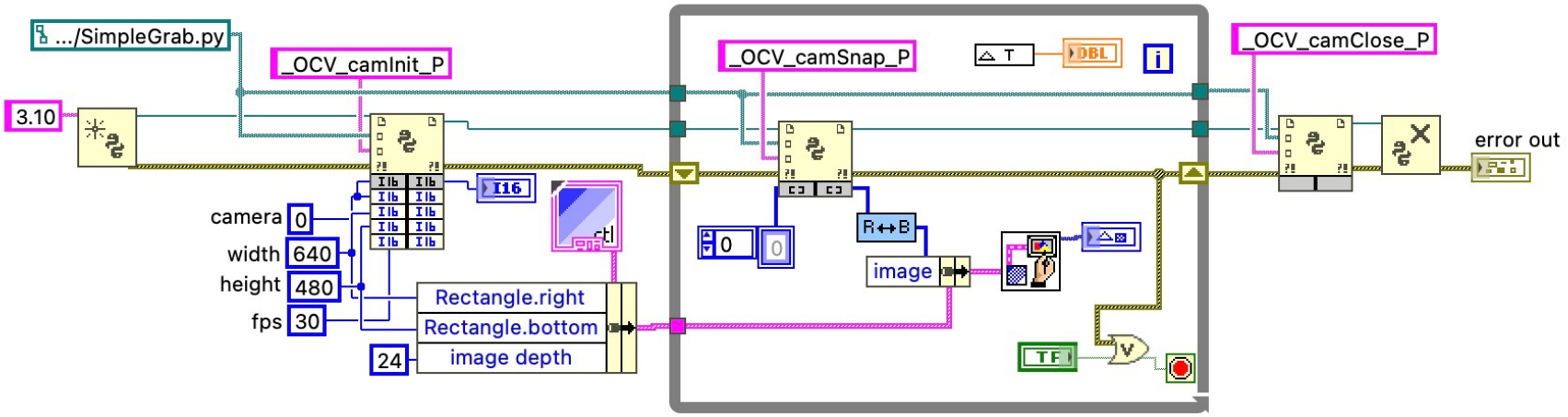
Keep the script handy

Call the function

LabVIEW + Python + openCV – grab image

```python
import cv2
import numpy as np

global gVCap

def _OCV_camInit_P(camera = 0,width = 640, height = 480, fps=50):
    global gVCap
    gVCap = cv2.VideoCapture(camera)
    if (gVCap.isOpened() == False):
                return -1
    gVCap.set(cv2.CAP_PROP_FRAME_WIDTH,width)
    gVCap.set(cv2.CAP_PROP_FRAME_HEIGHT,height)
    gVCap.set(cv2.CAP_PROP_FPS, fps)
    return 0

def _OCV_camSnap_P():
    global gVCap
    ret,frame = gVCap.read()
    return frame.flatten()

def _OCV_camClose_P():
    global gVCap
    gVCap.release()
    return 0
```





Need to check Python environment

Get official Python from https://www.python.org/downloads

You can have more than one version of python installed…

Use a module/environment manger ex **pip3** (other exist)

- *OpenCV* **pip3 install opencv-python**
- *numpy* (array + math) **pip3 install numpy**
- *logging* **pip3 install logging**

LabVIEW handles python environment to encapsulate your code, modules and specific version of python

Python debug in LabVIEW

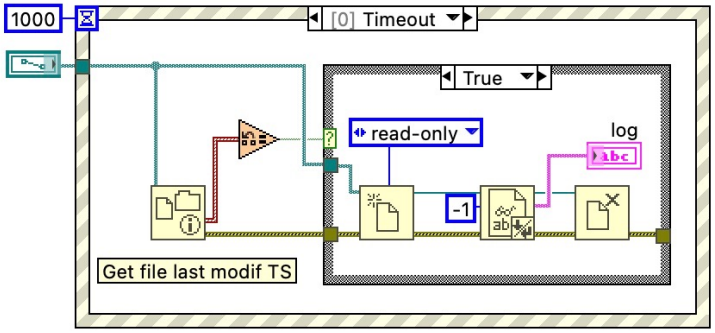
- Debugging Python code from when embedded in LabVIEW can be a pain…
- A simple solution is to use **python logging** which can write message from python in a file, then read this file from LabVIEW at regular pace
- Add these line at the top of your python file

```
import logging
logging.basicConfig(filename='/path/to/the/logfile.log', filemode='a',
format='%(asctime)s %(process)d [%(filename)s, %(funcName)s(), l:%(lineno)d] -  %(message)s', level=logging.INFO)
```

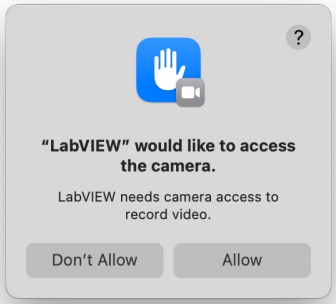
- Then log info/error when needed

```
logging.info('requested cam: %d, w:%d,h:%d, fps:%d',camera,width,height,fps)
logging.error("Unable to set camera size, try resizing the frame once captured")
```

- Display in LabVIEW



Demos time



Notes

- On OSX you need to allow LabVIEW to access cameras
- Current version has a bug and requires to modify LabVIEW, should be fixed in 2023Q3 patch2
- If not, see instructions provided in the readme file

Usefull commands

To get **python3 version** so you don’t have to edit your diagram

```
/usr/local/bin/python3 –V
```

To get installed **modules**

```
/usr/local/bin/pip3 freeze
```

To get a list of connected **cameras**

```
system_profiler SPCameraDataType
```

Conclusions

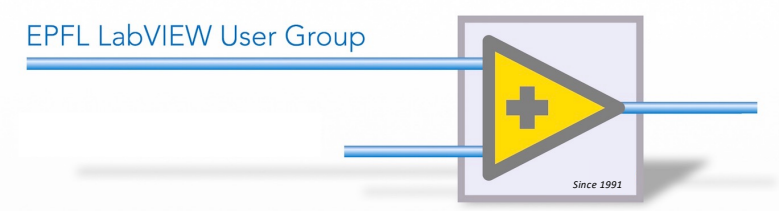
- Python node is a must, it can compensate/replace many LabVIEW missing parts, ex: no IMAQ -> OpenCV
- Python node works well
- LV<->Python exchange speed increased drastically since LabVIEW 2023 (?)
- Python can be cumbersome for LabVIEW (and C) programmers
- Python has zillions of modules
- Python limitations remains

Q: will python replace LabVIEW ?

Links

- Code presented here will be available here:

https://labview.epfl.ch



https://forums.ni.com/t5/Romandie-LabVIEW-User-Group/gh-p/romandie-labview-user-group