PRODSP TECHNOLOGIES

LabVIEW and Python

BudLUG Santa - 2023, László Balogh
laszlo.balogh@prodsp.hu

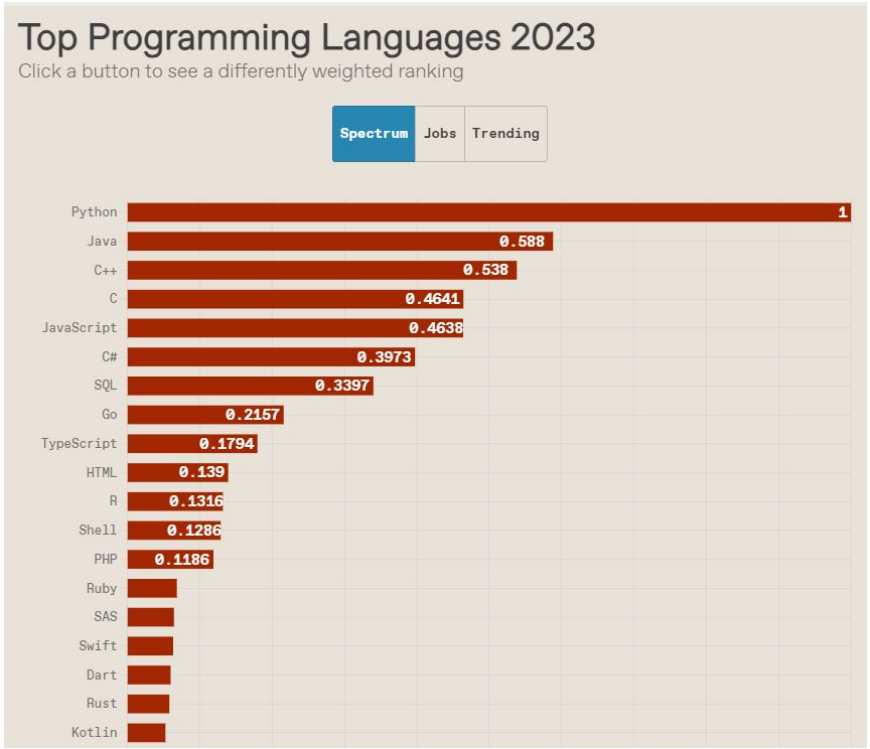
Content

PRODSP TECHNOLOGIES

- Why?
- How?
  - Embedding Labview
  - Processing TDMS File
  - Labview Node
  - Python via REST API
  - Scripting

Why Python?

The Top Programming Languages 2023 › Python and SQL are on top, but old languages shouldn't be forgotten



Source:

IEEE Spectrum

Python - Features

PRODSP TECHNOLOGIES

Pros

- Easy to learn
- Easy to read
- Text based
- All the platforms are supported (Win, Linux, Mac, etc.)
- Rich set of functionality

Cons

- Indentation
- Not type safe
- Different style of packages
- Deployment

```
1  def FuncHelloWorld():
2      print("Hello world!")
3
4
5  if __name__ == "__main__":
6      FuncHelloWorld()
7
```

Python and AI

PRODSP TECHNOLOGIES

Keras

Scikit-learn

TensorFlow

Pandas

PyTorch

NumPy

Theano

SciPy

Matplotlib

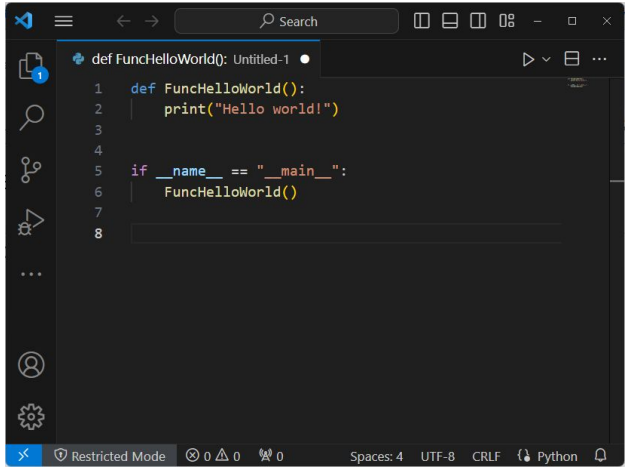
AI is the new electricity

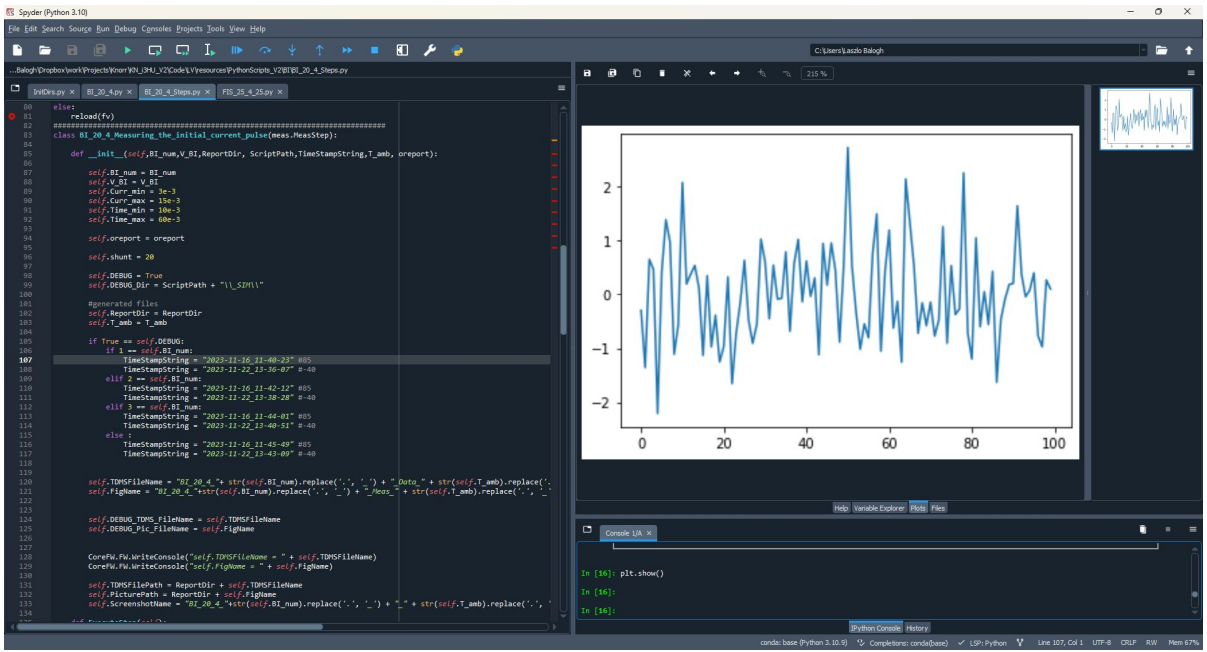
- 100 years ago, electricity transformed every industry.
- AI is bringing an equally big transformation.

Andrew Ng

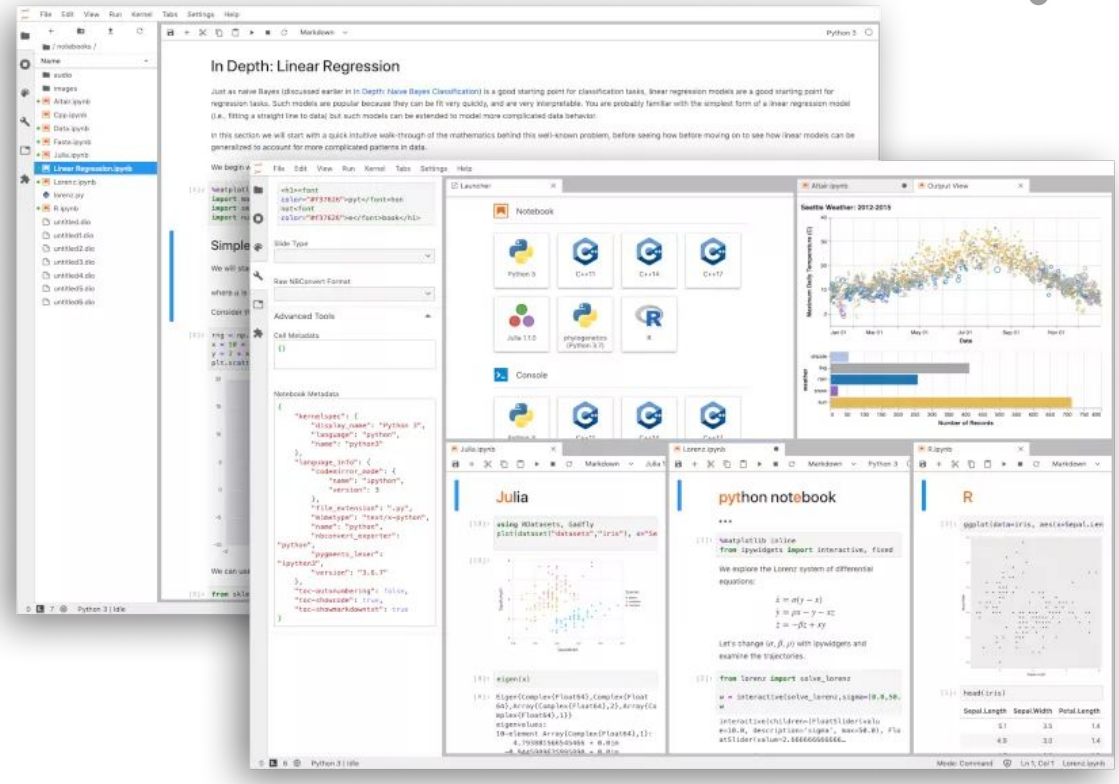
Stanford

Editors


Search
def FuncHelloWorld(): Untitled-1
1 def FuncHelloWorld():
2 print("Hello world!")
3
4
5 if __name__ == "__main__":
6 FuncHelloWorld()
7
8
Restricted Mode
0 0 0
Spaces: 4
UTF-8
CRLF
Python


Spyder (Python 3.10)
File Edit Search Source Run Debug Consoles Projects Tools View Help
BI_20_4_Steps.py
class BI_20_4_Measuring_the_initial_current_pulse(meas.MeasStep):
def __init__(self,BI_num,V_BI,ReportDir, ScriptPath,TimeStampString,T_amb, oreport):
self.BI_num = BI_num
self.V_BI = V_BI
self.Curr_min = 3e-3
self.Curr_max = 15e-3
self.Time_min = 10e-3
self.Time_max = 60e-3
self.oreport = oreport
self.shunt = 20
self.DEBUG = True
self.DEBUG_Dir = ScriptPath + "_SIM\\"
#generated files
self.ReportDir = ReportDir
self.T_amb = T_amb
if True == self.DEBUG:
if 1 == self.BI_num:
TimeStampString = "2023-11-16_11-40-23" #85
TimeStampString = "2023-11-22_13-36-07" #-40
elif 2 == self.BI_num:
TimeStampString = "2023-11-16_11-42-12" #85
TimeStampString = "2023-11-22_13-38-28" #-40
elif 3 == self.BI_num:
TimeStampString = "2023-11-16_11-44-01" #85
TimeStampString = "2023-11-22_13-40-51" #-40
else :
TimeStampString = "2023-11-16_11-45-49" #85
TimeStampString = "2023-11-22_13-43-09" #-40
self.TDMSFileName = "BI_20_4_"+ str(self.BI_num).replace('.', '_') + "_Data_" + str(self.T_amb).replace('.',
self.FigName = "BI_20_4_"+str(self.BI_num).replace('.', '_') + "_Meas_" + str(self.T_amb).replace('.', '_'
self.DEBUG_TDMS_FileName = self.TDMSFileName
self.DEBUG_Pic_FileName = self.FigName
CoreFW.FW.WriteConsole("self.TDMSFileName = " + self.TDMSFileName)
CoreFW.FW.WriteConsole("self.FigName = " + self.FigName)
self.TDMSFilePath = ReportDir + self.TDMSFileName
self.PicturePath = ReportDir + self.FigName
self.ScreenshotName = "BI_20_4_"+str(self.BI_num).replace('.', '_') + "_" + str(self.T_amb).replace('.',
2
1
0
−1
−2
0
20
40
60
80
100
Console 1/A
In [16]: plt.show()
In [16]:
In [16]:
IPython Console
History

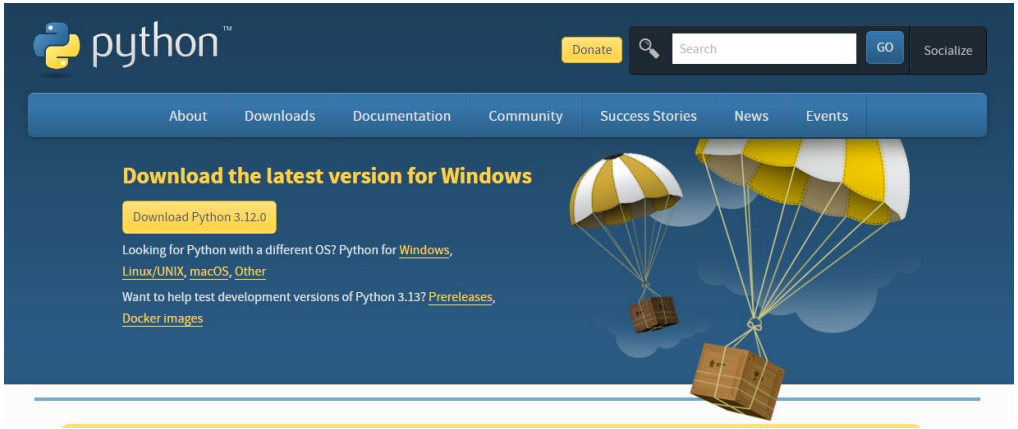
Python Notebooks



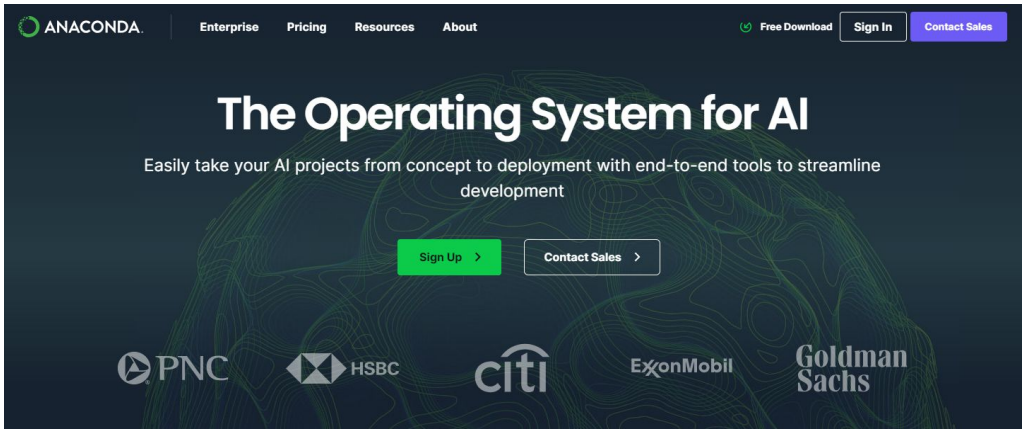
- Jupiter
- JupiterLab

Python Distributions

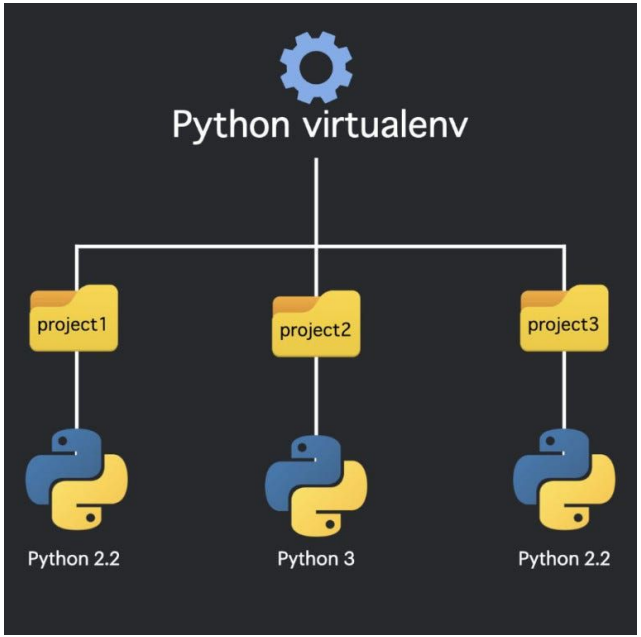
www.python.org

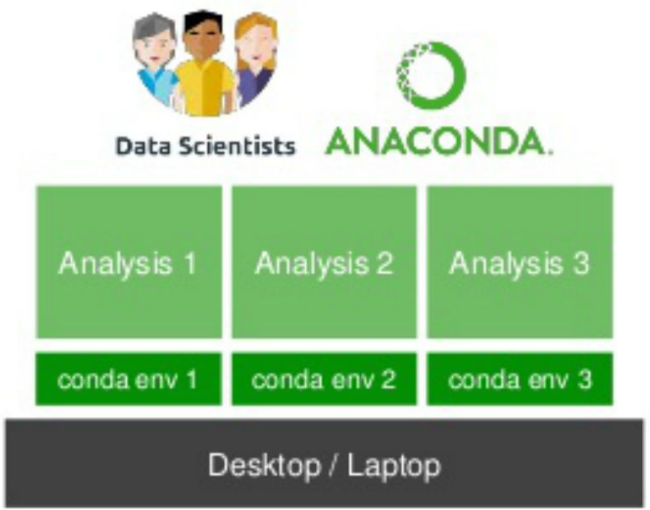


www.anaconda.com



penv and conda





Accessing NI Hardware from Python

PRODSP TECHNOLOGIES

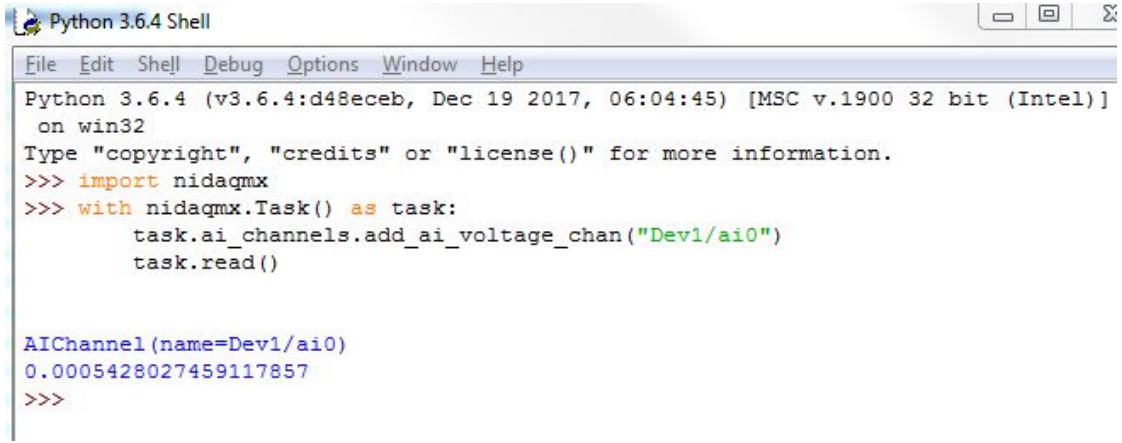
Using a NI DAQ Device with Python and NI DAQmx in Windows

Updated Aug 29, 2023

Environment

Driver
- NI-DAQmx

Programming Language
- Python

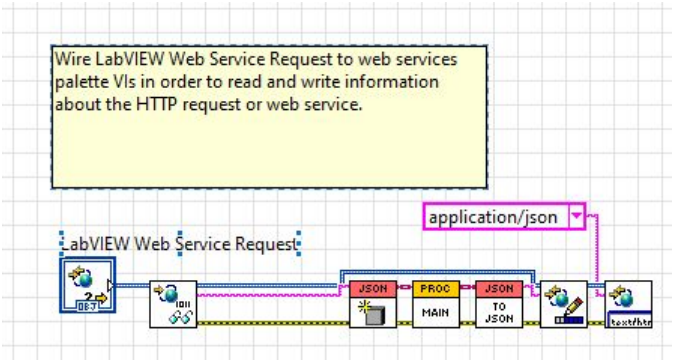


Embedding Labview

- Calling Labview generated dll:

```
from ctypes import *
from sys import exit

lvdll = WinDLL("labview-testdll.dll")
add = lvdll.Addition
add.argtypes = [c_int, c_int]
add.restype = c_int
print("Addition of 3 and 3:", add(3, 3))
exit()
```

- Web service in Labview

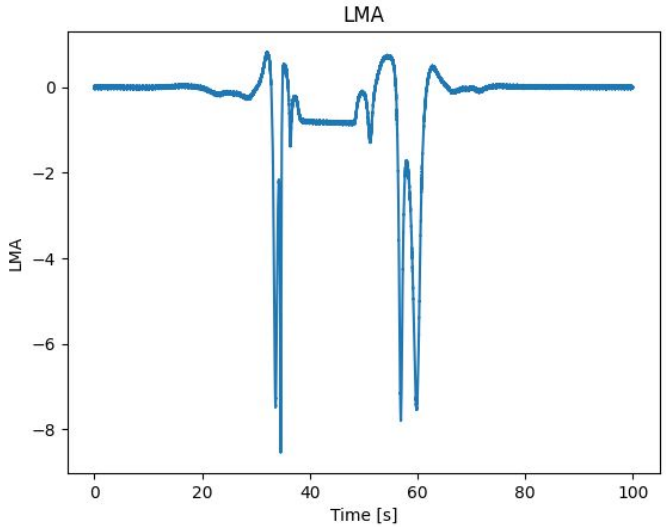


Processing TDMS File in Python

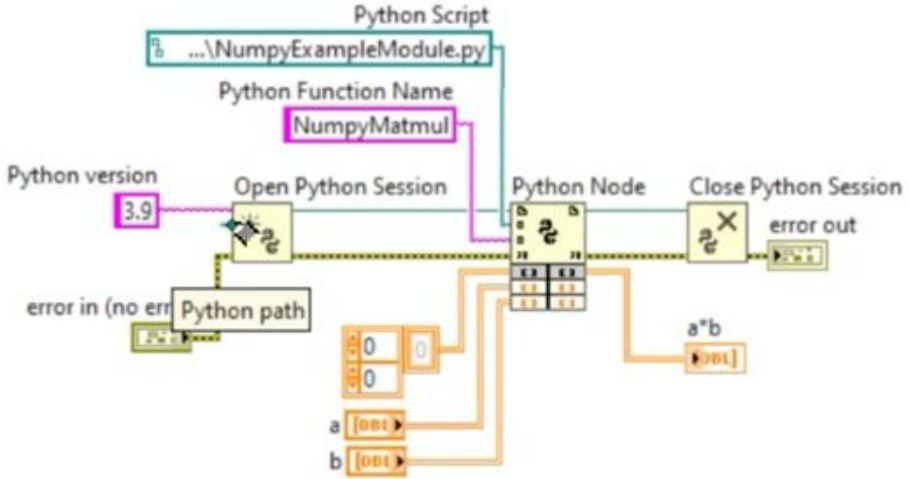
PRODSP TECHNOLOGIES

```
from nptdms import TdmsFile
import matplotlib.pyplot as plt
import pandas as pd

fn = "TEST LOW SPEED-20200410-114451-Part  0.tdms"

tdms_file = TdmsFile.read(fn)
group_names = []
for group in tdms_file.groups():
    group_name = group.name
    group_names.append(group_name)

#-------------------------------------------------------
# LMA v Time plot
group_LMA = tdms_file.groups()[group_names.index("LMA")]

LMA_channels = []
for ch in group_LMA.channels():
    channel_name = ch.name
    LMA_channels.append(channel_name)

LMA_ch = group_LMA.channels()[LMA_channels.index("LMA filtered")]


Data = LMA_ch[:]
Time = LMA_ch.time_track()

plt.plot(Time,Data)
plt.title("LMA")
plt.xlabel("Time [s]")
plt.ylabel("LMA")
plt.show()
```

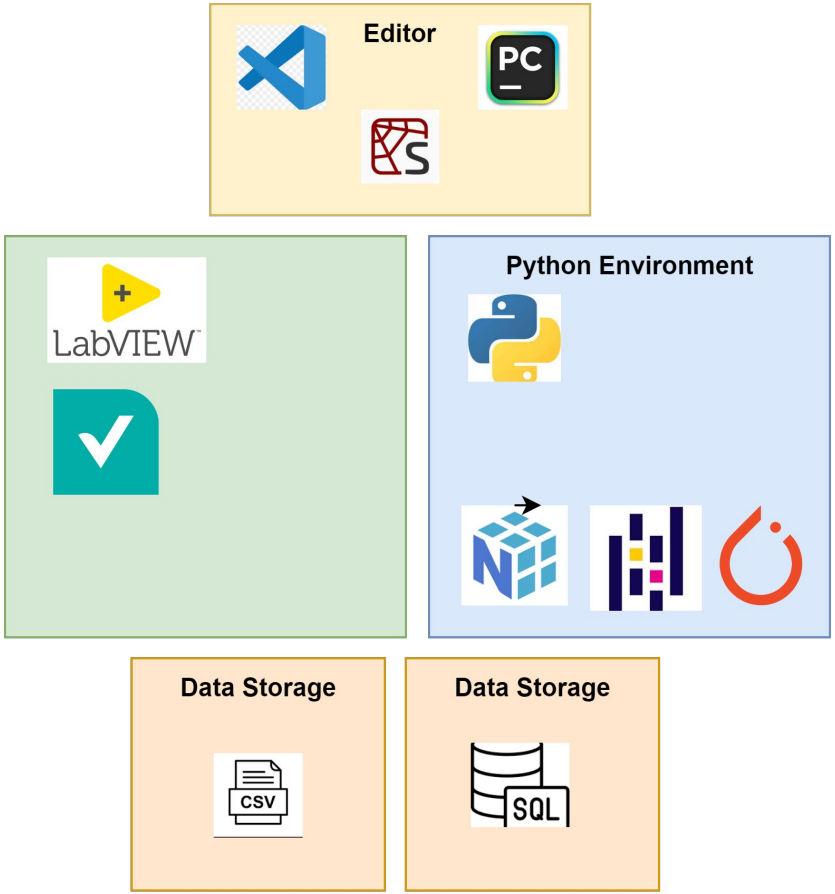
```
!pip install npTDMS
```



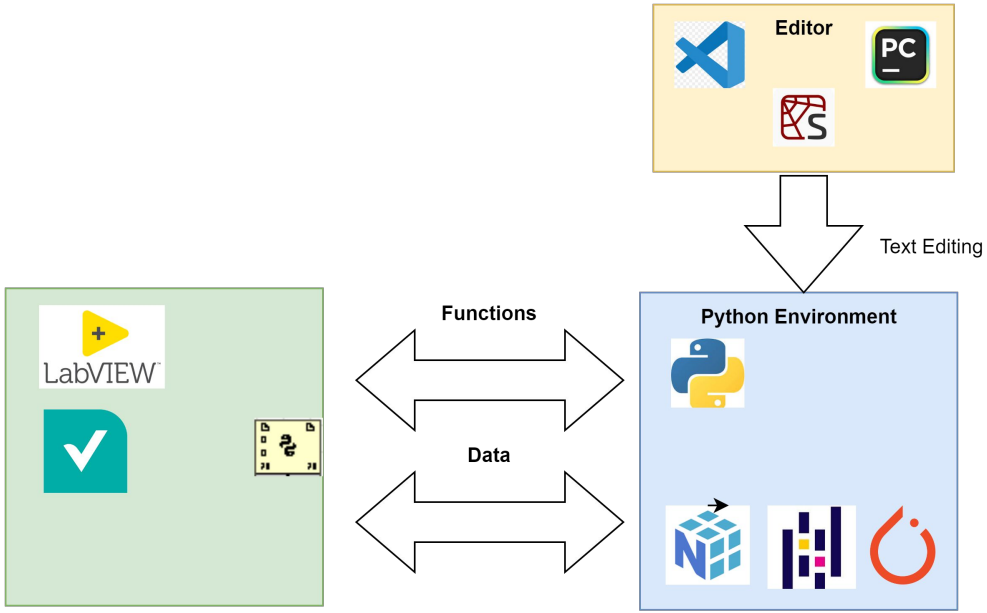
First Step: Python Node



Labview and Python - Components



Using Labview Python Node



Pros:

- Simple and quick

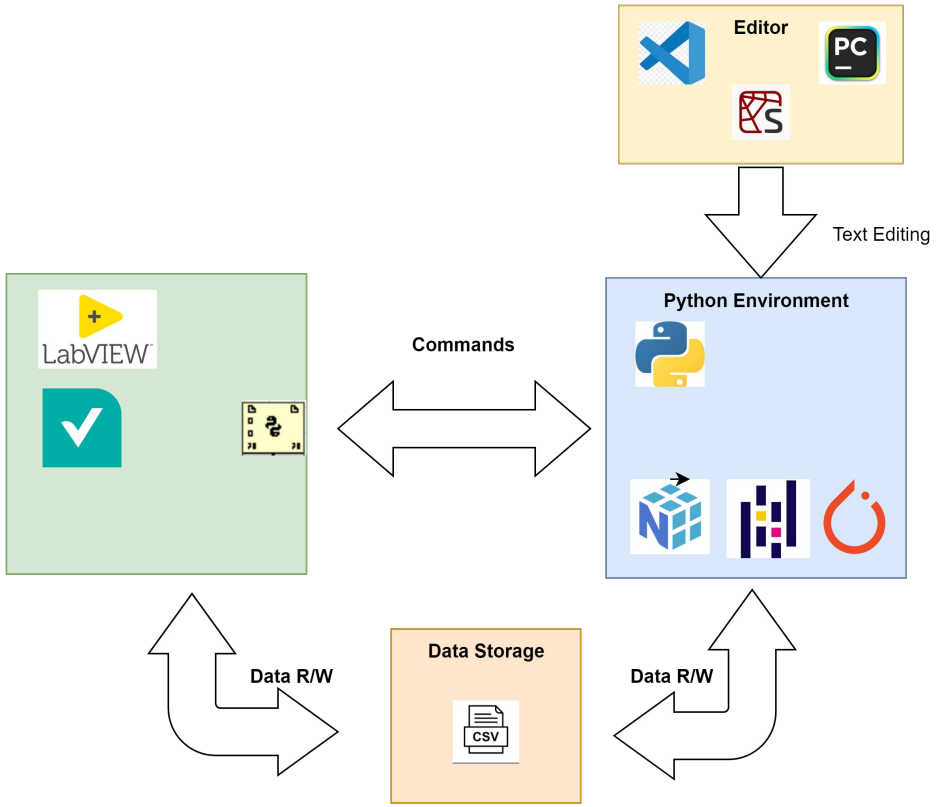
Cons:

- Limited type in data exchange
- No control in Python execution
- Big amount of data can be problematic
- Python environment is independent
- Deployment

Note:

- TestStand support Python (connector)

Using Python Node and File



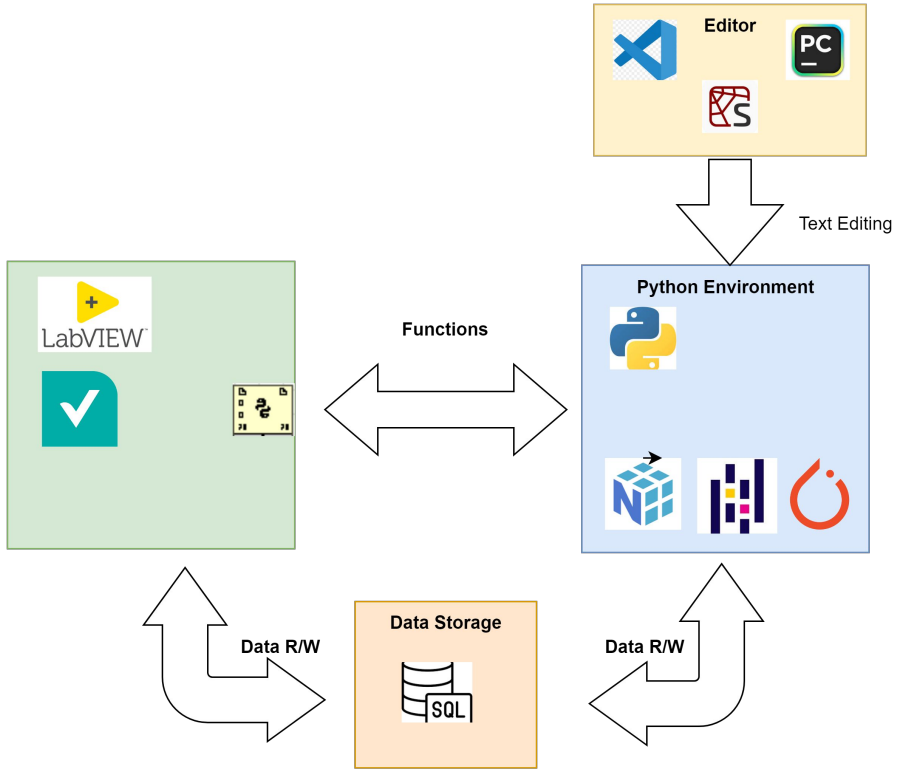
Pros:

- Simple
- Big data in files

Cons:

- No control in Python execution
- Maintenance: weak

Using Python Node and Database



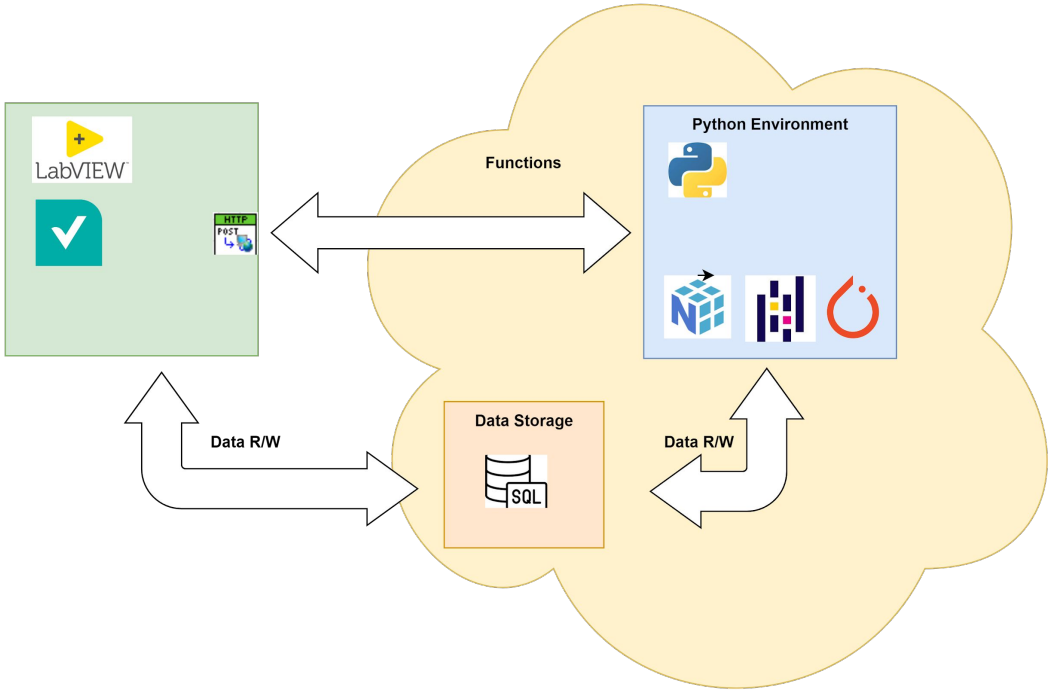
Pros:

- Big data in files
- Proper data management

Cons:

- No control in Python execution

Python in Cloud



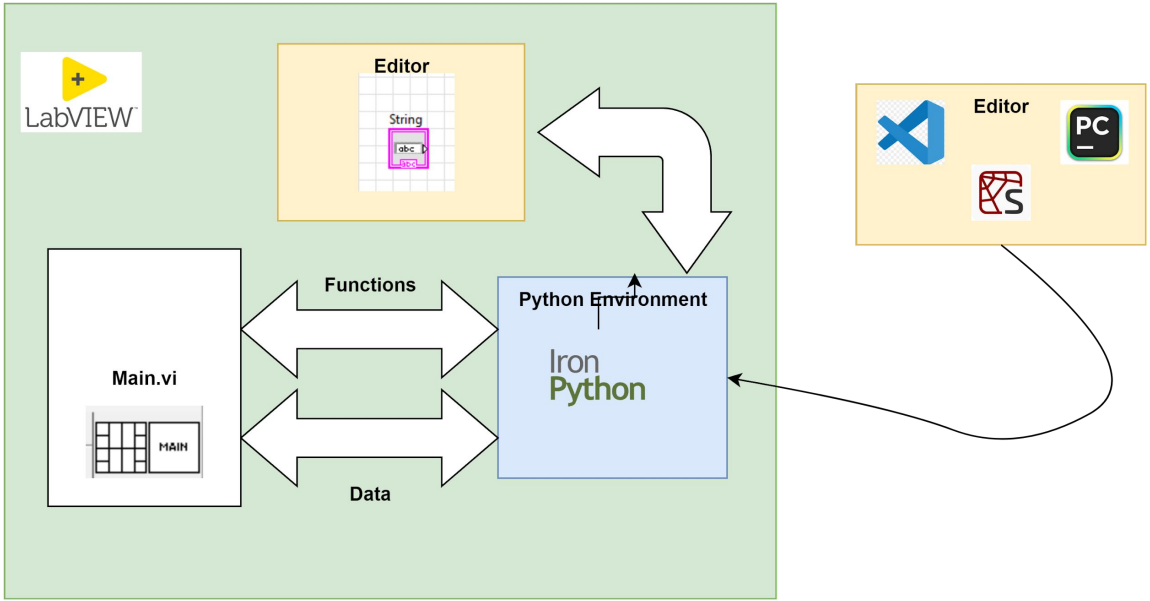
Pros:

- Big data in files
- Proper data management
- Independent data handling
- REST API

Cons:

- Deeper knowledge in cloud architecture

Embedding Python in Labview



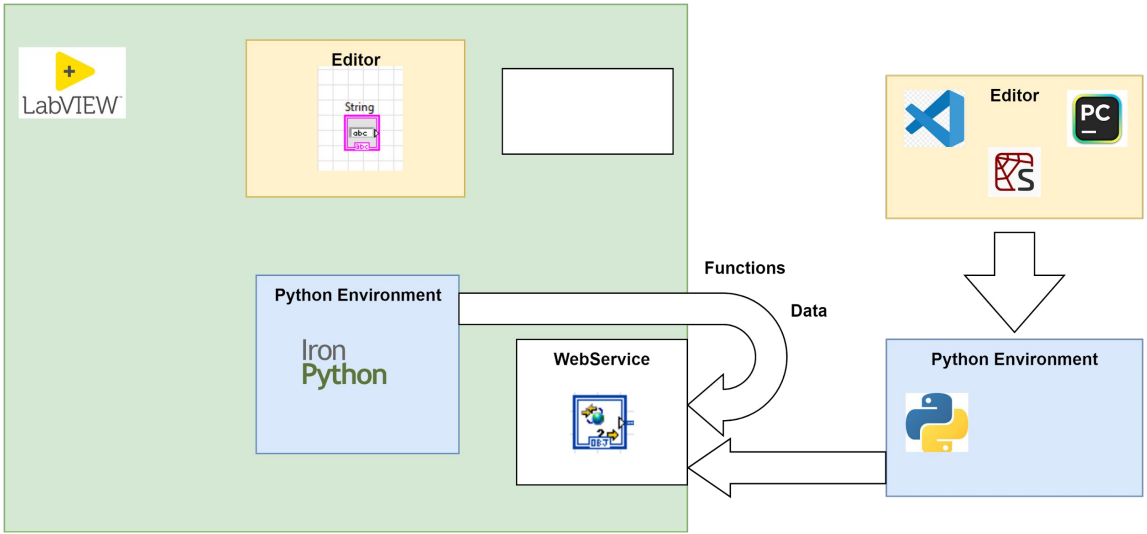
Pros:

- Full control
- Custom function
- Easy deployment

Cons:

- Limited library (just for scripting)
- Limited debug (yet)

Embedding Python in Labview



Pros:

- Full control
- Full debug
- Full featured environment

Cons:

- Web service is required

Scripting, TestStand vs Embedded Python

PRODSP TECHNOLOGIES

	Extensible	Modules	Speed	Format	Price
TestStand	Yes	Yes	OK	Binary	High
Python Scripting	Yes	No	OK	Text	Low

- Lightweight solution for scripting
- Competitive
- Language: Python

Summary

NetworkChuck
3.54M subscribers

One of the best ROI