Integrating LabVIEW and Python
Why LabVIEW?

- Seamless hardware integration
- Easy GUI development
- Rapid iteration & development
- Built-in parallelism
- Software engineering & productivity tools
- LabVIEW ecosystem
- Real-Time
- FPGA
- ...
Why Python?

- Thousands of mature, open source packages
Examples of using LabVIEW and Python together
Take advantage of thousands of Python packages

NumPy  matplotlib  pandas

TensorFlow  OpenCV  SciPy.org  plotly

SQLAlchemy  Flask  Requests
Take advantage of thousands of Python packages

- **NumPy**: the fundamental package for scientific computing with Python.
- **SciPy**: provides many user-friendly and efficient numerical routines.
- **scikit-learn**: machine learning in Python.
- **scikit-image**: image processing in Python.
- **matplotlib**: a Python 2D plotting library.
- **iPython / Jupyter**: interactive computing with Python.
- **pandas**: high-performance, easy-to-use data structures & data analysis tools
- **requests**: Python HTTP for humans.
- **SQLAlchemy**: database abstraction library.
- **Tensorflow**: machine learning platform
Integrate with anything that has a Python API
Leverage additional data visualizations from Python
Reuse existing Python scripts

Co-workers

Internet

```python
def create_barplot(data, target_image_filepath, labels, title):
    ind = np.arange(len(data))  # the x locations for the groups
    width = 0.35  # the width of the bars: can also be len(x) sequence
    fig, ax = plt.subplots(figsize=(5, 5))
    ax.set_title(title)
    ax.set_xlabel('')
    ax.set_ylabel('')
    ax.legend(loc='best')
    ax.set_ylabel('Number of Tests')
    ax.bar(ind, data, width, color='lightgrey')
    ax.bar(ind, data, width, bottom=data, color='lightgrey')
    plt.xticks(ind, labels)
    plt.title(title)
    plt.legend(['Test', 'Fail', 'Pass'])
    plt.savefig(target_image_filepath.replace('\','./'))
```

```
def create_boxplot(data, target_image_filepath, labels, title, colors):
    fig, ax = plt.subplots(figsize=(5, 5))
    ax.set_title(title)
    ax.set_xlabel('')
    ax.set_ylabel('')
    ax.legend(loc='best')
    ax.plot(data, color=colors)
    ax.boxplot(data, vert=False, showmeans=True)
    ax.set_xticks(range(1, data.shape[0] + 1))
    ax.set_yticks(range(1, data.shape[1] + 1))
    ax.set_yticklabels([''] + labels)
    ```
Methods of calling Python from LabVIEW
Methods

- Python Node
- Python Integration Toolkit for LabVIEW by Enthought
Methods of calling Python from LabVIEW

Python Node
Python Node

- Supported in LabVIEW 2018 and later
Python Node

- Open session & specify Python version
Python Node

- Call Python module (.py) and function

```python
import math

def Add(a, b):
    return a+b;

def ConcatenateStrings(str1, str2):
    return str1 + str2;

def EuclideanDistance(point1, point2):
    xDiff = point1[0] - point2[0];
    yDiff = point1[1] - point2[1];
    return math.sqrt(xDiff*xDiff + yDiff)
```
### Python Node – Supported Data Types

<table>
<thead>
<tr>
<th>LabVIEW Data Type</th>
<th>Python Data Type</th>
</tr>
</thead>
<tbody>
<tr>
<td>Integers</td>
<td>int</td>
</tr>
<tr>
<td>SGL, DBL</td>
<td>float</td>
</tr>
<tr>
<td>String</td>
<td>str</td>
</tr>
<tr>
<td>Boolean</td>
<td>bool</td>
</tr>
<tr>
<td>Array</td>
<td>list or NumPy array</td>
</tr>
<tr>
<td>Cluster</td>
<td>tuple</td>
</tr>
</tbody>
</table>

**Note** LabVIEW 2019 adds support for NumPy arrays.
Python Node

- Close the session
1. Install LabVIEW 2018 or later.

2. Install Python.
   ▪ Install supported version (2.7, 3.6)
   ▪ Install same bitness (32-bit, 64-bit) as LabVIEW

3. Locate your Python script and function
   ▪ Install any Python packages required by the Python script

4. Write/Run LabVIEW VI that calls Python script/function using Python Node
Python Node – Getting Started

1. Install LabVIEW 2018 or later.

2. Install Python.
   - Install supported version (2.7, 3.6)
   - Install same bitness (32-bit, 64-bit) as LabVIEW

3. Locate your Python script and function
   - Install any Python packages required by the Python script

4. Write/Run LabVIEW VI that calls Python script/function using Python Node

Python 3.6  https://www.python.org/downloads/release/python-368/
Python 2.7  https://www.python.org/downloads/release/python-2716/
1. Install LabVIEW 2018 or later.

2. Install Python.
   - Install supported version (2.7, 3.6)
   - Install same bitness (32-bit, 64-bit) as LabVIEW

3. Locate your Python script and function
   - Install any Python packages required by the Python script

4. Write/Run LabVIEW VI that calls Python script/function using Python Node

   If using Python 3.6, enable this checkbox in the installer
Python Node – Getting Started

1. Install LabVIEW 2018 or later.

2. Install Python.
   - Install supported version (2.7, 3.6)
   - Install same bitness (32-bit, 64-bit) as LabVIEW

3. Locate your Python script and function
   - Install any Python packages required by the Python script

4. Write/Run LabVIEW VI that calls Python script/function using Python Node

```
import sys
import cv2
import numpy as np
import os.path as op
```
Python Node – Getting Started

1. Install LabVIEW 2018 or later.

2. Install Python.
   - Install supported version (2.7, 3.6)
   - Install same bitness (32-bit, 64-bit) as LabVIEW

3. Locate your Python script and function
   - Install any Python packages required by the Python script

4. Write/Run LabVIEW VI that calls Python script/function using Python Node
Python Node Demos
Reuse Python Visualizations (matplotlib) Demo
Reuse Python API for Slack (Demo)
Python Node – Tips

- If Python function has a NumPy array parameter, then marshal your LabVIEW numeric array as a NumPy array (instead of a list)
Python Node – Tips

- Debugging
  - Add token to `<labview>\LabVIEW.ini` to show the Python console while Python Node is running

![Image of LabVIEW.ini configuration](image1.png)

![Image of Python console output](image2.png)
Python Node – Tips

- Debugging
  - Add token to `<labview>\LabVIEW.ini` to show the Python console while Python Node is running

Tip Place a breakpoint on the Close Python Session to keep the console window open
Python Node – Tips

- Read/write to global variable by using a wrapper function

```python
my_global_variable = "abc"

def read_my_global_variable():
    global my_global_variable
    return my_global_variable

def write_my_global_variable(value):
    global my_global_variable
    my_global_variable = value
```
Python Node – Tips

- Installing a set of packages
  - Use `pip freeze > requirements.txt` to generate file that lists installed packages.
  - Use `pip install -r requirements.txt` to install the list of installed packages.
Methods of calling Python from LabVIEW

Python Integration Toolkit
(By Enthought)
Python Integration Toolkit

- Supported in LabVIEW 2015 and later
- Price: $749
Python Integration Toolkit

- Specify which Python installation to use

1. Select Default Python

Welcome to the Python Integration Toolkit!

Use of the toolkit requires access to a Python installation, so please select one of the options below:

- Use Enthought Canopy

  Canopy is already installed at:
  C:\Users\Administrator\AppData\Local\Enthought\Canopy\admin\envs\User\python.exe.

- Locate an existing Python Installation

  C:\Users\Administrator\AppData\Local\Enthought\Canopy\admin\envs\User\python.exe

[Continue]
Python Integration Toolkit

- Open session & specify Python module (.py)
Python Integration Toolkit

- Call Python function

```
import math

def Add(a, b):
    return a + b;

def ConcatenateStrings(str1, str2):
    return str1 + str2;

def EuclideanDistance(point1, point2):
    xDiff = point1[0] - point2[0];
    yDiff = point1[1] - point2[1];
    return math.sqrt(xDiff*xDiff + yDiff)
```
Python Integration Toolkit

- Close the session
Python Integration Toolkit – Getting Started

1. Install LabVIEW 2015 or later.

2. Install Python Integration Toolkit
   - Automatically installs Python and Canopy IDE

3. Locate your Python script and function
   - Install Python packages required by Python functions via Canopy IDE

4. Run LabVIEW VI that calls Python script/function using Python Integration Toolkit API
Python Integration Toolkit – Getting Started

1. Install LabVIEW 2015 or later.

2. Install Python Integration Toolkit
   - Automatically installs Python and Canopy IDE

3. Locate your Python script and function
   - Install Python packages required by Python functions via Canopy IDE

4. Run LabVIEW VI that calls Python script/function using Python Integration Toolkit API
Python Integration Toolkit – Getting Started

1. Install LabVIEW 2015 or later.

2. Install Python Integration Toolkit
   - Automatically installs Python and Canopy IDE

3. Locate your Python script and function
   - Install Python packages required by Python functions via Canopy IDE

4. Run LabVIEW VI that calls Python script/function using Python Integration Toolkit API

```python
import sys
import cv2
import numpy as np
import os.path as op
```

![Python Integration Toolkit API](image)
Python Integration Toolkit – Getting Started

1. Install LabVIEW 2015 or later.

2. Install Python Integration Toolkit
   - Automatically installs Python and Canopy IDE

3. Locate your Python script and function
   - Install Python packages required by Python functions via Canopy IDE

4. Run LabVIEW VI that calls Python script/function using Python Integration Toolkit API
Python Integration Toolkit – Benefits

- Easy deployment of LabVIEW EXE and Python environment
- Includes Canopy Python IDE
- Native functions for the following:
  - Get/set Python global variables
  - Display Python console
  - Specify which Python installation to use
- Supports Python classes
- Variety of shipping examples
Easy deployment of LabVIEW EXE and Python environment

1. **[Toolkit]** Export Python Runtime
   - Tools»Python Integration Toolkit»Export Python Runtime
     - Choose Python packages to export

2. **[LabVIEW EXE Buildspec]** Create EXE

3. **[LabVIEW Installer Buildspec]** Create installer
   - Installs LabVIEW-built EXE
   - Installs Python Runtime to subfolder of installed application. Python Runtime is scoped just to the application.
   - Calls Python Runtime as post-install step

http://docs.enthought.com/python-for-LabVIEW/guide/deployment.html
Includes Canopy Python IDE

- Analysis and development environment for scientists and engineers using Python
- Provides curated set of packages managed through a package manager GUI
- Tailored to needs and workflows of scientists, analysts, and engineers
Additional Functions

Get and set Python global variables using native functions

Display Python console for debugging

Specify path to Python.exe
Variety of Shipping Examples

- Face detection using OpenCV package
- Frequency burst detection
- Call a cloud service
- Publish data to the web with plot.ly
- Python-powered web dashboard
- Get and set Python global variables
- Capture Printed Output
- Communication between concurrent Python script and LabVIEW VI
- ...
Enthought Python Integration Toolkit Demos

- Face detection using OpenCV package
- Frequency burst detection
- Call a cloud service
- Publish data to the web with plot.ly
- Python-powered web dashboard
- Get and set Python global variables
- Capture Printed Output
- …
Additional Use Cases
Concurrent communication between LabVIEW and long-running Python operation

How

- LabVIEW calls Python function that launches a long-running operation in a new thread… which can communicate data via global variable or Python queue.

- LabVIEW and the long-running Python function can communicate via Python queues and/or global variables

Demo
Calling LabVIEW from Python

- https://github.com/ni/python_labview_automation

```
from labview_automation import LabVIEW
lv = LabVIEW()

lv.start() # Launches the active LabVIEW with the listener VI
with lv.client() as c:
    control_values = {
        "DBL Control": 5.0,
        "String Control": "Hello World!",
        "Error In": {
            "status": False,
            "code": 0,
            "source": ""
        }
    }

indicators = c.run_vi_synchronous(
    vi_path, control_values)
print(indicators['Result'])
error_message = c.describe_error(indicators['Error Out'])

lv.kill() # Stop LabVIEW
```
Additional Resources
Download Python demo code

- https://github.com/allenh-ni/labview-python-demo

Python data visualizations

Slack API
Additional Content

- Calling Python Class Methods Using LabVIEW Python Node
  https://knowledge.ni.com/KnowledgeArticleDetails?id=kA00Z0000019UFmSAM&l=en-US

- Using Optional Arguments with Python Node in LabVIEW
  https://knowledge.ni.com/KnowledgeArticleDetails?id=kA00Z0000019WoASAU&l=en-US
NIWeek Hands-On: Integrate LabVIEW with Python

- **When:** Wed, May 22\(^\text{nd}\), 10:30 – 11:30am
- **Where:** Meeting Room 18C
- **Speaker:** Danielle Jobe, VI Technologies
Before you go, take the survey.
Stay Connected During and After NIWeek

ni.com/niweekcommunity
facebook.com/NationalInstruments
twitter.com/niglobal
youtube.com/nationalinstruments