TELEDYNE PRINCETON INSTRUMENTS
Everywhereyoulook™

Part of the Teledyne Imaging Group

Introduction to Automating LightField 6 with Python™

4411-0163
Issue 3
March 18, 2019

www.princetoninstruments.com

Revision History

Issue	Date	List of Changes
Issue 3	March 18, 2019	Issue 3 of this document incorporates the following changes: • Added new sample application: — Section 5.2.14, synchronous_acquisition.py, on page 23. • Rebranded as Teledyne Princeton Instruments.
Issue 2	January 2, 2018	Issue 2 of this document incorporates the following changes: • Updated the copyright date.
Issue 1	November 27, 2017	This is the initial release of this document.

Table of Contents

List of Figures

List of Tables

Chapter 1: Introduction to Automation

LightField 6 is able to interface with several third-party applications in order to automate the configuration and performance of experiments within LightField.

Automation is the ability to use a programming environment (e.g., Python™,) to control LightField externally to perform tasks such as:

- Modifying experiment configuration parameters such as:
  - Exposure Time;
  - Frames to Save;
- Connecting/disconnecting hardware;
- Displaying live data as it is being acquired;
- Importing previously acquired data that have been saved.

Automation differs from a standard LightField Add-in in that LightField is actually being controlled by an external program as opposed to running what is, effectively, a LightField subroutine.

This document provides an introduction to developing automation routines using Python™ programming environment.

NOTE:

This document is not intended to be a tutorial about using Python™. It is written with the assumption that the reader possesses a basic knowledge of the Python™ programming language.

This document provides information necessary to automate LightField 6 with Python™, including:

- Installing all required software;
- Configuring LightField 6 for automation by Python™;
- How to build an automation application;
- Experiment settings;
- Sample Python™ automation applications.

1.1 Prerequisites

In order to automate LightField 6 using Python™, the following requirements must be satisfied:

- LightField 6 must be installed on the host computer, including the **Add-Ins and Automation SDK** option.

  NOTE:

  For complete information, refer to the installation instructions included with LightField 6.

- 64-bit compatible Python™ version 3.6.3 (or later) must be installed on the host computer.

  The most recent 64-bit version of Python™ is available for download from the following website:

  https://www.python.org/downloads/windows/

  NOTE:

  If Python™ is not already installed, refer to Section A.1, Install Python™, on page 25 for complete information.

- 64-bit compatible **Python for .Net** version 3.6.0 (or later), must be installed on the host computer.

  Python for .Net is required to integrate Python™ applications with .NET 4.0+ Common Language Runtime (CLR) on Windows. This is required to run sample automation scripts created by Teledyne Princeton Instruments.

  Additional information about Python for .Net is available here:

  https://pythonnet.github.io/

  NOTE:

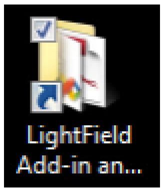
  If **Python for .Net** is not already installed on the host computer, refer to Section A.2, Install Python for .Net, on page 28 for complete information.

Chapter 2: Prepare LightField

This chapter describes the steps necessary to prepare LightField for automation using Python™.

2.1 Verify SDK Installation

Before developing any automation routines, verify the **Add-Ins and Automation SDK** has been included as part of the LightField installation.



When the SDK has been included, LightField places a shortcut on the host computer desktop. This shortcut points to a directory in which information required by the automation application developer is stored. This information includes:

- LightField Add-ins and Automation Programming Manual.pdf;

  Provides detailed programming information about the API and .NET function required to develop an automation application.
- Experiment XML Specification.pdf;

  This is the specification that defines how a LightField experiment file is structured.
- SPE 3.0 File Format Specification.pdf;

  This is the specification that defines how LightField saves acquired data.

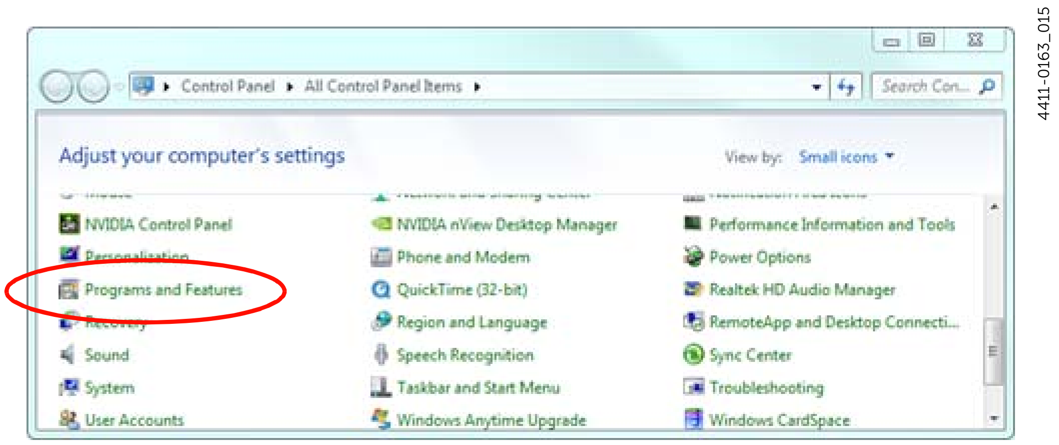
  In order to access data after it has been saved, how it is stored/save must first be understood which is detailed in this document.
- LightField Experiment Settings.chm.

  This Windows Help File provides information about all settings used when interacting with LightField. The information included in this help file is required when developing automation applications.

If the shortcut is not located on the host computer's desktop, it is a good indication that the SDK has not been included in the current LightField installation. Perform the following procedure to modify the LightField installation on the host computer:

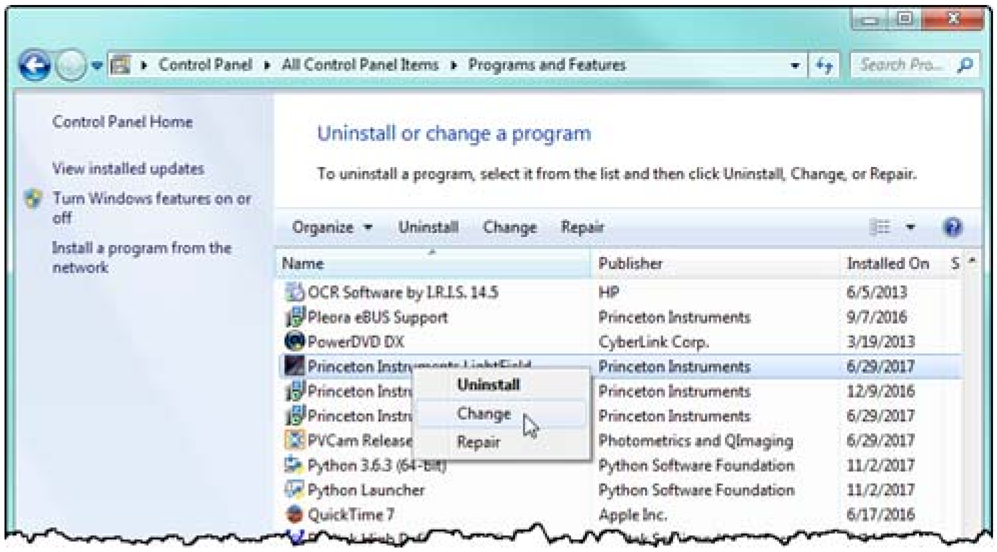
1. Open the **Windows Control Panel** and select **Programs and Features**. See Figure 2-1.

Figure 2-1: Typical Control Panel Window



2. From the list of installed programs, scroll to locate **Princeton Instruments LightField**, right-click on it, and select **Change** from the pull-down menu. See Figure 2-2.

Figure 2-2: Typical *Programs and Features* Dialog: Change


4411-0163_016
Control Panel ▸ All Control Panel Items ▸ Programs and Features
Search Pro...
Control Panel Home
View installed updates
Turn Windows features on or off
Install a program from the network
Organize
Uninstall
Change
Repair
Uninstall
Change
Repair

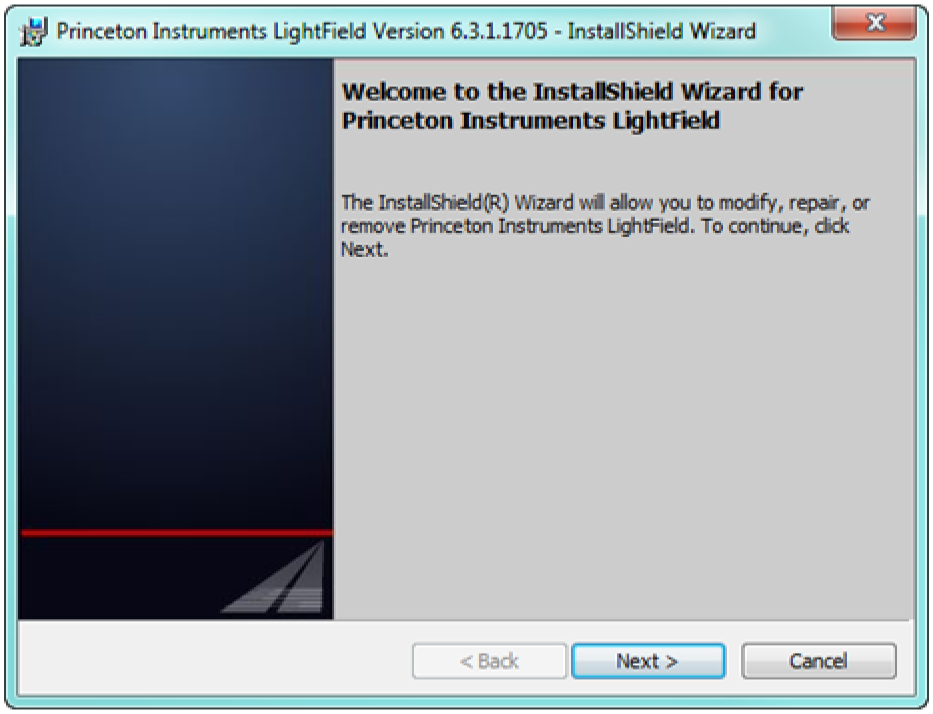
Uninstall or change a program

To uninstall a program, select it from the list and then click Uninstall, Change, or Repair.

Name	Publisher	Installed On
OCR Software by I.R.I.S. 14.5	HP	6/5/2013
Pleora eBUS Support	Princeton Instruments	9/7/2016
PowerDVD DX	CyberLink Corp.	3/19/2013
Princeton Instruments LightField	Princeton Instruments	6/29/2017
Princeton Instru	Princeton Instruments	12/9/2016
Princeton Instru	Princeton Instruments	6/29/2017
PVCam Release	Photometrics and QImaging	6/29/2017
Python 3.6.3 (64-bit)	Python Software Foundation	11/2/2017
Python Launcher	Python Software Foundation	11/2/2017
QuickTime 7	Apple Inc.	6/17/2016

3. The **InstallShield Wizard** will launch. Follow on-screen prompts to modify the LightField Installation to include **LightField Add-in and Automation SDK**. See Figure 2-3.

Figure 2-3: Typical InstallShield Wizard Dialog



Princeton Instruments LightField Version 6.3.1.1705 - InstallShield Wizard

Welcome to the InstallShield Wizard for Princeton Instruments LightField

The InstallShield(R) Wizard will allow you to modify, repair, or remove Princeton Instruments LightField. To continue, click Next.

2.2 Create Default Experiment

It is recommended that a baseline, default LightField experiment be created and saved prior to beginning any automation application development. Doing so will reduce the number of preliminary and initial configuration steps required by the automation application.

Perform the following procedure to create a default experiment that will serve as the basic experiment for automation routines:

1. Launch LightField.
2. Once LightField has finished initializing, verify that all devices are properly installed and have been placed on the Device Grid within the Experiment Workspace.
3. Configure a baseline set of experiment parameters.

   These should be appropriate for the specific experiment and application for which an automation application is to be developed.
4. Once the baseline configuration is set, save the experiment with a descriptive name that will be easily recognizable for future use. The name of this file will be required as an input when developing the Python™ automation application.

This page is intentionally blank.

Chapter 3: Build a Simple Automation Application

Once a preliminary LightField project template has been created and saved as described in Chapter 2, Prepare LightField, on page 7, development of an automation application can begin.

NOTE:

This document is not intended to be a tutorial about using or programming in Python™. It is written with the assumption that the reader possesses a basic knowledge of the Python™ programming language.

3.1 Development Tools

The recommended way to build Python™ applications is using the Python™ Integrated Development and Learning Environment (IDLE). IDLE provides two primary window types/functions:

- Editor;

  The Editor provides the workspace in which application coding is performed.
- Shell.

  The Shell window is used to execute Python™ applications.

NOTE:

For proper application execution, LightField must be running and a supported device must be connected to the Host Computer.

Key features of IDLE include:

- Supports having multiple Editor windows open simultaneously;
- Depressing the F5 key from within the Editor window will launch and run the current code/sample.

Complete information about IDLE is available at the following website:

`https://docs.python.org/2/library/idle.html`

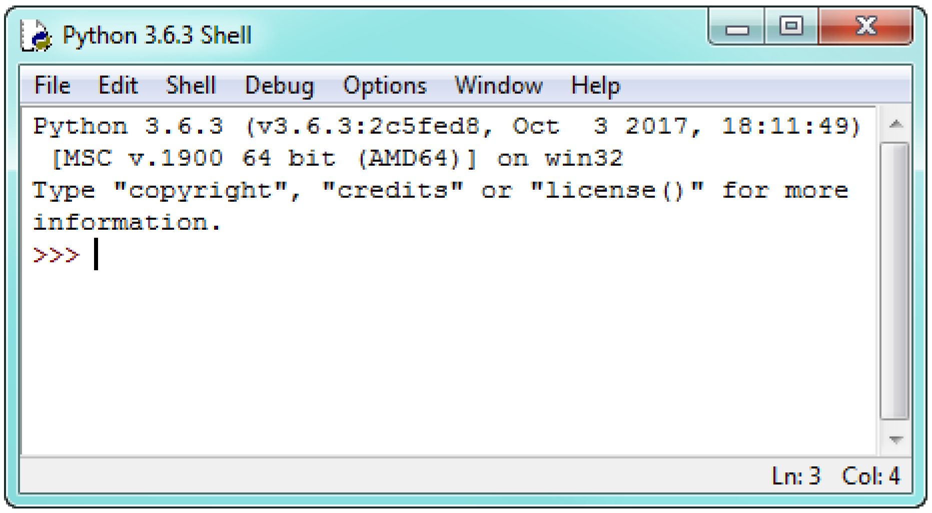
3.2 Sample Application Development

Perform the following procedure to develop a simple automation application that will:

- Launch LightField;
- Modify the experiment exposure time;
- Acquire an image;
- Close LightField.

1. From the host computer's desktop, launch IDLE. A new Python™ Shell Window is opened. See Figure 3-1.

Figure 3-1: Typical New Python™ Shell Window

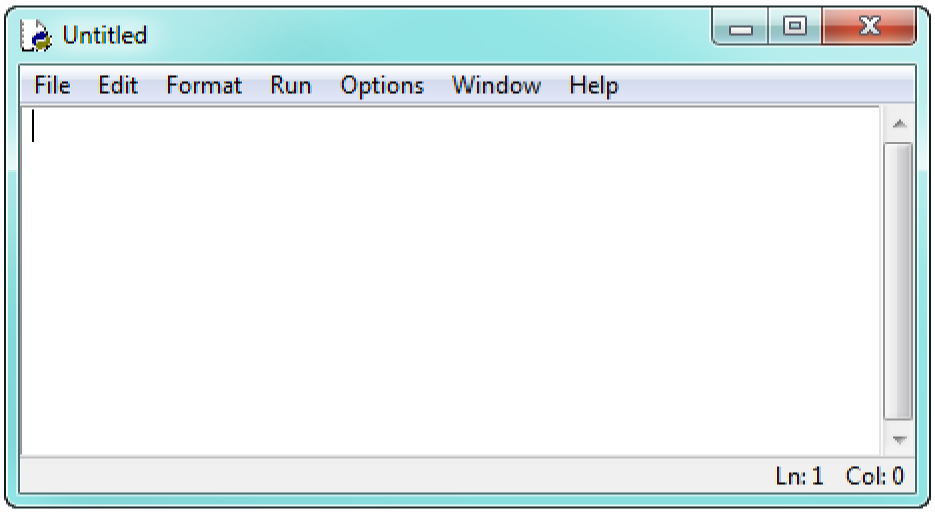


```
Python 3.6.3 (v3.6.3:2c5fed8, Oct  3 2017, 18:11:49)
 [MSC v.1900 64 bit (AMD64)] on win32
Type "copyright", "credits" or "license()" for more
information.
>>> 
```

4411-0163_001

2. From the menu bar, select **File ► New File**, or type Ctrl + N. A new Python™ Editor window is opened. See Figure 3-2.

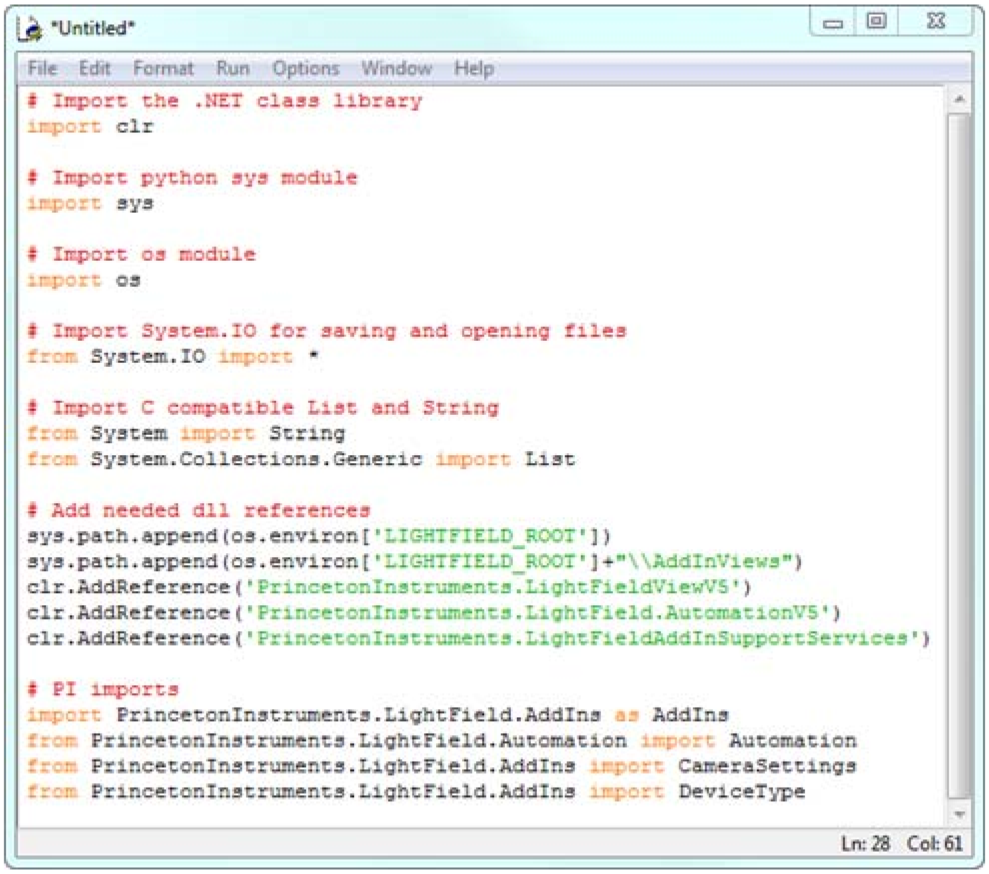
Figure 3-2: Typical New Python™ Editor Window



4411-0163_002

3. Within the **Editor** window, add the required **Imports** and **DLL References**. See Figure 3-3.

Figure 3-3: Typical Python™ Code Snippet: Required Imports



```
# Import the .NET class library
import clr

# Import python sys module
import sys

# Import os module
import os

# Import System.IO for saving and opening files
from System.IO import *

# Import C compatible List and String
from System import String
from System.Collections.Generic import List

# Add needed dll references
sys.path.append(os.environ['LIGHTFIELD_ROOT'])
sys.path.append(os.environ['LIGHTFIELD_ROOT']+"\\AddInViews")
clr.AddReference('PrincetonInstruments.LightFieldViewV5')
clr.AddReference('PrincetonInstruments.LightField.AutomationV5')
clr.AddReference('PrincetonInstruments.LightFieldAddInSupportServices')

# PI imports
import PrincetonInstruments.LightField.AddIns as AddIns
from PrincetonInstruments.LightField.Automation import Automation
from PrincetonInstruments.LightField.AddIns import CameraSettings
from PrincetonInstruments.LightField.AddIns import DeviceType
```

4. Next, add the following code to verify a camera setting, such as **Analog Gain**:

```
def set_value(setting, value):
# Check for existence before setting
# gain, adc rate, or adc quality
if experiment.Exists(setting):
   experiment.SetValue(setting, value)
```

If the specified camera setting does not exist, the `experiment.SetValue` function will not be executed.

5. Next, add the following code to locate a connected device:

```
def device_found():
   # Find connected device
   for device in experiment.ExperimentDevices:
      if (device.Type == DeviceType.Camera):
         return True

# If connected device is not a camera inform the user
print("Camera not found. Please add a camera and try again.")
return False
```

6. Add the following code to create an instance of LightField that loads with no experiment:

```
# Create the LightField Application (true for visible)
# The 2nd parameter forces LF to load with no experiment
auto = Automation(True, List[String]())
```

7. Next, add the following code to get an experiment object:

```
# Get experiment object
experiment = auto.LightFieldApplication.Experiment
```

8. Finally, add the following code to set the exposure time and acquire an image:

```
if (device_found()==True):
   #Set exposure time
   set_value(CameraSettings.ShutterTimingExposureTime, 20.0)

   # Acquire image
   experiment.Acquire()
```

Chapter 4: Using LightField Experiment Settings Help File

Information required to modify LightField settings using Python™ code is in the **LightField Experiment Settings.chm** help file found in the **Add-in and Automation SDK** folder.

The help file includes the following Classes available for configuration updates via Python™:

- `CameraSettings`
- `ExperimentSettings`
- `SpectrometerSettings`

This chapter describes how to identify this information when creating a Python™ automation application.

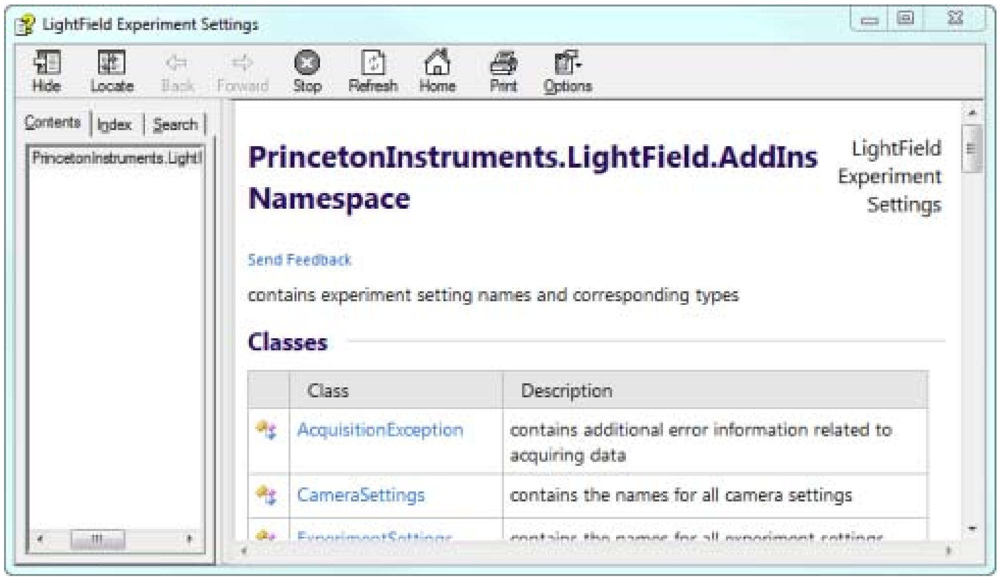
4.1 Identify Device Parameters and Valid Enumeration Values

Perform the following procedure to identify required parameters and associated valid enumeration values required in order to develop Python™ automation applications.

NOTE:
In this example, the parameter **Analog to Digital Conversion Quality** will be spotlighted.

1. Open the **Add-in and Automation SDK** folder using the shortcut on the host computer's desktop.
2. Locate the file named **LightField Experiment Settings.chm** and double-click on it to open the Help file. See Figure 4-1.

Figure 4-1: Typical *LightField Experiment Settings.chm* Help File



PrincetonInstruments.LightField.AddIns Namespace

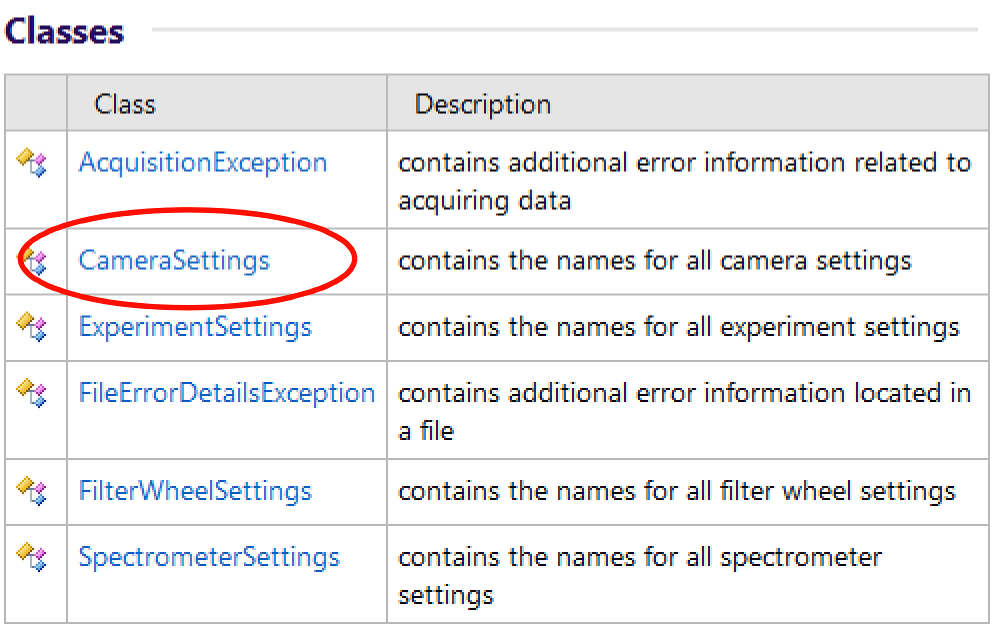
contains experiment setting names and corresponding types

Classes

Class	Description
AcquisitionException	contains additional error information related to acquiring data
CameraSettings	contains the names for all camera settings
ExperimentSettings	contains the names for all experiment settings

3. The `ADC Quality` setting is a **Camera** setting. Within the `Classes` table, click on `CameraSettings`. See Figure 4-2.

Figure 4-2: Typical Classes Table

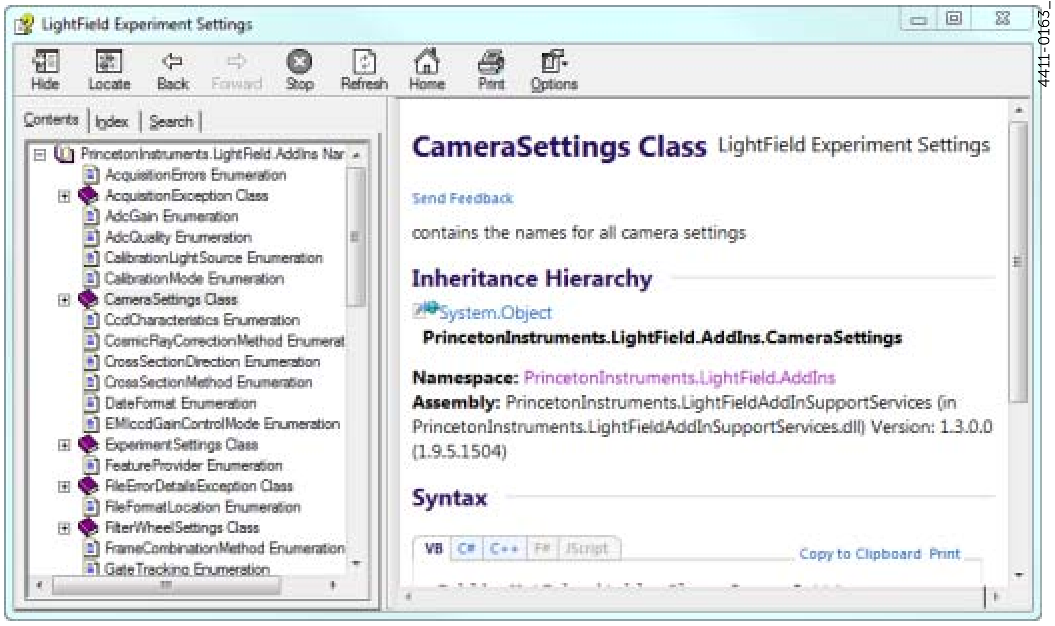

4411-0163_007

Classes

	Class	Description
	AcquisitionException	contains additional error information related to acquiring data
	CameraSettings	contains the names for all camera settings
	ExperimentSettings	contains the names for all experiment settings
	FileErrorDetailsException	contains additional error information located in a file
	FilterWheelSettings	contains the names for all filter wheel settings
	SpectrometerSettings	contains the names for all spectrometer settings

Information about the `CameraSettings` class is displayed in the Help window primary frame, and the Contents tab expands. See Figure 4-3.

Figure 4-3: CameraSettings Class Information



PrincetonInstruments.LightField.AddIns Nar
AcquisitionErrors Enumeration
AcquisitionException Class
AdcGain Enumeration
AdcQuality Enumeration
CalibrationLightSource Enumeration
CalibrationMode Enumeration
CameraSettings Class
CcdCharacteristics Enumeration
CosmicRayCorrectionMethod Enumerat
CrossSectionDirection Enumeration
CrossSectionMethod Enumeration
DateFormat Enumeration
EMIccdGainControlMode Enumeration
ExperimentSettings Class
FeatureProvider Enumeration
FileErrorDetailsException Class
FileFormatLocation Enumeration
FilterWheelSettings Class
FrameCombinationMethod Enumeration
GateTracking Enumeration

CameraSettings Class LightField Experiment Settings

Send Feedback

contains the names for all camera settings

Inheritance Hierarchy

System.Object
PrincetonInstruments.LightField.AddIns.CameraSettings

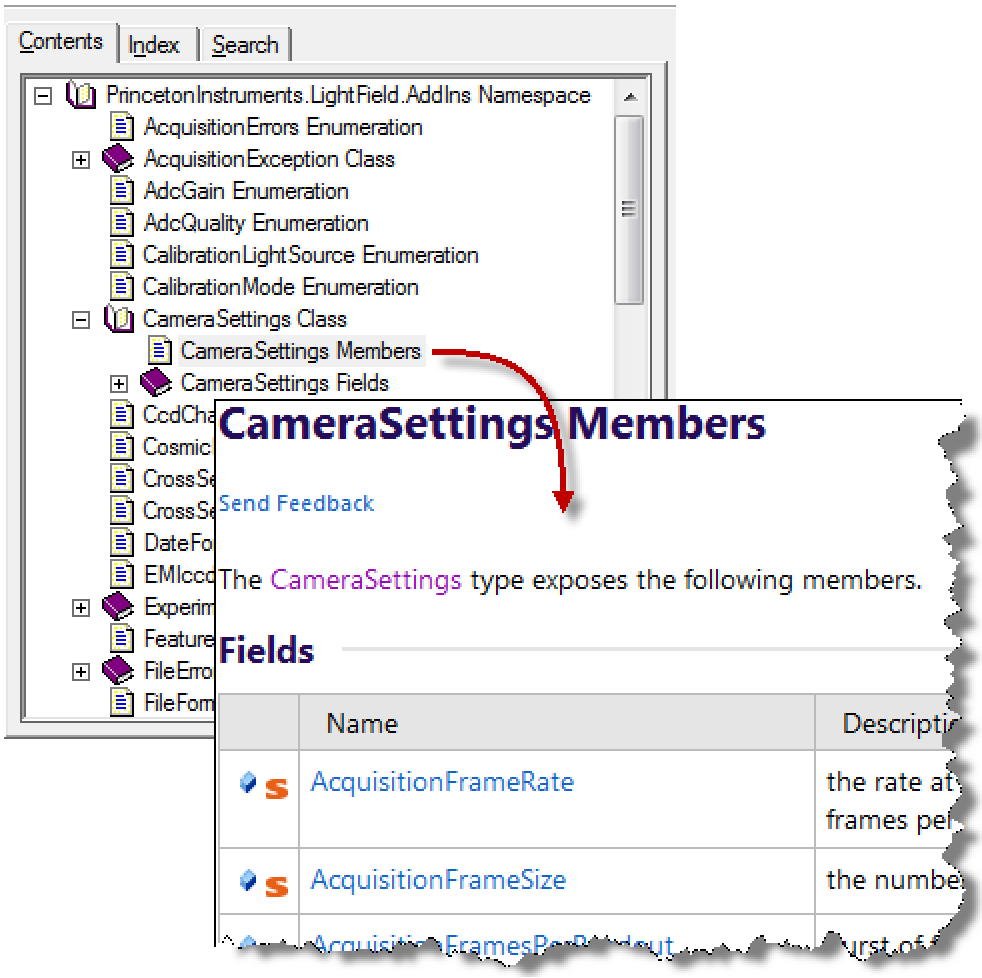
Namespace: PrincetonInstruments.LightField.AddIns
Assembly: PrincetonInstruments.LightFieldAddInSupportServices (in PrincetonInstruments.LightFieldAddInSupportServices.dll) Version: 1.3.0.0 (1.9.5.1504)

Syntax

VB C# C++ F# JScript Copy to Clipboard Print

4. Within the **Contents** tab, expand the `CameraSettings Class` book and click on the `CameraSettings Members` entry to view the members within the primary help panel. Figure 4-4 illustrates a partial view of the `CameraSettings Members` table.

Figure 4-4: Partial View: CameraSettings Members Table



- PrincetonInstruments.LightField.AddIns Namespace
  - AcquisitionErrors Enumeration
  - AcquisitionException Class
  - AdcGain Enumeration
  - AdcQuality Enumeration
  - CalibrationLightSource Enumeration
  - CalibrationMode Enumeration
  - CameraSettings Class
    - CameraSettings Members
    - CameraSettings Fields
  - CcdCha
  - Cosmic
  - CrossSe
  - CrossSe
  - DateFo
  - EMIccd
  - Experim
  - Feature
  - FileErro
  - FileForm

CameraSettings Members

Send Feedback

The CameraSettings type exposes the following members.

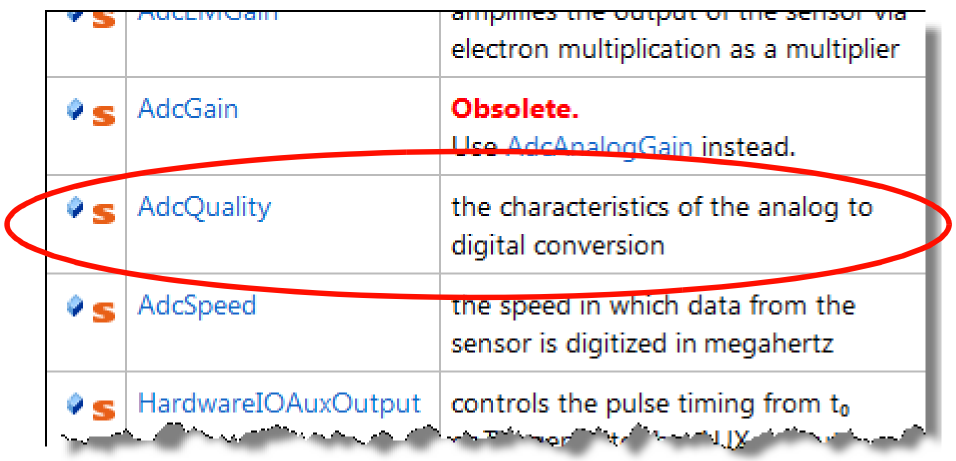
Fields

	Name	Descripti
S	AcquisitionFrameRate	the rate at frames pe
S	AcquisitionFrameSize	the numbe

4411-0163_009

5. Scroll the `CameraSettings Members` table until `AdcQuality` is visible. See Figure 4-5.

Figure 4-5: CameraSettings Members: AdcQuality Entry

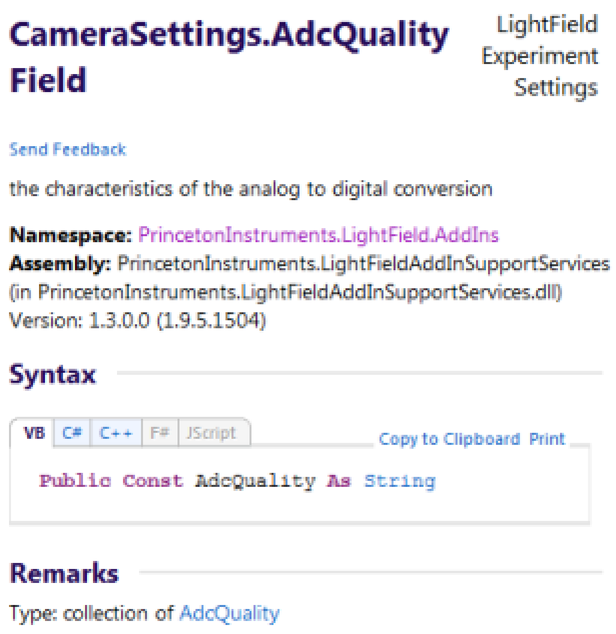


S	AdcEMGain	amplifies the output of the sensor via electron multiplication as a multiplier
S	AdcGain	**Obsolete.** Use AdcAnalogGain instead.
S	AdcQuality	the characteristics of the analog to digital conversion
S	AdcSpeed	the speed in which data from the sensor is digitized in megahertz
S	HardwareIOAuxOutput	controls the pulse timing from t_0

4411-0163_010

6. Click on `AdcQuality` to view information about this camera setting in the primary help panel. See Figure 4-6.

Figure 4-6: AdcQuality Camera Setting Information



CameraSettings.AdcQuality Field

LightField Experiment Settings

Send Feedback

the characteristics of the analog to digital conversion

Namespace: PrincetonInstruments.LightField.AddIns
Assembly: PrincetonInstruments.LightFieldAddInSupportServices (in PrincetonInstruments.LightFieldAddInSupportServices.dll)
Version: 1.3.0.0 (1.9.5.1504)

Syntax

VB | C# | C++ | F# | JScript — Copy to Clipboard Print

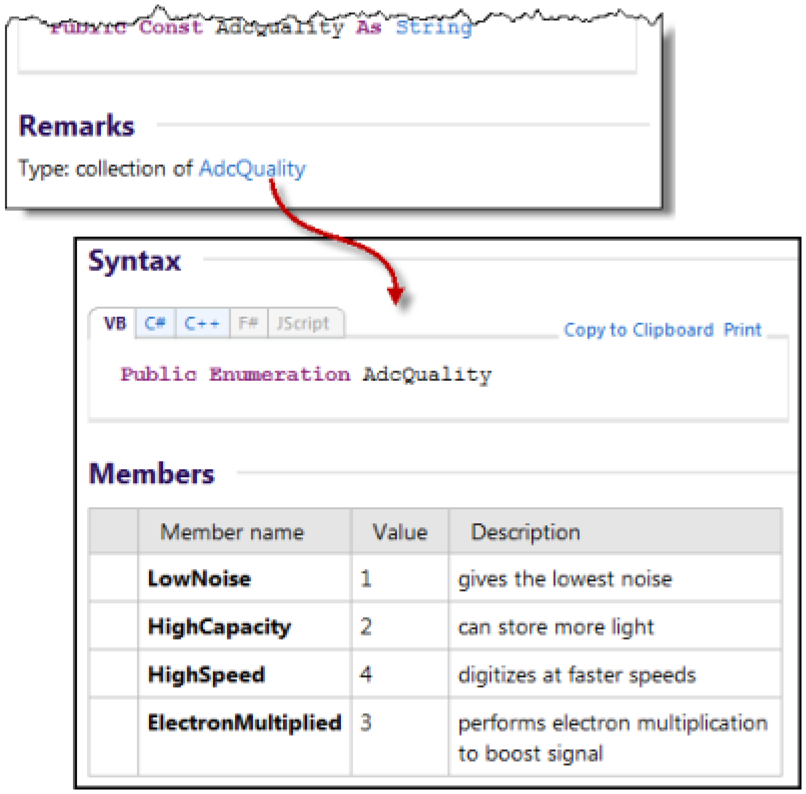
```
Public Const AdcQuality As String
```

Remarks

Type: collection of AdcQuality

7. Within the **Remarks** section, click on the `AdcQuality` link to view the set of valid Enumeration values/members. See Figure 4-7.

Figure 4-7: AdcQuality Camera Setting Information



```
Public Const AdcQuality As String
```

Remarks

Type: collection of AdcQuality

Syntax

VB | C# | C++ | F# | JScript — Copy to Clipboard Print

```
Public Enumeration AdcQuality
```

Members

	Member name	Value	Description
	LowNoise	1	gives the lowest noise
	HighCapacity	2	can store more light
	HighSpeed	4	digitizes at faster speeds
	ElectronMultiplied	3	performs electron multiplication to boost signal

All information required to modify `AdcQuality` using Python™ is now known.

Chapter 5: Sample Applications

Teledyne Princeton Instruments includes a library of sample Python™ applications. This chapter provides information about each of these sample applications.

Refer to Table 5-1 for a list of pre-written applications that are available and where additional information can be found.

Table 5-1: Sample Teledyne Princeton Instruments Python™ Applications

File/Application Name	For Complete Information, refer to...
`acquisition_filename.py`	Section 5.2.1, acquisition_filename.py, on page 20
`add_remove-device.py`	Section 5.2.2, add_remove-device.py, on page 21
`emccd.py`	Section 5.2.3, emccd.py, on page 21
`event_handling.py`	Section 5.2.4, event_handling.py, on page 21
`exposure_acquire.py`	Section 5.2.5, exposure_acquire.py, on page 21
`multiple_lightfields.py`	Section 5.2.6, multiple_lightfields.py, on page 21
`multiple_roi.py`	Section 5.2.7, multiple_roi.py, on page 22
`repetitive_gating.py`	Section 5.2.8, repetitive_gating.py, on page 22
`saving_loading_experiments.py`	Section 5.2.9, saving_loading_experiments.py, on page 22
`sequential_gating.py`	Section 5.2.10, sequential_gating.py, on page 22
`settings.py`	Section 5.2.11, settings.py, on page 23
`spe_file_access.py`	Section 5.2.12, spe_file_access.py, on page 23
`spectrometer.py`	Section 5.2.13, spectrometer.py, on page 23
synchronous_acquisition.py	Section 5.2.14, synchronous_acquisition.py, on page 23
`temperature.py`	Section 5.2.15, temperature.py, on page 24

5.1 Common Program Elements

This section provides information about program elements that are required by all Python™ applications designed to automate LightField:

Refer to the following sections for complete information:

- Section 5.1.1, Required DLLs, on page 20;
- Section 5.1.2, Required Imports, on page 20.

5.1.1 Required DLLs

The set of required DLLs is declared in the following block of code near the beginning of each Python™ file:

```
# Add needed dll references
sys.path.append(os.environ['LIGHTFIELD_ROOT'])
sys.path.append(os.environ['LIGHTFIELD_ROOT']+"\\AddInViews")
clr.AddReference('PrincetonInstruments.LightFieldViewV5')
clr.AddReference('PrincetonInstruments.LightField.AutomationV5')
clr.AddReference('PrincetonInstruments.LightFieldAddInSupport
    Services')
```

5.1.2 Required Imports

The following required elements are imported by all Python™ applications:

- `Addins`;
- `Automation`.

The set of required imports is declared near the beginning of each Python™ file in a block of code resembling the following:

```
# PI imports
from PrincetonInstruments.LightField.Automation import Automation
from PrincetonInstruments.LightField.AddIns import [Name_of_import]

[additional_import_statements]
```

where `[Name_of_import]` can be a/an:

- `class;`
- `enumeration;`
- `variable;`
- `etc.`

Application-specific import requirements are provided in the following section.

5.2 Application-Specific Information

This section provides information specific to each sample application supplied by Teledyne Princeton Instruments.

5.2.1 `acquisition_filename.py`

Description

When called, `acquisition_filename.py` saves an acquired image in the default LightField directory.

Required Imports

The following additional imports are required by `acquisition_filename.py`:

- `ExperimentSettings`
- `DeviceType`

5.2.2 `add_remove-device.py`

Description

When called, `add_remove-device.py` instructs LightField to add the first available device. Once the current device is no longer required, LightField then removes it.

Required Imports

There are no additional imports required by `add_remove-device.py`.

5.2.3 `emccd.py`

Description

When called, `emccd.py` configures **EM ADC Quality** and the first **ADC Rate** available. It also sets the **EM Gain** to 10, and acquires an image.

Required Imports

The following additional imports are required by `emccd.py`:

- `CameraSettings`
- `AdcQuality`
- `DeviceType`

5.2.4 `event_handling.py`

Description

When called, `event_handling.py` informs the user when LightField is ready to run.

Required Imports

The following additional imports are required by `event_handling.py`:

- `DeviceType`

5.2.5 `exposure_acquire.py`

Description

When called, `exposure_acquire.py` configures LightField and acquires an exposure.

Required Imports

The following additional imports are required by `exposure_acquire.py`:

- `CameraSettings`
- `DeviceType`

5.2.6 `multiple_lightfields.py`

Description

When called, `multiple_lightfields.py` enables the acquisition of image data from two discreet instances of LightField.

Required Imports

The following additional imports are required by `multiple_lightfields.py`:

- `IDevice`
- `DeviceType`
- `ExperimentSettings`

5.2.7 `multiple_roi.py`

Description

When called, `multiple_roi.py` defines two regions of interest within LightField. The dimensions of each ROI are printed. These ROIs are then used to acquire an image.

Required Imports

The following additional imports are required by `multiple_roi.py`:

- `RegionOfInterest`
- `DeviceType`

5.2.8 `repetitive_gating.py`

Description

When called, `repetitive_gating.py`:

- Sets and gets a value for on-chip accumulations;
- Configures Repetitive Gate Pulse Width and Delay;
- Acquires an image.

Required Imports

The following additional imports are required by `repetitive_gating.py`:

- `CameraSettings`
- `GatingMode`
- `Pulse`
- `DeviceType`

5.2.9 `saving_loading_experiments.py`

Description

When called, `saving_loading_experiments.py` saves an experiment and prints saved experiments.

Required Class Imports

There are no additional imports are required by `saving_loading_experiments.py`.

5.2.10 `sequential_gating.py`

Description

When called, `sequential_gating.py`:

- Sets and gets a value for on-chip accumulations;
- Configures starting and ending sequential pulse width and delay;
- Configures the number of frames;
- Acquires an image.

Required Imports

The following additional imports are required by `sequential_gating.py`:

- `CameraSettings`
- `ExperimentSettings`
- `GatingMode`
- `Pulse`
- `DeviceType`

5.2.11 `settings.py`

Description

When called, `settings.py` displays a device's capabilities.

Required Imports

The following additional imports are required by `settings.py`:

- `CameraSettings`
- `DeviceType`

5.2.12 `spe_file_access.py`

Description

When called, `spe_file_access.py`:

- Acquires an image;
- Opens the corresponding `*.spe` file;
- Returns the image dimensions, in pixels (e.g., 1024 x 1024);
- Reports the intensity of the first 10 pixels.

Required Imports

The following additional imports are required by `spe_file_access.py`:

- `ExperimentSettings`
- `DeviceType`

5.2.13 `spectrometer.py`

Description

When called, `spectrometer.py` configures an attached spectrometer device.

Required Imports

The following additional imports are required by `spectrometer.py`:

- `SpectrometerSettings`
- `DeviceType`

5.2.14 `synchronous_acquisition.py`

Description

When called, `synchronous_acquisition.py`:

- Synchronously Acquires 3 images using LightField's Capture function;
- Processes data using the Numpy extension module.

Perform the following procedure to install Numpy:

1. Open a Command Prompt.
2. Change the working directory to be the Python™ directory.
   For example:
   ```
   cd C:\Users\[user_name]AppData\Local\Programs\Python\Python36
   ```
3. Enter the following command:
   ```
   python -m pip install numpy [enter]
   ```
4. Wait for **Successfully Installed** to be displayed.

Required Imports

The following additional imports are required by `synchronous_acquisition.py`:

- `Numpy;`
- `ExperimentSettings;`
- `DeviceType.`

NOTES:

If Numpy is not installed, open the sample code, locate the following lines, and comment them out:

```
# numpy import
import numpy as np
```

```
def print_statistics(image_data, image_frame):

     # Create a new 1-dimensional array from an iterable object of
        type float.

     # Image data can be returned as uint[], ushort[], float[]

     # Use float for best results with/without background subtraction
        array = np.fromiter(image_data, float)

     # Calculated mean
        print("\nMean: %s" % str(round(np.mean(array))))

     # Calculated median
     print("\nMedian: %s" % str(round(np.median(array))))

     # Calculated max
     print("\nMax Value: %s" % str(round(np.max(array))))

     # Calculated min
     print("\nMin Value: %s" % str(np.min(array)))
```

```
print_statitstics(image_data, image_frame)
```

5.2.15 `temperature.py`

Description

When called, `temperature.py`:

- Changes the temperature set point to −10 °C;
- Reports the current temperature;
- Reports the current status (i.e., locked/unlocked.)

Required Imports

The following additional imports are required by `temperature.py`:

- `CameraSettings`
- `SensorTemperatureStatus`
- `DeviceType`

Appendix A: Install Python™

This appendix provides the information necessary to install Python™ and Python.Net on a host computer in order to automate LightField 6.

A.1 Install Python™

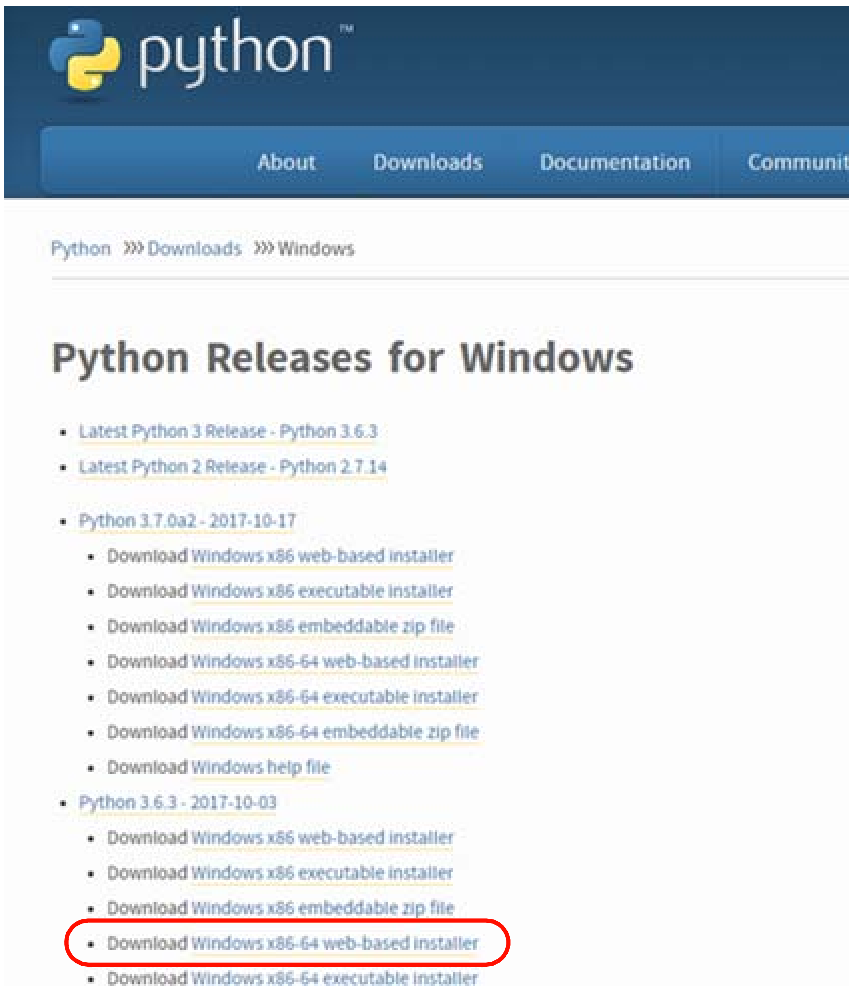
Perform the following procedure to install Python™ on the host computer:

1. Download the most recent 64-bit version of Python™ from here:

   https://www.python.org/downloads/windows/

NOTE:
As of the date of this document, the most recent 64-bit version is Python™ 3.6.3.

Figure A-1: List of Available Python™ Versions

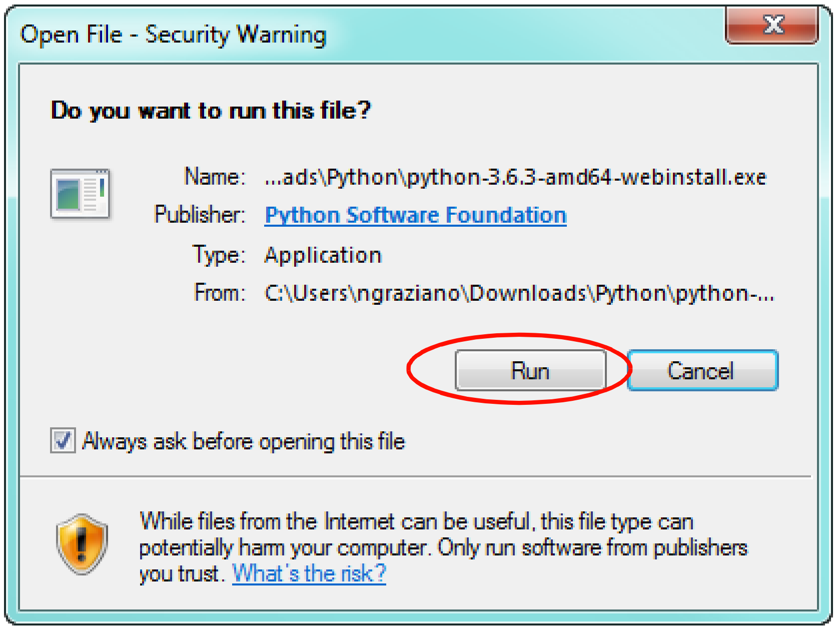


Python Releases for Windows

- Latest Python 3 Release - Python 3.6.3
- Latest Python 2 Release - Python 2.7.14
- Python 3.7.0a2 - 2017-10-17
  - Download Windows x86 web-based installer
  - Download Windows x86 executable installer
  - Download Windows x86 embeddable zip file
  - Download Windows x86-64 web-based installer
  - Download Windows x86-64 executable installer
  - Download Windows x86-64 embeddable zip file
  - Download Windows help file
- Python 3.6.3 - 2017-10-03
  - Download Windows x86 web-based installer
  - Download Windows x86 executable installer
  - Download Windows x86 embeddable zip file
  - Download Windows x86-64 web-based installer
  - Download Windows x86-64 executable installer

2. When the **Security Warning** dialog is displayed, click **Run**. See Figure A-2.

Figure A-2: Open File - Security Warning



Open File - Security Warning

Do you want to run this file?

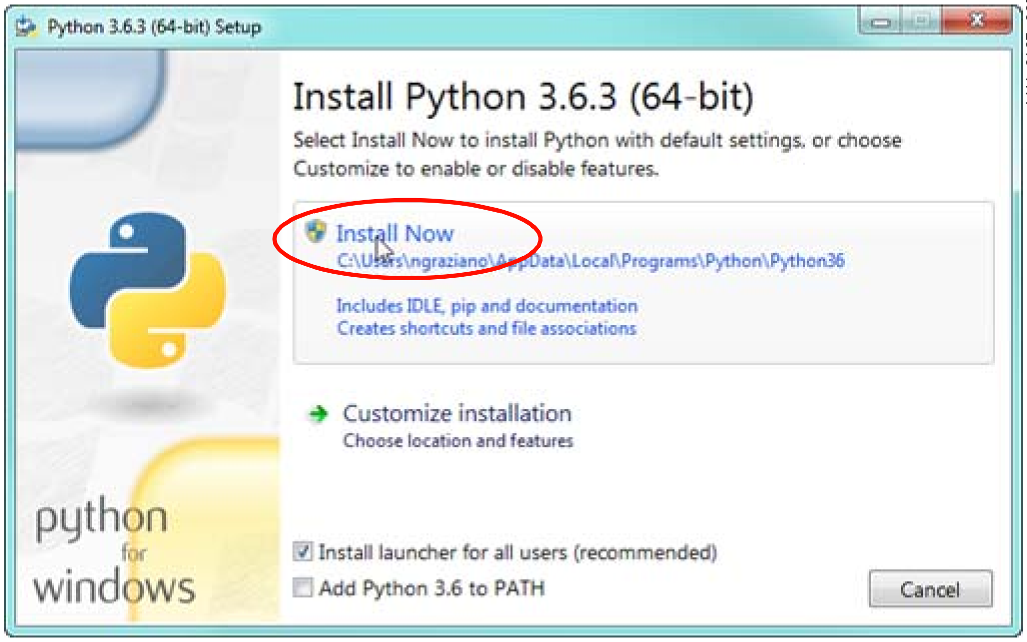
Name: ...ads\Python\python-3.6.3-amd64-webinstall.exe
Publisher: Python Software Foundation
Type: Application
From: C:\Users\ngraziano\Downloads\Python\python-...

☑ Always ask before opening this file

While files from the Internet can be useful, this file type can potentially harm your computer. Only run software from publishers you trust. What's the risk?

3. When prompted, click **Install Now**. See Figure A-3.

Figure A-3: Typical Python™ Setup Dialog



Python 3.6.3 (64-bit) Setup

Install Python 3.6.3 (64-bit)

Select Install Now to install Python with default settings, or choose Customize to enable or disable features.

Install Now
C:\Users\ngraziano\AppData\Local\Programs\Python\Python36

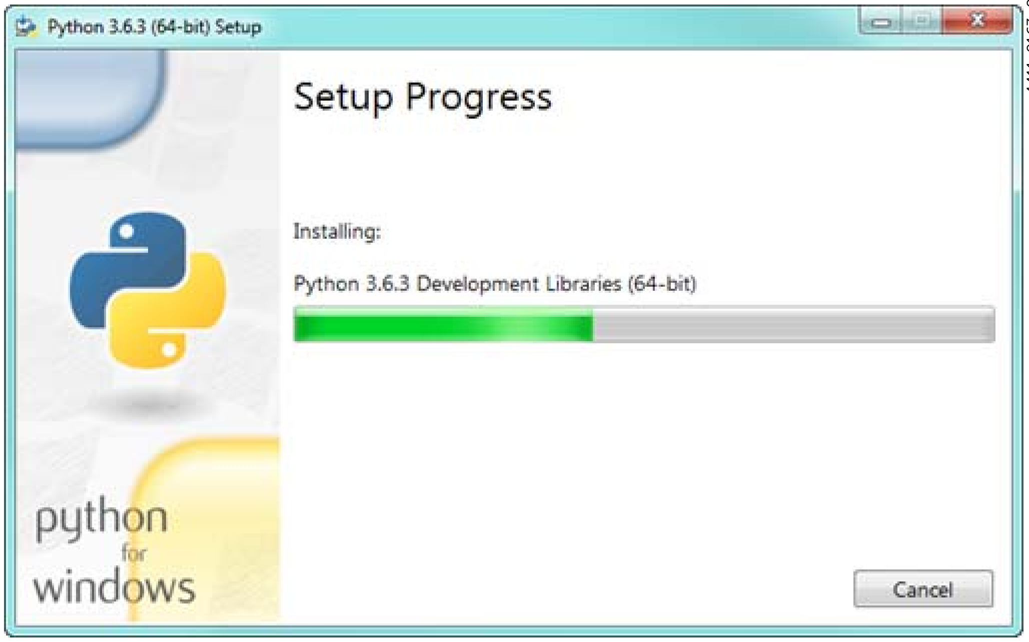
Includes IDLE, pip and documentation
Creates shortcuts and file associations

Customize installation
Choose location and features

☑ Install launcher for all users (recommended)
☐ Add Python 3.6 to PATH

4. The installation will begin, and the progress is indicated with a standard Windows progress bar. See Figure A-4.

Figure A-4: Typical Installation Progress Bar



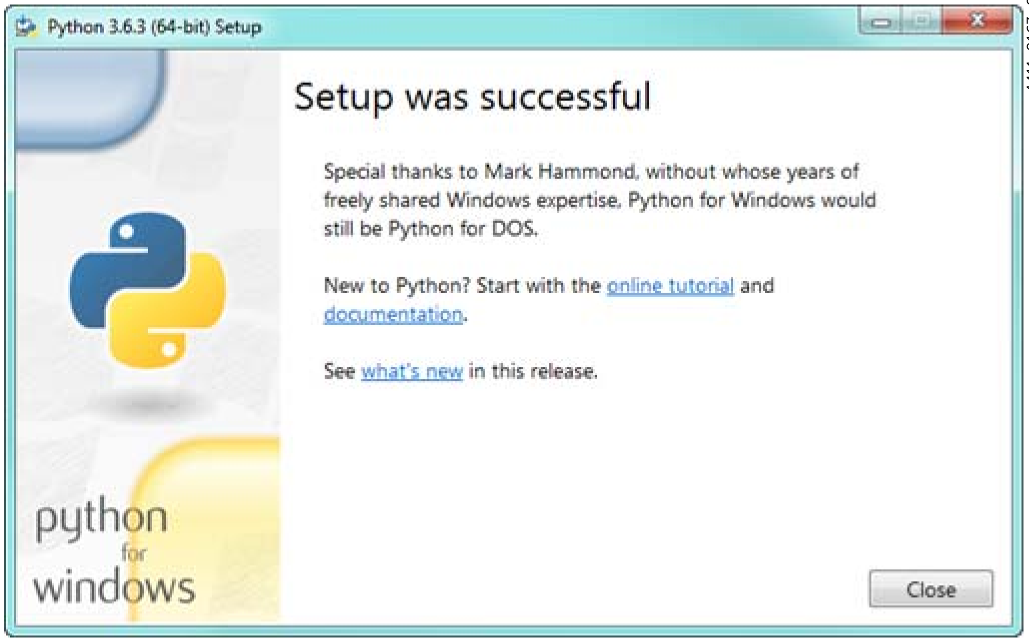
Setup Progress

Installing:

Python 3.6.3 Development Libraries (64-bit)

5. When installation is complete, click **Close**. See Figure A-5.

Figure A-5: Typical Setup Success Dialog



Setup was successful

Special thanks to Mark Hammond, without whose years of freely shared Windows expertise, Python for Windows would still be Python for DOS.

New to Python? Start with the online tutorial and documentation.

See what's new in this release.

A.2 Install Python for .Net

Perform the following procedure to install PythonNet:

1. Download the most recent 64-bit version **Python for .Net** which is available here:

   https://pypi.python.org/pypi/pythonnet/2.3.0

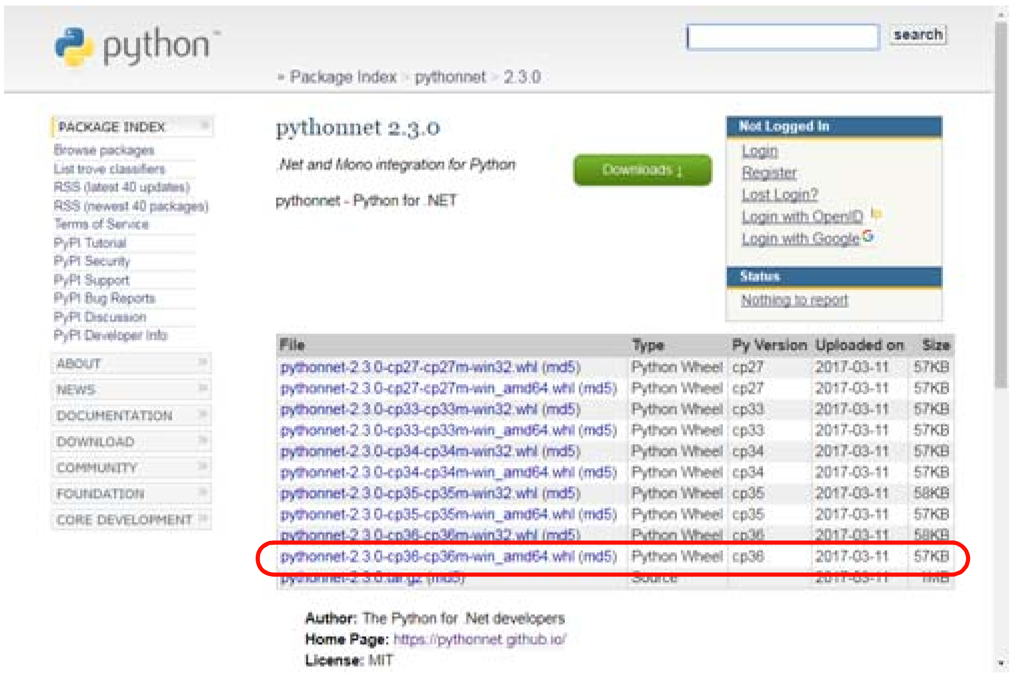
   NOTE:

   As of the date of this document, the most recent 64-bit version is:

   ```
   pythonnet-2.3.0-cp36-cp36m-win_amd64.whl
   ```

   See Figure A-6.

Figure A-6: List of Available Python for .Net Download Packages



pythonnet 2.3.0

.Net and Mono integration for Python

pythonnet - Python for .NET

File	Type	Py Version	Uploaded on	Size
pythonnet-2.3.0-cp27-cp27m-win32.whl (md5)	Python Wheel	cp27	2017-03-11	57KB
pythonnet-2.3.0-cp27-cp27m-win_amd64.whl (md5)	Python Wheel	cp27	2017-03-11	57KB
pythonnet-2.3.0-cp33-cp33m-win32.whl (md5)	Python Wheel	cp33	2017-03-11	57KB
pythonnet-2.3.0-cp33-cp33m-win_amd64.whl (md5)	Python Wheel	cp33	2017-03-11	57KB
pythonnet-2.3.0-cp34-cp34m-win32.whl (md5)	Python Wheel	cp34	2017-03-11	57KB
pythonnet-2.3.0-cp34-cp34m-win_amd64.whl (md5)	Python Wheel	cp34	2017-03-11	57KB
pythonnet-2.3.0-cp35-cp35m-win32.whl (md5)	Python Wheel	cp35	2017-03-11	58KB
pythonnet-2.3.0-cp35-cp35m-win_amd64.whl (md5)	Python Wheel	cp35	2017-03-11	57KB
pythonnet-2.3.0-cp36-cp36m-win32.whl (md5)	Python Wheel	cp36	2017-03-11	58KB
pythonnet-2.3.0-cp36-cp36m-win_amd64.whl (md5)	Python Wheel	cp36	2017-03-11	57KB
pythonnet-2.3.0.tar.gz (md5)	Source		2017-03-11	1MB

Author: The Python for .Net developers
Home Page: https://pythonnet.github.io/
License: MIT

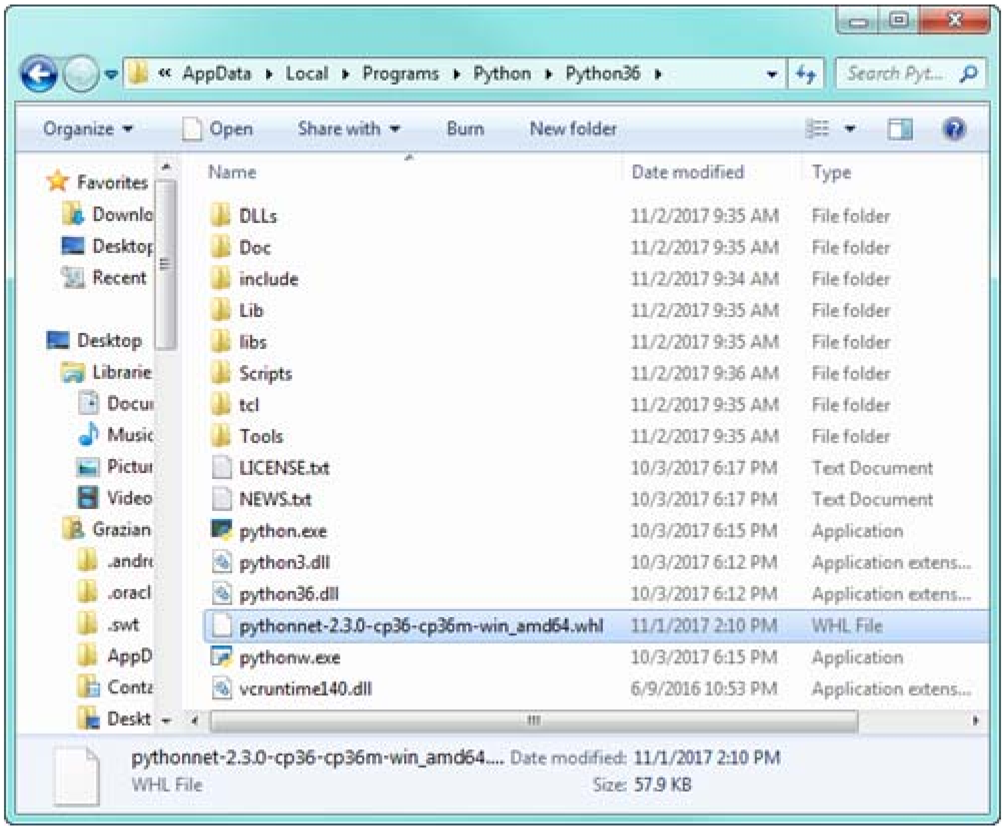
2. Open a Windows Explorer window and navigate to the directory in which the **Python for .Net** file was saved.

3. Open a second Explorer window and navigate to the Python™ installation directory on the host computer. This is typically:

   ```
   C:\Users\[user_name]\AppData\Local\Programs\Python\Python36
   ```

4. Move or place a copy of the **Python for .Net** file into the Python™ installation directory. See Figure A-7.

Figure A-7: Python™ Installation Directory with Python for .Net File

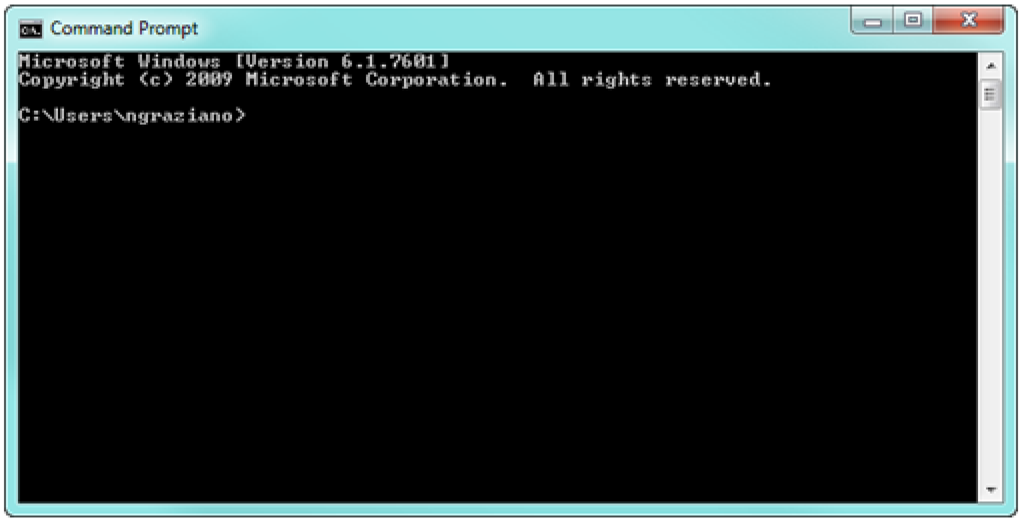


Name	Date modified	Type
DLLs	11/2/2017 9:35 AM	File folder
Doc	11/2/2017 9:35 AM	File folder
include	11/2/2017 9:34 AM	File folder
Lib	11/2/2017 9:35 AM	File folder
libs	11/2/2017 9:35 AM	File folder
Scripts	11/2/2017 9:36 AM	File folder
tcl	11/2/2017 9:35 AM	File folder
Tools	11/2/2017 9:35 AM	File folder
LICENSE.txt	10/3/2017 6:17 PM	Text Document
NEWS.txt	10/3/2017 6:17 PM	Text Document
python.exe	10/3/2017 6:15 PM	Application
python3.dll	10/3/2017 6:12 PM	Application extens...
python36.dll	10/3/2017 6:12 PM	Application extens...
pythonnet-2.3.0-cp36-cp36m-win_amd64.whl	11/1/2017 2:10 PM	WHL File
pythonw.exe	10/3/2017 6:15 PM	Application
vcruntime140.dll	6/9/2016 10:53 PM	Application extens...

pythonnet-2.3.0-cp36-cp36m-win_amd64.... Date modified: 11/1/2017 2:10 PM
WHL File Size: 57.9 KB

5. Open a **Command Prompt** window.

Figure A-8: Typical Command Prompt Window



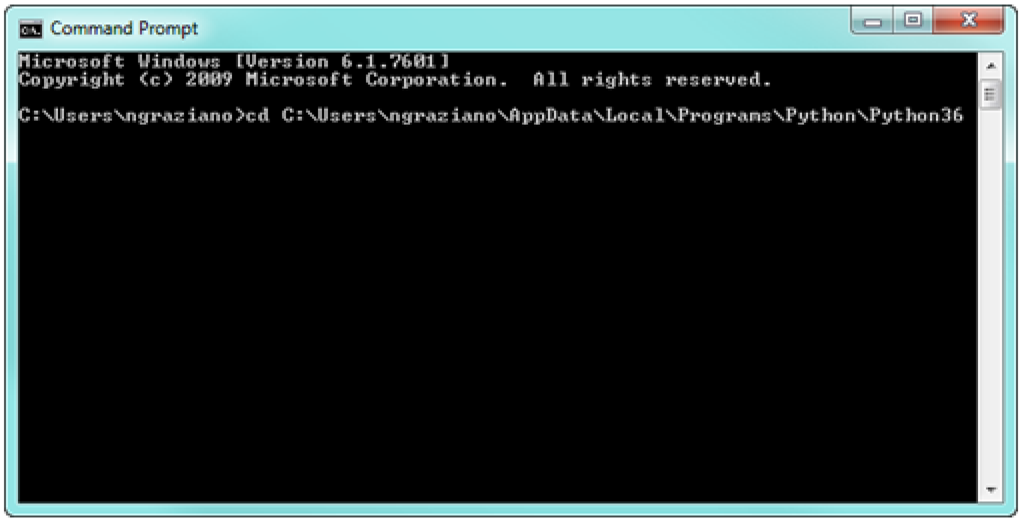
```
Microsoft Windows [Version 6.1.7601]
Copyright (c) 2009 Microsoft Corporation.  All rights reserved.

C:\Users\ngraziano>
```

6. Issue the following command to change to the installation directory for Python™.

   ```
   CD
   C:\Users\[user_name]\AppData\Local\Programs\Python\Python3
   6
   ```

   See Figure A-9.

Figure A-9: Typical Command Prompt Window: Python™ Installation Directory



4411-0163_029

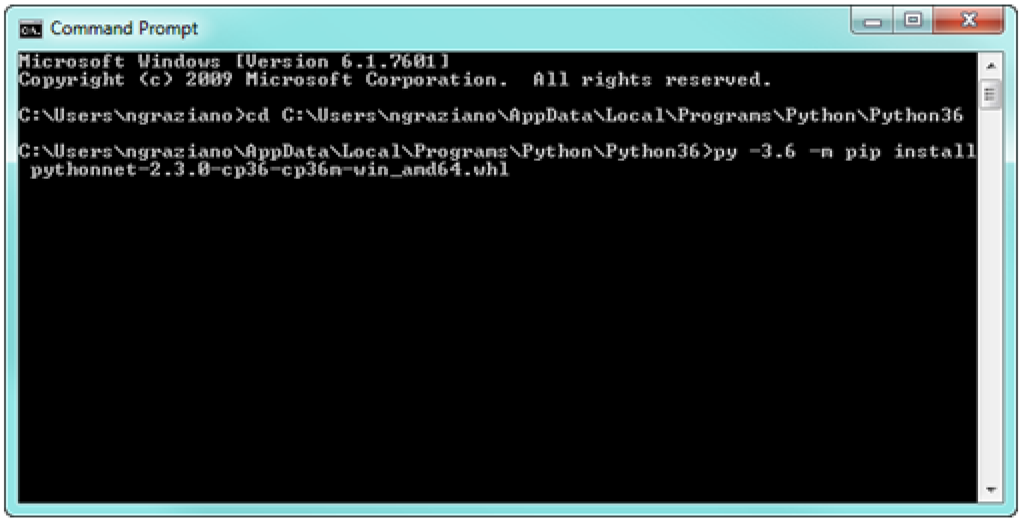
```
Command Prompt
Microsoft Windows [Version 6.1.7601]
Copyright (c) 2009 Microsoft Corporation.  All rights reserved.

C:\Users\ngraziano>cd C:\Users\ngraziano\AppData\Local\Programs\Python\Python36
```

7. Issue the command to instruct the **pip package manager**[a] to install the **Python for .Net** `*.whl` file:

   ```
   py -3.6 -m pip install pythonnet-2.3.0-cp36-cp36m-
   win_amd64.whl
   ```

   See Figure A-10.

Figure A-10: Command to Launch PIP Package Manager



4411-0163_030

```
Command Prompt
Microsoft Windows [Version 6.1.7601]
Copyright (c) 2009 Microsoft Corporation.  All rights reserved.

C:\Users\ngraziano>cd C:\Users\ngraziano\AppData\Local\Programs\Python\Python36

C:\Users\ngraziano\AppData\Local\Programs\Python\Python36>py -3.6 -m pip install
 pythonnet-2.3.0-cp36-cp36m-win_amd64.whl
```

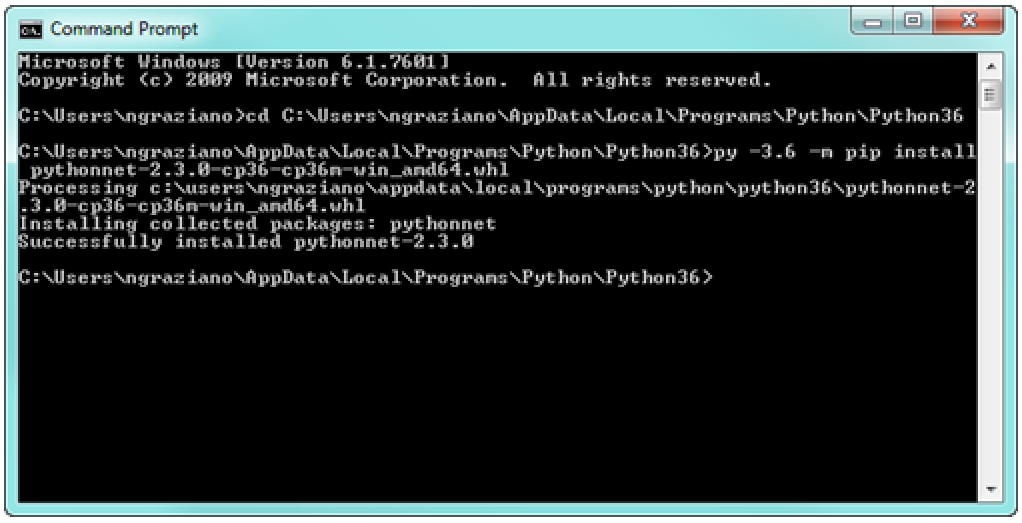
a. **pip** is a package management system used to install and manage software packages written in Python™.

8. Once installed, they system will return a **Successfully Installed pythonnet-x.y.z** message indicating that Python™ scripts may now be executed. See Figure A-11.

NOTE:

x.y.z corresponds to the specific version of PythonNet that has been installed.

Figure A-11: Typical Python for .Net Installation Successful Message



```
Microsoft Windows [Version 6.1.7601]
Copyright (c) 2009 Microsoft Corporation.  All rights reserved.

C:\Users\ngraziano>cd C:\Users\ngraziano\AppData\Local\Programs\Python\Python36

C:\Users\ngraziano\AppData\Local\Programs\Python\Python36>py -3.6 -m pip install
 pythonnet-2.3.0-cp36-cp36m-win_amd64.whl
Processing c:\users\ngraziano\appdata\local\programs\python\python36\pythonnet-2
.3.0-cp36-cp36m-win_amd64.whl
Installing collected packages: pythonnet
Successfully installed pythonnet-2.3.0

C:\Users\ngraziano\AppData\Local\Programs\Python\Python36>
```

This page is intentionally blank.

This page is intentionally blank.

www.princetoninstruments.com

TELEDYNE PRINCETON INSTRUMENTS
Everywhereyoulook™

Part of the Teledyne Imaging Group

info@princetoninstruments.com
USA +1 877-474-2286 | **France** +33 (1) 60 86 03 65 | **Germany** +49 (0) 89 660 7793 | **UK & Ireland** +44 (0) 1628 472 346
Singapore +65 6408 6240 | **China** +86 10 659 16460 | **Japan** +81 (3) 5639 2741