acontis technologies GmbH

SOFTWARE

EC-Master

Python Programming Interface

Version 3.2

Edition: February 22, 2024

Contents

1 Introduction

The Python Wrapper provides a Python interface to use EC-Master, EC-Simulator and RAS Client/Server.

1.1 Requirements

Python v3.7 and above

- Python Pause. Required for ticked timing with pause.until(…) to lower JobTask's drift, e.g. for Distributed Clocks

  ```
  $ pip install pause
  ```

- PyQt5 (v5.15.1). Only required to run the GUI demo

  ```
  $ pip install pyqt5
  ```

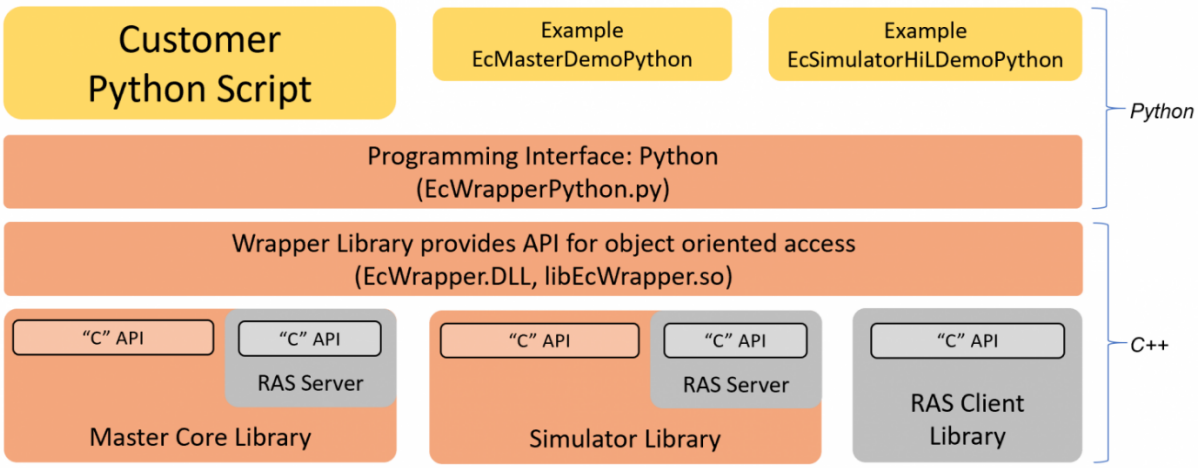
Windows (x86/x64)

- Microsoft Windows 7 and above
- Microsoft Visual C++ 2010 Runtime

Linux (x86/x64/ARM)

- Ubuntu 12.04 and above

1.2 Architecture



The architecture contains 4 basic layers:

Customer Python Script or our examples (EcMasterDemoPython, …)

- Demo application, written in Python

Programming Interface (EcWrapperPython)

- Provides an object oriented API written in Python

Wrapper Library (EcWrapper)

- Native wrapper library, which provides API for object oriented access

Native Libraries

- Master Core Library
- Simulator Library
- RAS Client Library

2 Programmers Guide

2.1 Sample Scripts

There are currently 2 scripts available:

EcMasterDemoPython.bat
Starts the console demo application

EcMasterDemoPythonInteractive.bat
Starts the interactive demo application

The scripts will start the demo application. The interactive demo application waits for user input where the user can enter the following commands:

```
# Write variable
demo.processImage.variables.Slave_1005__EL2008_.Channel_1.Output.set(1)

# Read variable
demo.processImage.variables.Slave_1005__EL2008_.Channel_1.Output.get()

# Print properties of variable
demo.processImage.variables.Slave_1005__EL2008_.Channel_1.Output.dmp()

# Stop the demo:
demo.stopDemo()
```

2.2 Sample Code

The Python demo application contains of 3 modules:

EcDemoApp.py:
Console demo application

EcDemoAppGui.py:
Gui demo application, based on Qt5

EcDemoAppInteractive.py:
Interactive demo application

2.3 Wrapper

2.3.1 Modules

The Python Wrapper contains of 4 modules:

EcWrapperPython.py

class CEcWrapperPython
EC-Wrapper base class

class CEcMasterPython
EC-Master

class CEcMasterMbxGatewayClientPython
Mailbox Gateway Client for EC-Master

class CEcMasterMbxGatewayServerPython
Mailbox Gateway Server for EC-Master

class CEcSimulatorPython
EC-Simulator

class CEcSimulatorRasServerPython
RAS Server for EC-Simulator

class CEcRasClientPython
RAS Client for EcMaster / EcSimulator

EcWrapperPythonTypes.py
Python types

EcWrapper.py
CPython interface (internal)

EcWrapperTypes.py
CPython types (internal)

2.3.2 Return code vs. exception handling

The most of all API functions returns a return code for error handling. This behaviour can be changed to throw an exception in error case by simply setting:

```
CEcWrapperPython.EnableExceptionHandling = True # default is False
```

2.3.3 API with “out” or “ref” parameters

The Python Wrapper API is based on C# code. C# supports `out` and `ref` keywords for parameters. This is not supported in Python and is solved by simply submitting `CEcWrapperPythonOutParam` or `CEcWrapperPythonRefParam` to those functions:

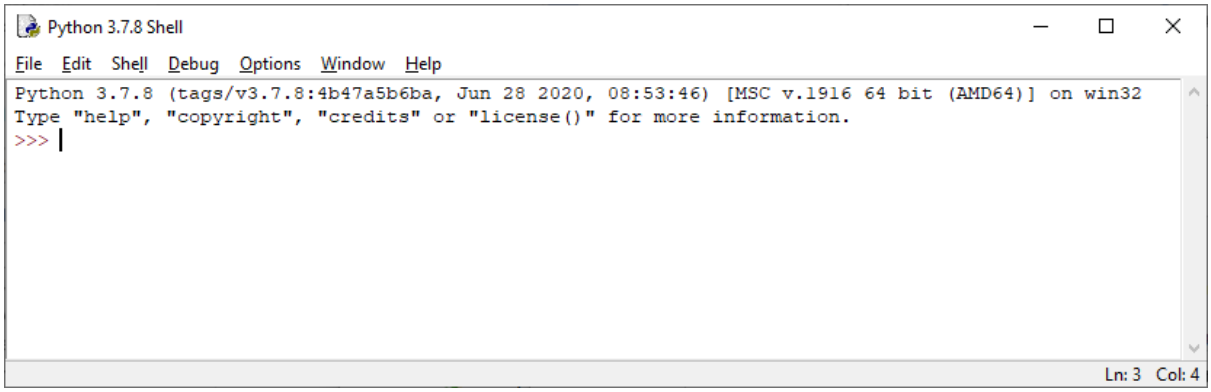
```
# This function has an "out" parameter "out_oSbStatus"
def GetScanBusStatus(self, out_oSbStatus):
    # ...
    return

# Create "out" parameter
out_oStatus = CEcWrapperPythonOutParam()
# Call function
pythonWrapper.GetScanBusStatus(out_oStatus)
# Get the "out" parameter value
oStatus = out_oStatus.value
# Now, the "oStatus" object can be used
print(oStatus.dwResultCode)
```

2.4 Supported IDEs

2.4.1 Python Shell IDLE

This is the default IDE.

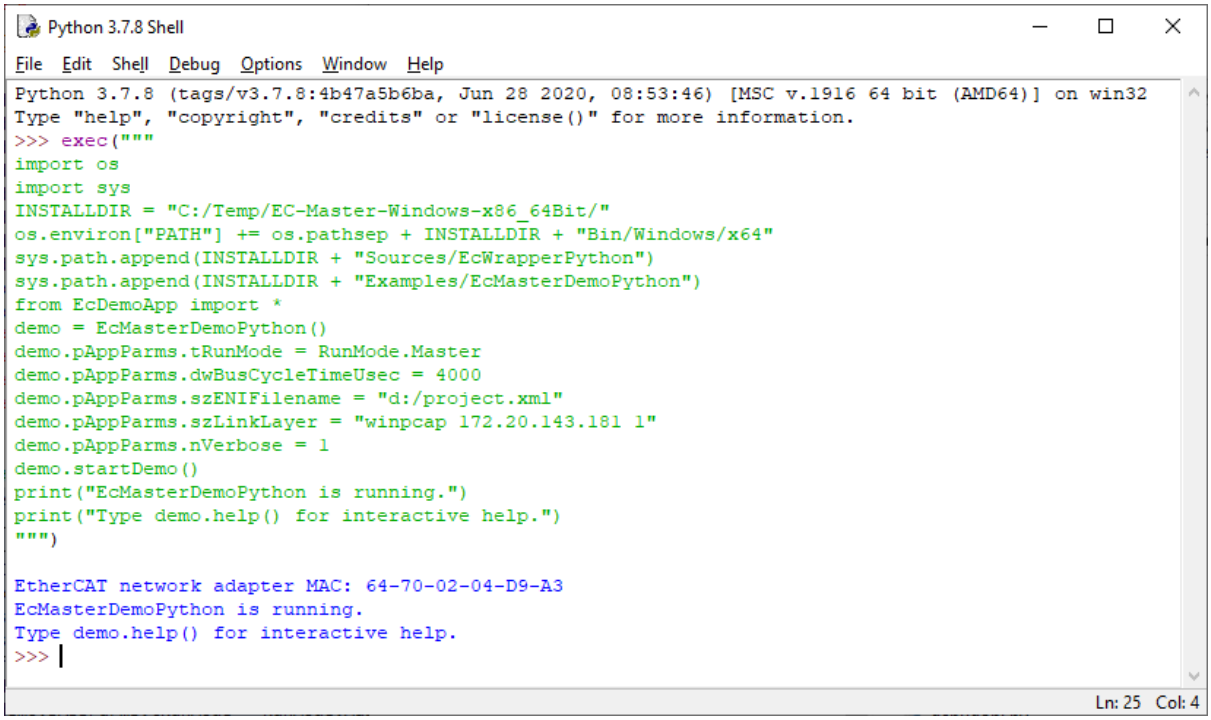
It can be started from Windows Start Menu or by calling `C:/Python/Lib/idlelib/idle.py`:



In this shell, the user can simply copy&paste the sample code from: `Examples/EcMasterDemoPython/EcDemoAppInteractive.py`

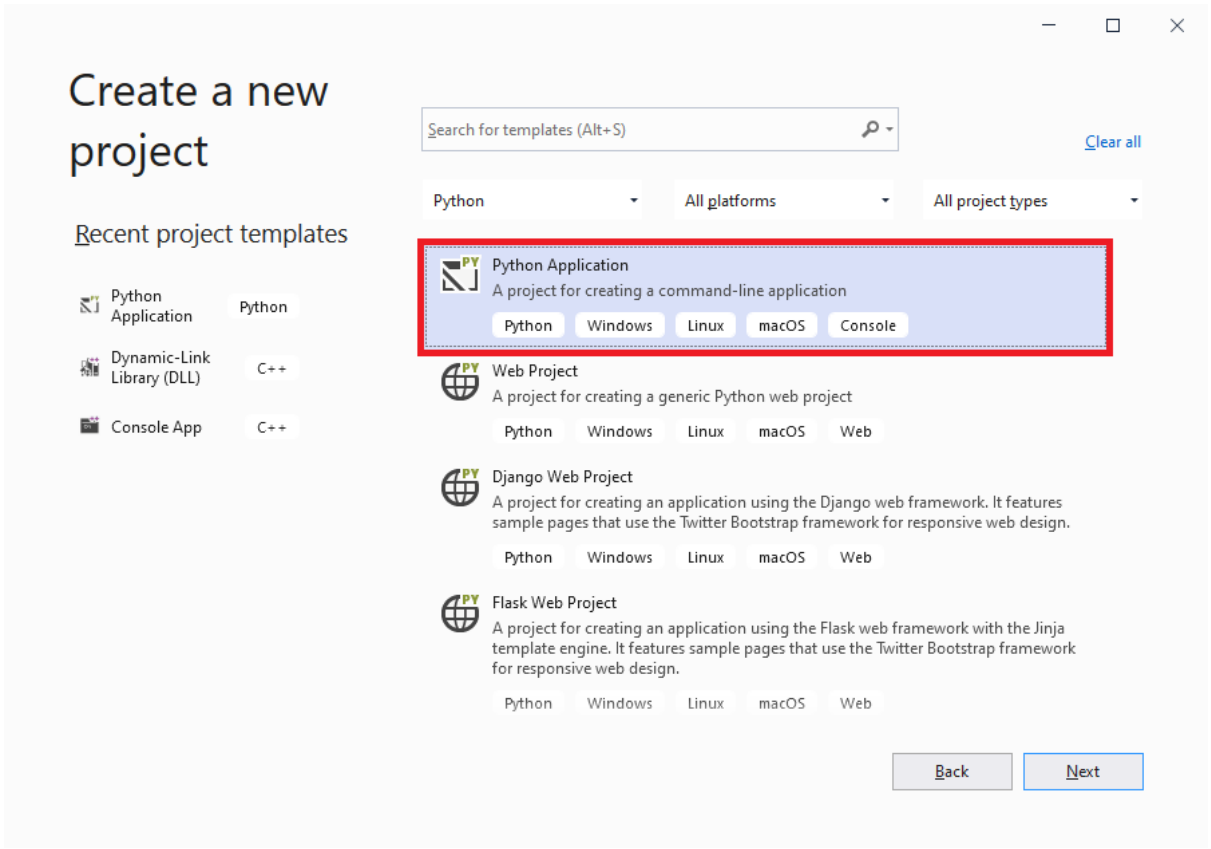
```
exec("""
import os
import sys
INSTALLDIR = "C:/Program
Files/acontis_technologies/EC-Master-Windows-x86_64Bit/"
os.environ["PATH"] += os.pathsep + INSTALLDIR + "Bin/Windows/x64"
sys.path.append(INSTALLDIR + "Sources/EcWrapperPython")
sys.path.append(INSTALLDIR + "Examples/EcMasterDemoPython")
from EcDemoApp import \*
demo = EcMasterDemoPython()
demo.pAppParms.tRunMode = RunMode.Master
demo.pAppParms.dwBusCycleTimeUsec = 4000
demo.pAppParms.szENIFilename = "ENI.xml"
demo.pAppParms.szLinkLayer = "winpcap 127.0.0.0 1"
demo.pAppParms.nVerbose = 3
demo.startDemo()
print("EcMasterDemoPython is running.")
print("Type demo.help() for interactive help.")
""")
```

… and the demo is running.



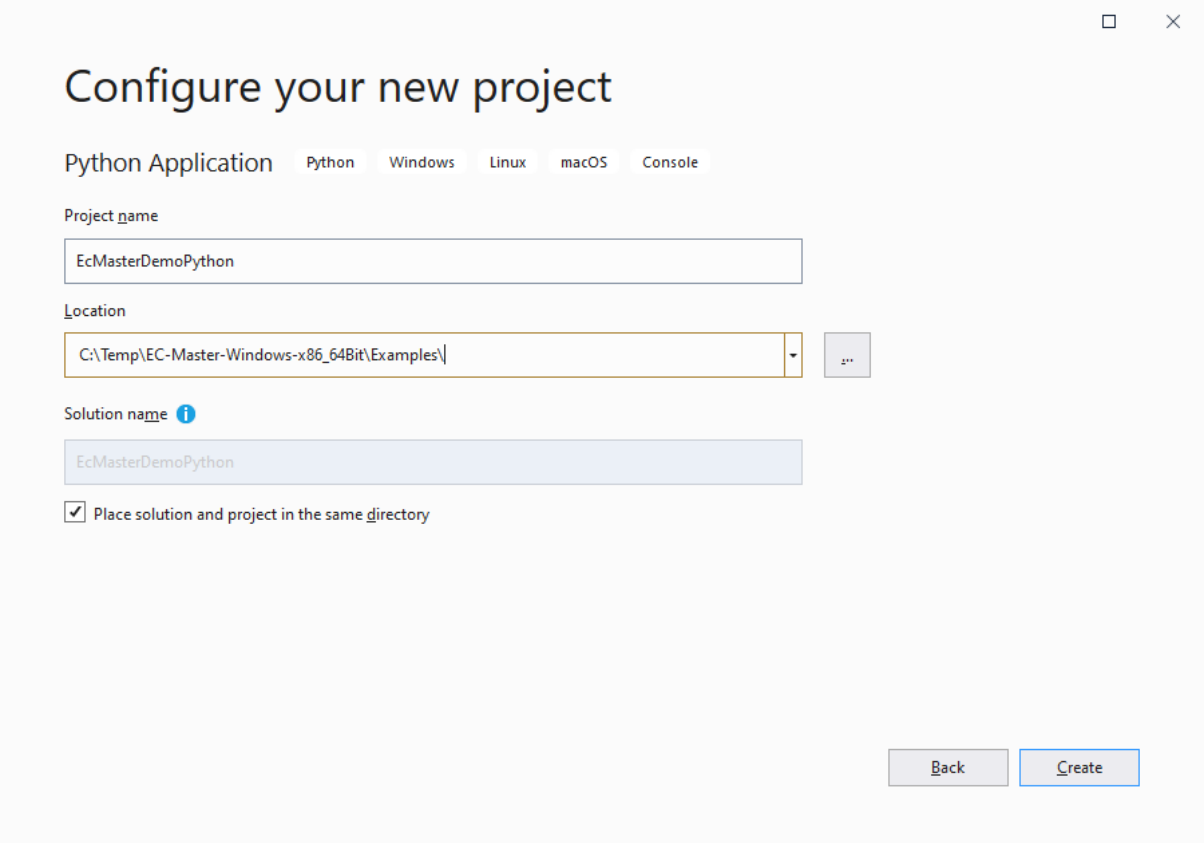
2.4.2 Visual Studio 2019

Create a new project:



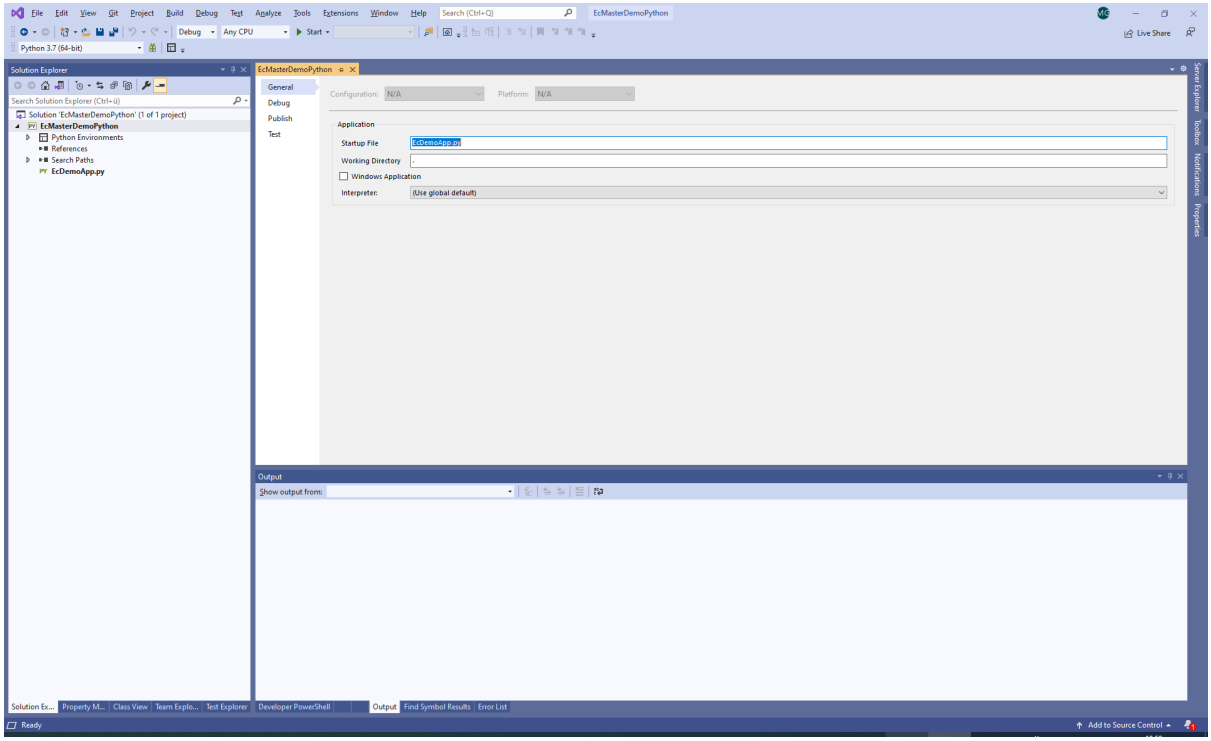
Configure the project:

- Replace the generated file `EcMasterDemoPython.py` with the existing `EcDemoApp.py`.



Configure project *General* settings:

- Startup File: `EcDemoApp.py`



Configure project *Debug* settings:

- Search Paths:

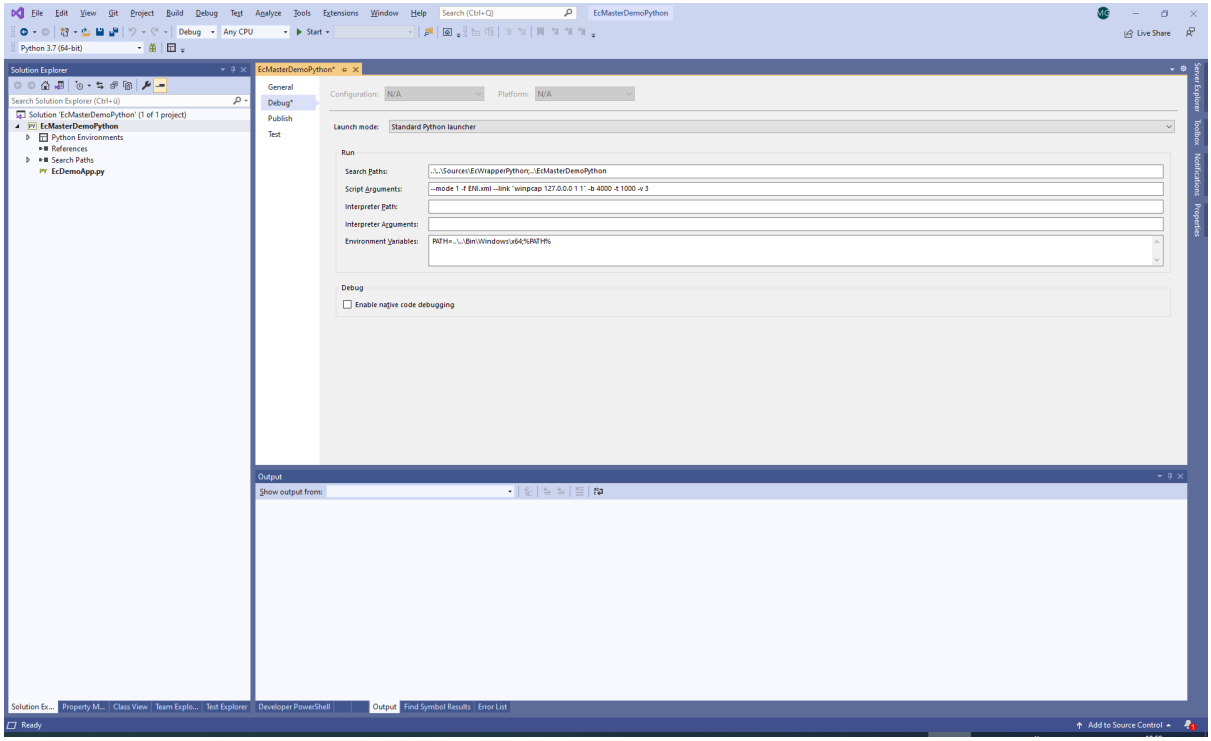
```
../../Sources/EcWrapperPython;../EcMasterDemoPython
```

- Script Arguments:

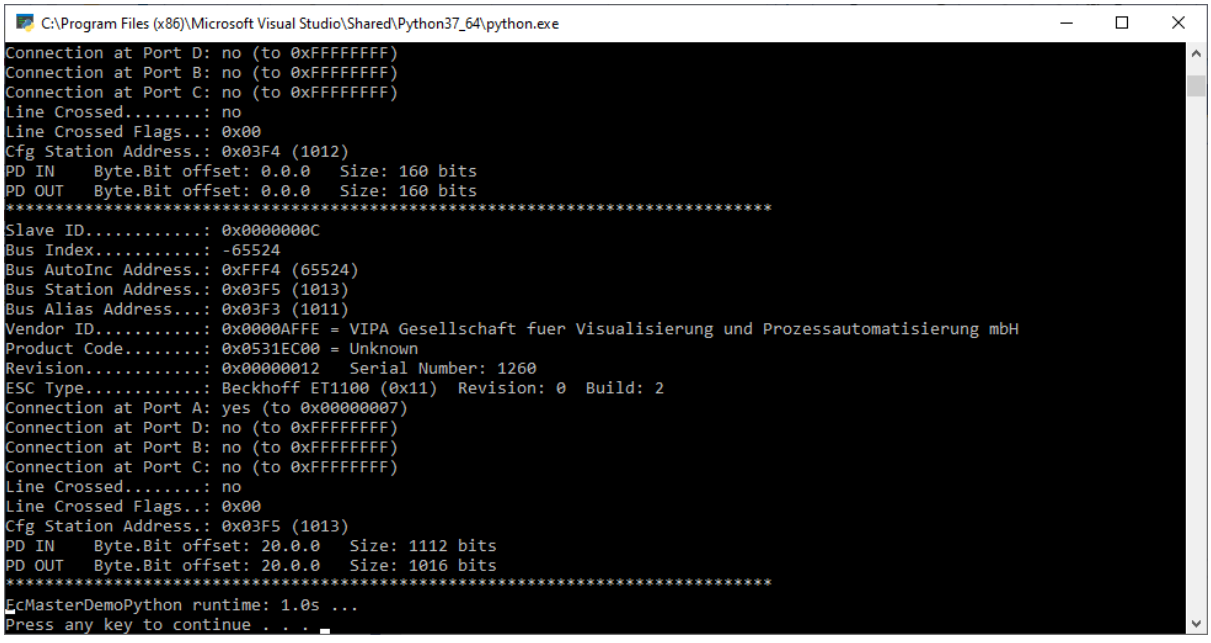
```
--mode 1 -f ENI.xml --link "winpcap 127.0.0.0 1 1" -b 4000 -t 1000 -v 3
```

- Environment Variables:

```
PATH=../../Bin/Windows/x64;%PATH%
```

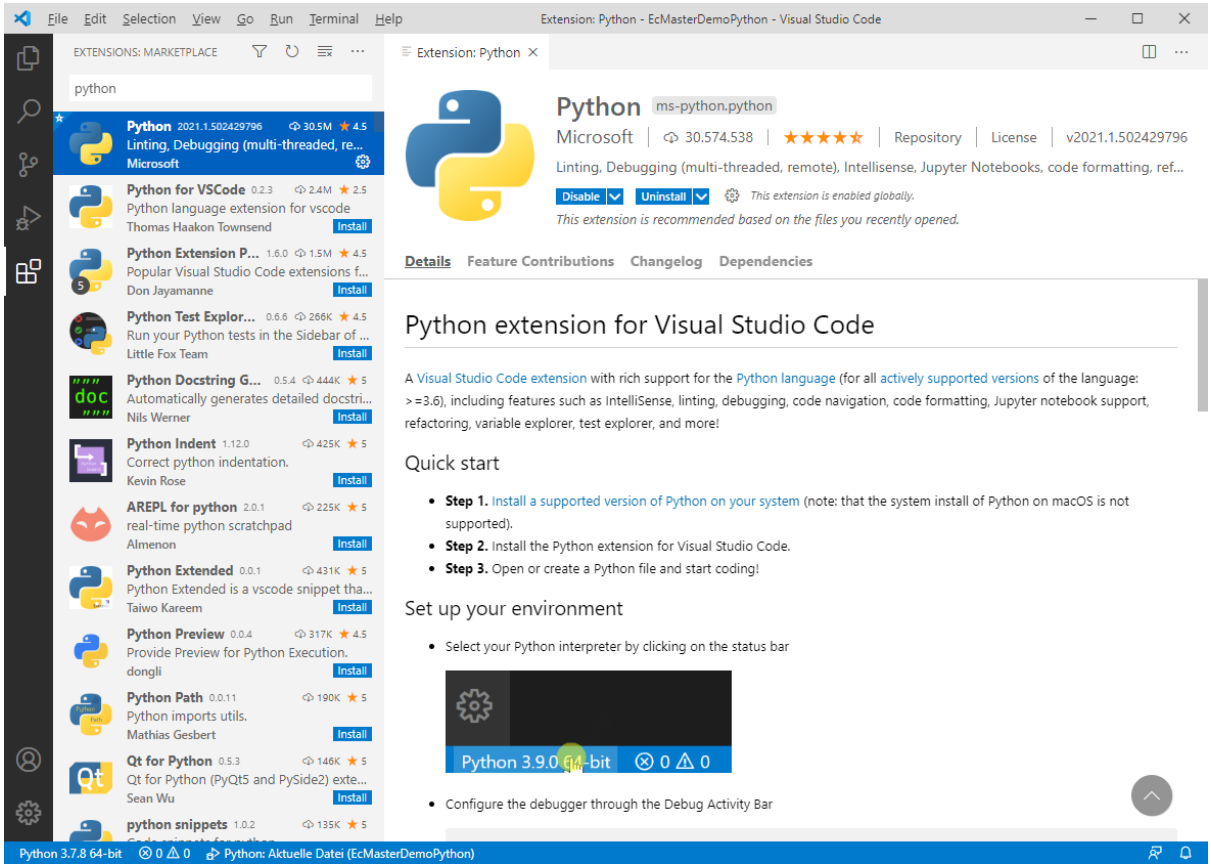


Press *Start* and the demo is running:



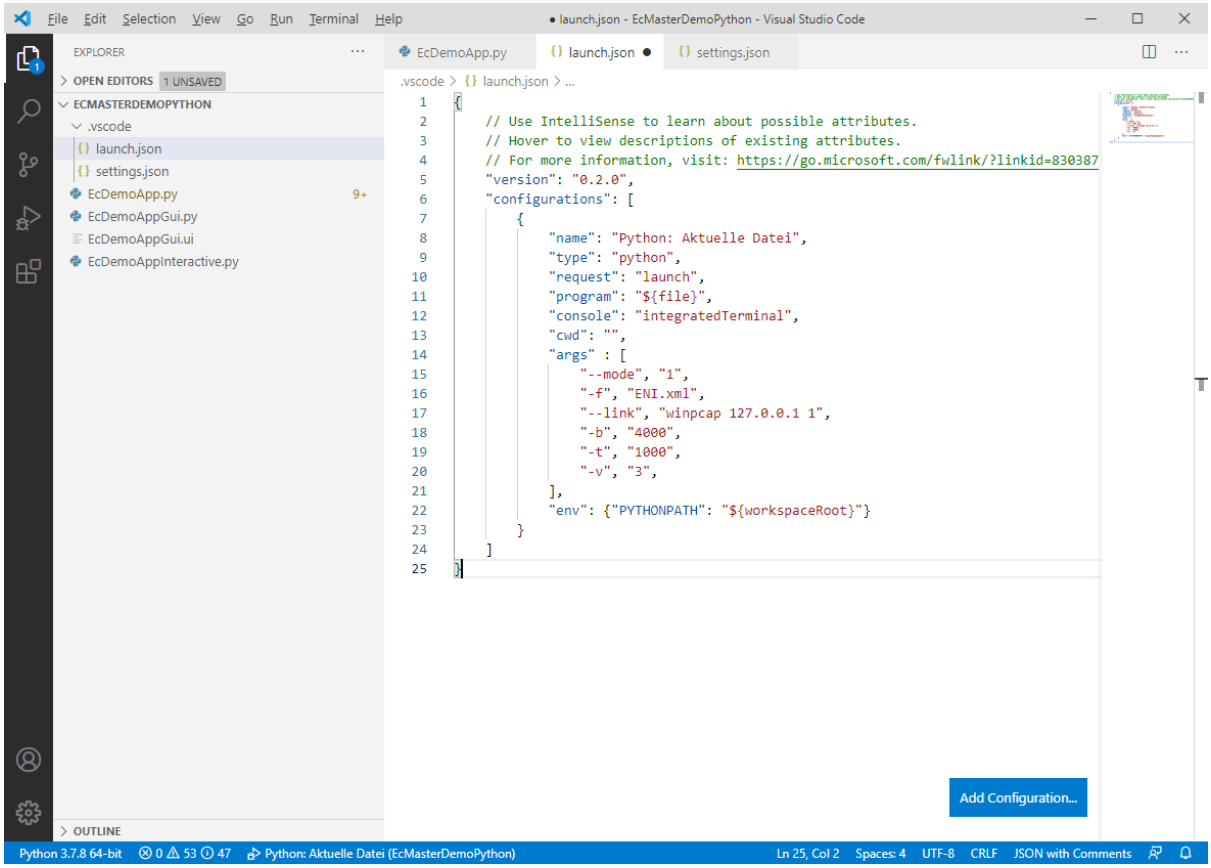
2.4.3 Visual Studio Code

Install python extension by open extension tab and enter *python*:



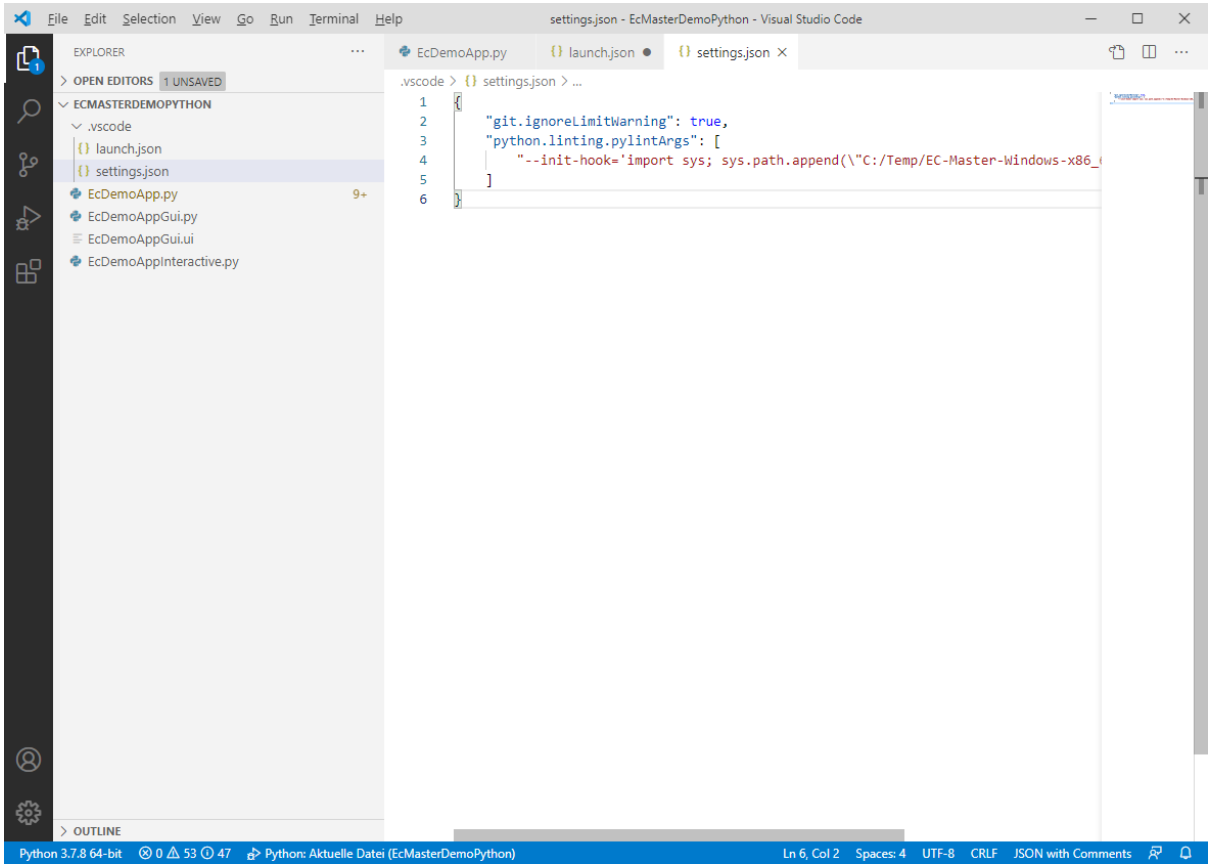
Open folder `Examples/EcMasterDemoPython` and configure the `launch.json`:

```
{
    "version": "0.2.0",
    "configurations": [
        {
            "name": "Python: Aktuelle Datei",
            "type": "python",
            "request": "launch",
            "program": "${file}",
            "console": "integratedTerminal",
            "cwd": "",
            "args" : [
                "--mode", "1",
                "-f", "ENI.xml",
                "--link", "winpcap 127.0.0.1 1",
                "-b", "4000",
                "-t", "1000",
                "-v", "3",
            ],
            "env": {"PYTHONPATH": "${workspaceRoot}"}
        }
    ]
}
```



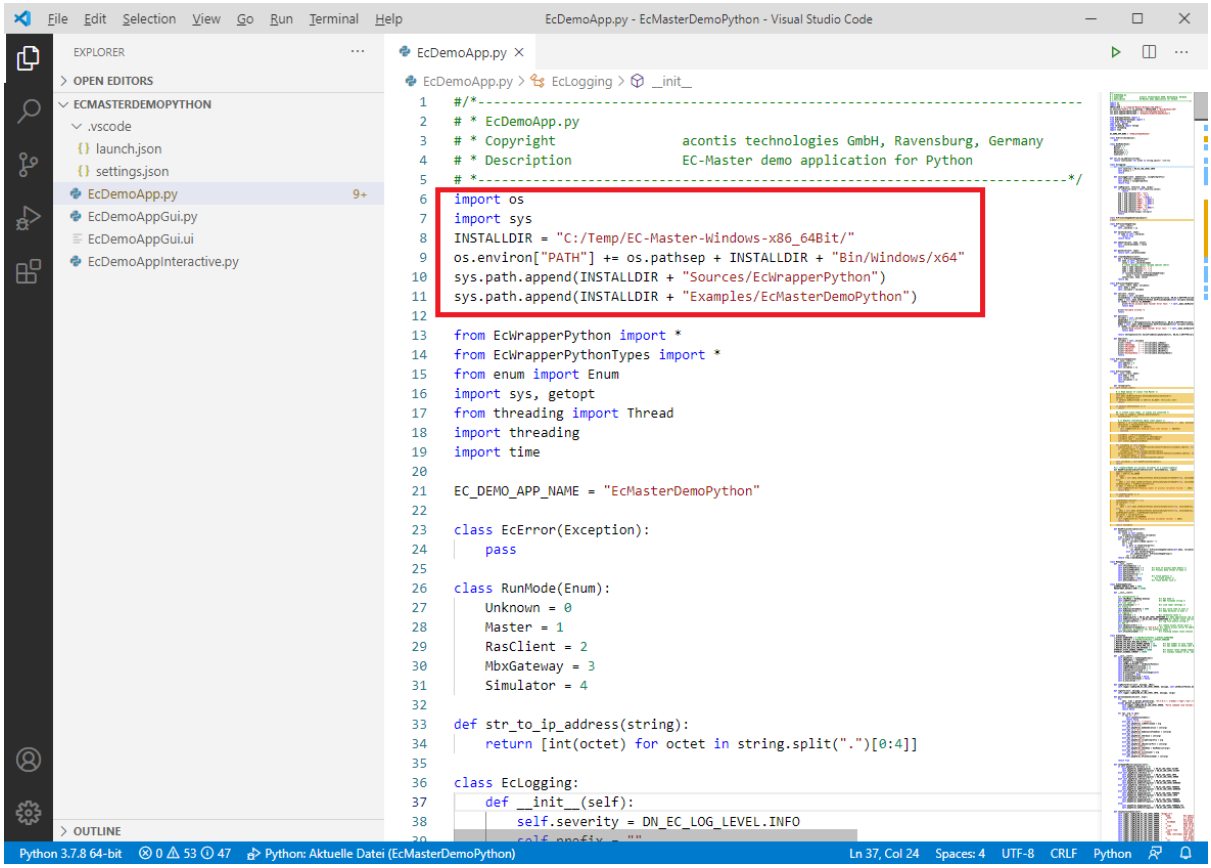
Configure linter in `settings.json`:

```
{
      "git.ignoreLimitWarning": true,
      "python.linting.pylintArgs": [
          "--init-hook='import sys; sys.path.append(\"C:/Temp/EC-Master-
↪Windows-x86_64Bit/Sources/EcWrapperPython\")'"
      ]
}
```

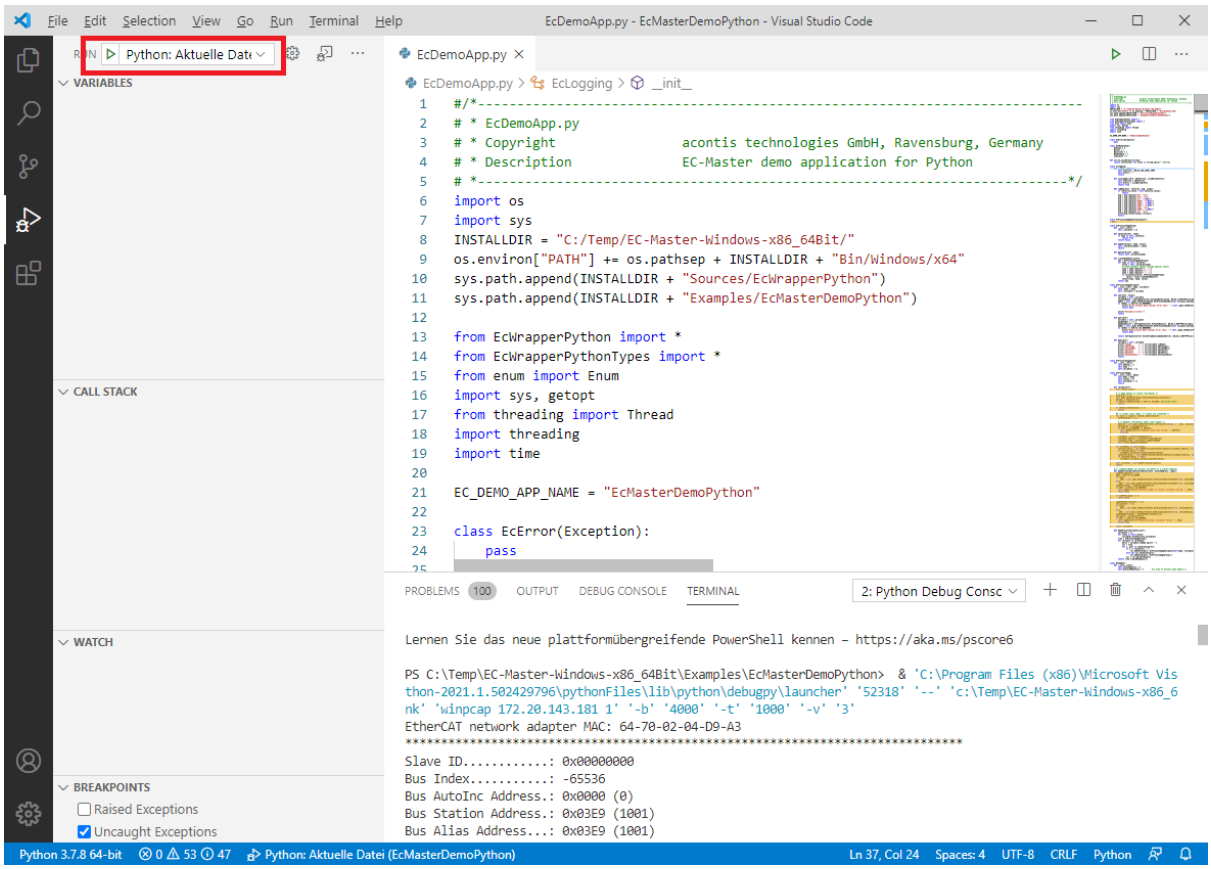


Open `EcDemoApp.py` and the following lines to set environment:

```
import os
import sys
INSTALLDIR = "C:/Temp/EC-Master-Windows-x86_64Bit/"
os.environ["PATH"] += os.pathsep + INSTALLDIR + "Bin/Windows/x64"
sys.path.append(INSTALLDIR + "Sources/EcWrapperPython")
sys.path.append(INSTALLDIR + "Examples/EcMasterDemoPython")
```



Start debugging and the demo output will be written into the terminal:



3 FAQ

PyQt5 cannot be installed on Ubuntu 14.04 x64, because it requires Python 3.5. How can I install it?

It can be installed by calling

```
$ sudo apt-get install python3-pyqt5
```

I installed Python and the demo crashes with strange errors. What can I do?

This might be a problem of mixing x86 with x64 binaries. Verify that if you have installed the Python runtime for x64 bit, please install also EC-Master for x64 bit.