PalmSens

PyPalmSens

Automating electrochemistry with Python

Webinar (20 min) + Q&A (20 min)

Our SDKs

PyPalmSens

Automate experiments

Script workflows, manage instruments, and analyze data.

.NET MAUI

Build native apps

Develop cross-platform desktop, iOS, and Android applications.

MethodSCRIPT

Program the potentiostat

Execute custom scripts directly on-device

dev.palmsens.com

What is PyPalmSens?

Python API

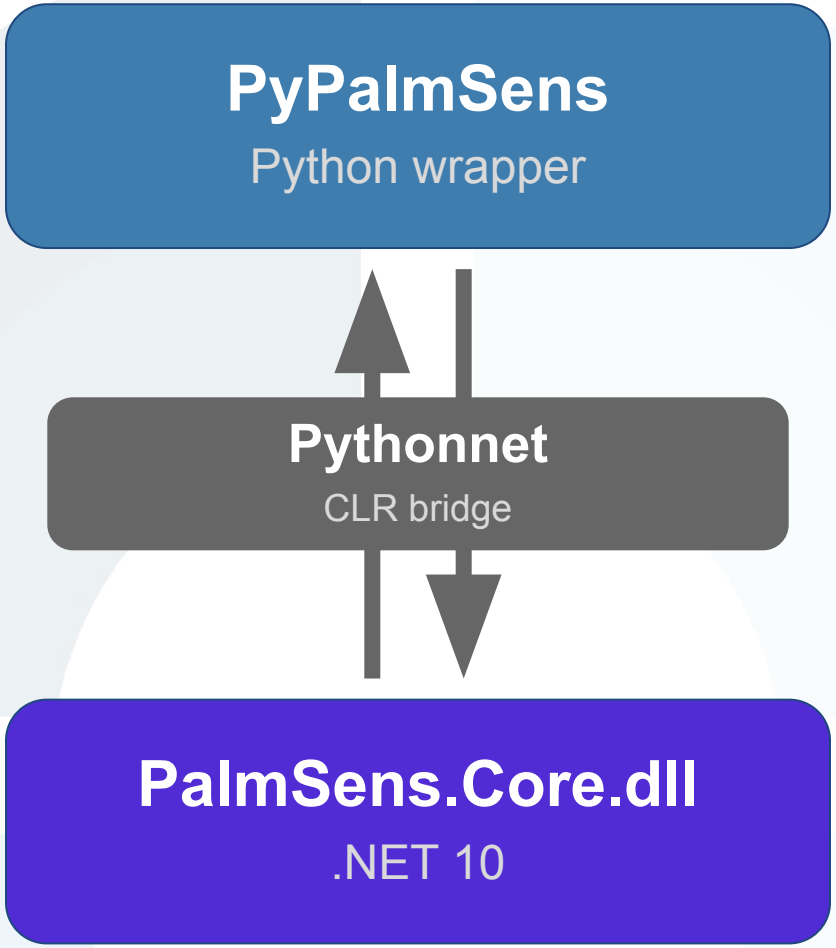
Works with all PalmSens potentiostats

Shared core

Same foundation as PSTrace

Open-source

Freely available on GitHub (BSD license)



What can you do with PyPalmSens?

Manage instruments

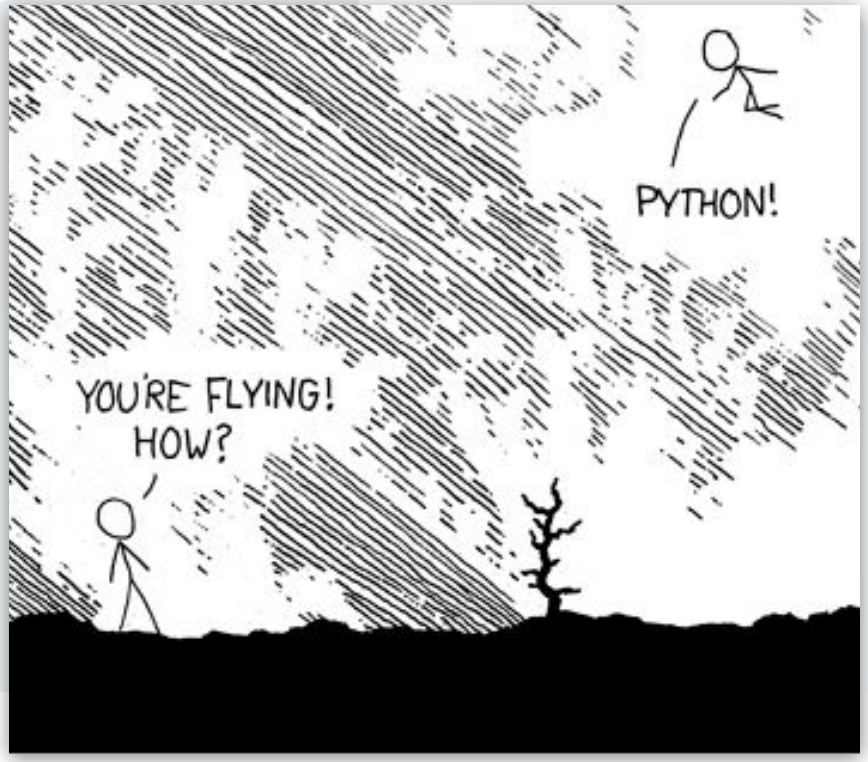
Discover, connect, and control single or multi-channel instruments

Automate experiments

Supports all methods and MethodSCRIPT scripts

Analyze data

Peak detection, impedance fitting, export to pandas/numpy, and read/write .pssession files



xkcd.com/353/

Built for developers

Pythonic API

- Ergonomic for interactive use
- Hides complex .NET classes
- Supports sync/async

IDE support

- Full type annotations
- Autocomplete / docstrings for LSPs

Best practices

- Automated releases to PyPI
- High test coverage (**~90%**).
- Online documentation

Links

github.com/palmsens/palmsens-sdk

dev.palmsens.com/python/latest

pypi.org/p/pypalmsens

Install from PyPI

```
pip install pypalmsens
```

Platform support

Windows, Linux, MacOS (arm/x64)

Prerequisites

- Python 3.10+
- .NET 10 runtime
- Drivers (see documentation)

dev.palmsens.com/python/latest

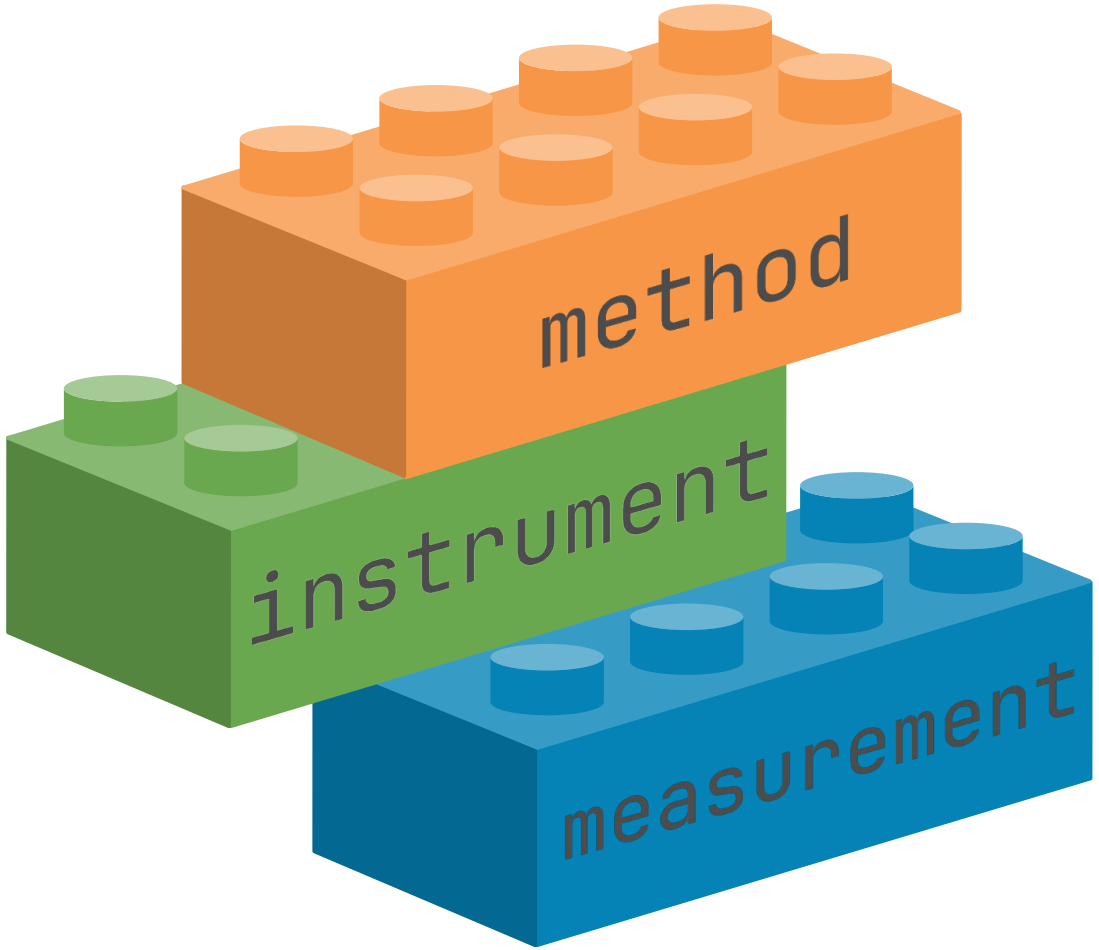
Electrochemistry in 3 lines

```python
import pypalmsens as ps

method = ps.ChronoAmperometry(
    interval_time=0.1,
    potential=1.0,
    run_time=5.0,
)
measurement = ps.measure(method, callback=print)
```

`ps.measure` finds your instrument, runs the measurement, and returns the result

The 3 building blocks



Method

Define experiment parameters

Instrument (manager)

Connect and send instructions

Measurement

Analyze and process data

Example 1: Instrument management

```python
import pypalmsens as ps

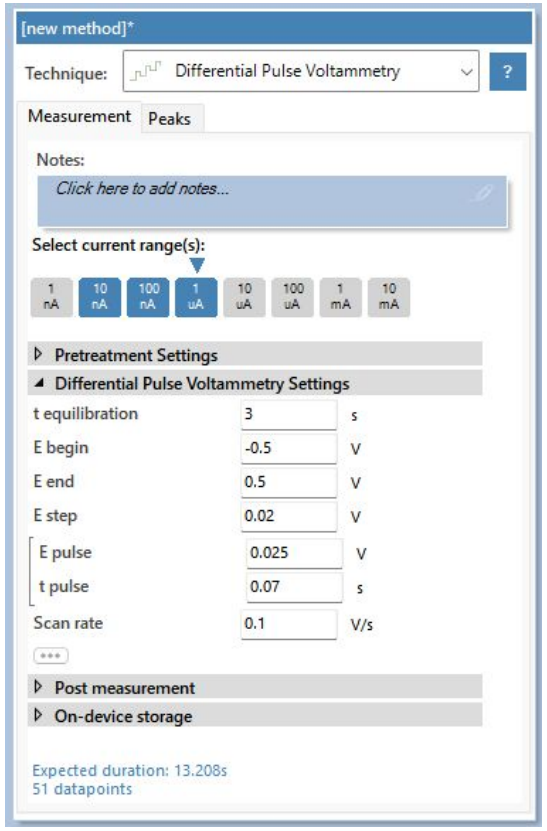
instrument, *_ = ps.discover()

with ps.connect(instrument) as manager:
    manager.set_cell(True)
    manager.set_potential(1)  # V
    manager.set_current_range('1mA')

    current = manager.read_current()
    print(f'current = {current} µA')

    manager.set_cell(False)
```

Example 2: DPV with peak finding




```python
import pypalmsens as ps

method = ps.DifferentialPulseVoltammetry(
    equilibration_time=3,
    begin_potential=-0.5,
    end_potential=0.5,
    step_potential=0.02,
    pulse_potential=0.025,
    pulse_time=0.07,
    scan_rate=0.1
)
measurement = ps.measure(method, callback=print)

curve = measurement.curves[0]
peaks = curve.find_peaks()
curve.plot().show()
```

Example 3: Inspect current ranges

```python
import pypalmsens as ps

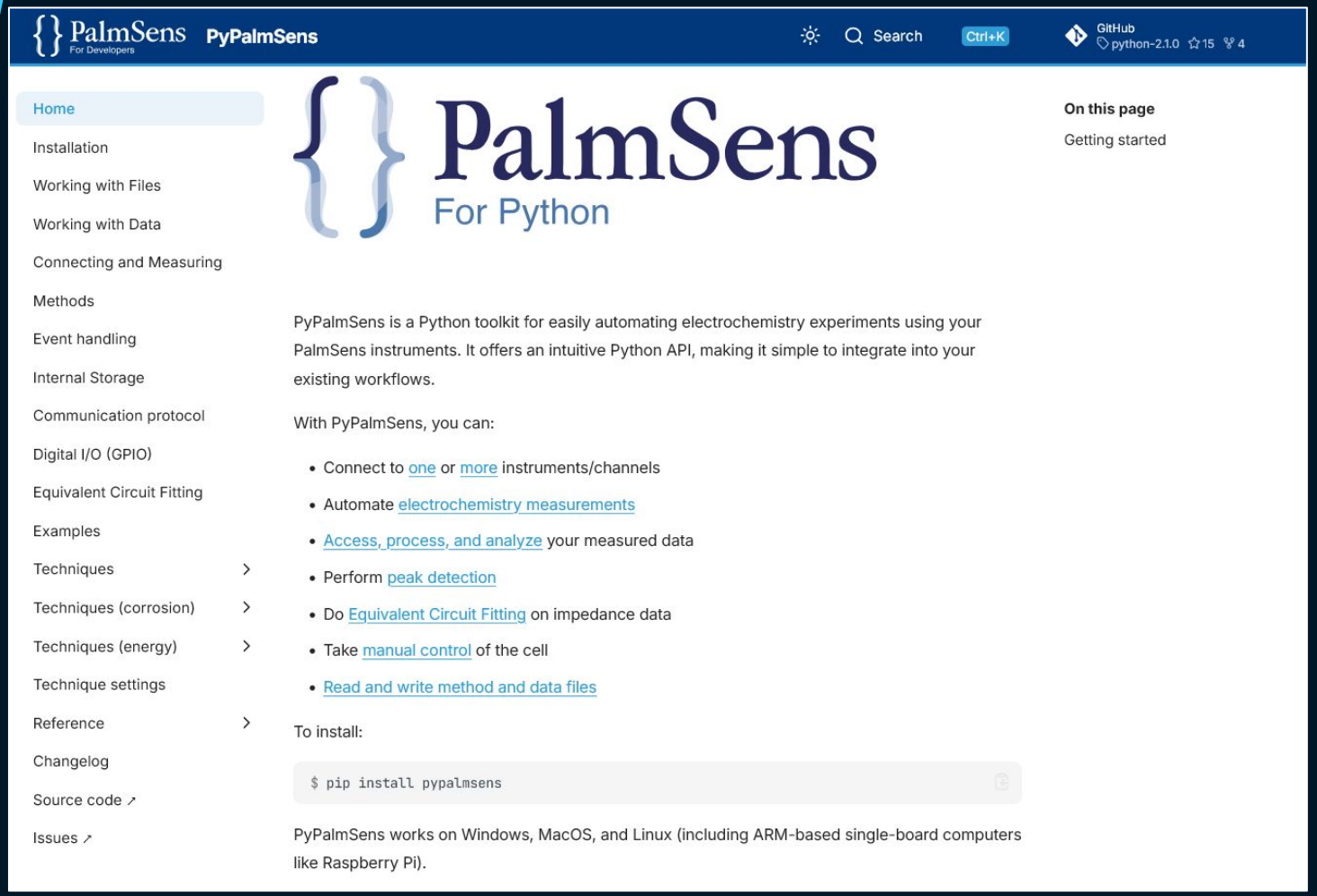
cv = ps.CyclicVoltammetry()
cv.current_range.min = '1nA'
cv.current_range.min = '1MA' # validation error
cv.current_range.max = '1mA'
cv.current_range.start = '1mA'

measurement = ps.measure(cv, callback=print)

df = measurement.dataset.to_dataframe()
print(df.head())

ps.save_measurement('cv.pssession', measurement)
```

More information



dev.palmsens.com/python/latest

Questions, issues, feedback?

support@palmsens.com

Python for PalmSens Webinar

- **Session I** - Getting started (Oct 7, 2026)
- **Session II** - Impedance Spectroscopy (Nov 4, 2026)
- **Session III** - Advanced topics: Become a pro (Dec 2, 2026)

Load and save data

```
import pypalmsens as ps

method = ps.load_method_file('cv.psmethod')

measurement = ps.measure(method, callback=print)

ps.save_measurement('cv_data.pssession', measurement)

measurement2 = ps.load_measurement('cv_data.pssession)
```

Run MethodSCRIPT




```
import pypalmsens as pypalmsens

script = """..."""

method = ps.methodscript(
    script=script
)

measurement = ps.measure(method)
```

```
script = """e
set_acquisition_frac_autoadjust 50
var i
var e
set_pgstat_chan 0
set_pgstat_mode 3
set_max_bandwidth 58505m
set_range ba 2100u
set_autoranging ba 2100n 2100u
set_e 500m
cell_on
meas_loop_ca e i 500m 100m 10100m
  pck_start
    pck_add e
    pck_add i
  pck_end
endloop
on_finished:
  cell_off

"""
```