dev week

14th to 17th of April 2025

A silx tour

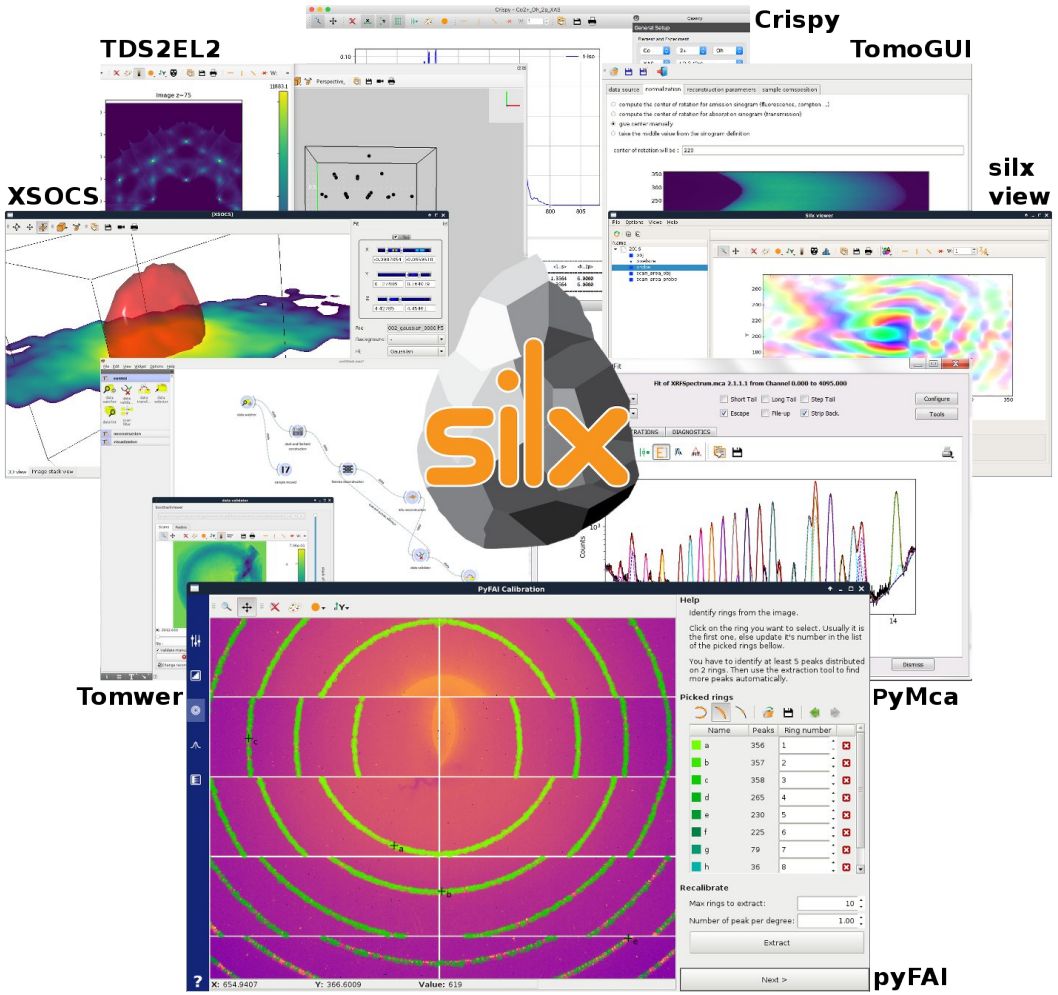
The purpose of the *silx* project is to provide a collection of Python packages to support the development of data assessment, reduction and analysis applications at synchrotron radiation facilities.

silx aims to provide reading/writing tools for different file formats, data reduction routines and a set of Qt widgets to browse and visualize data.

- Documentation: https://www.silx.org/doc/silx/latest/
- Repository: https://github.com/silx-kit/silx
- Contribution guidelines: https://www.silx.org/doc/silx/latest/contributing.html
- Available from pypi, conda-forge, debian, ubuntu

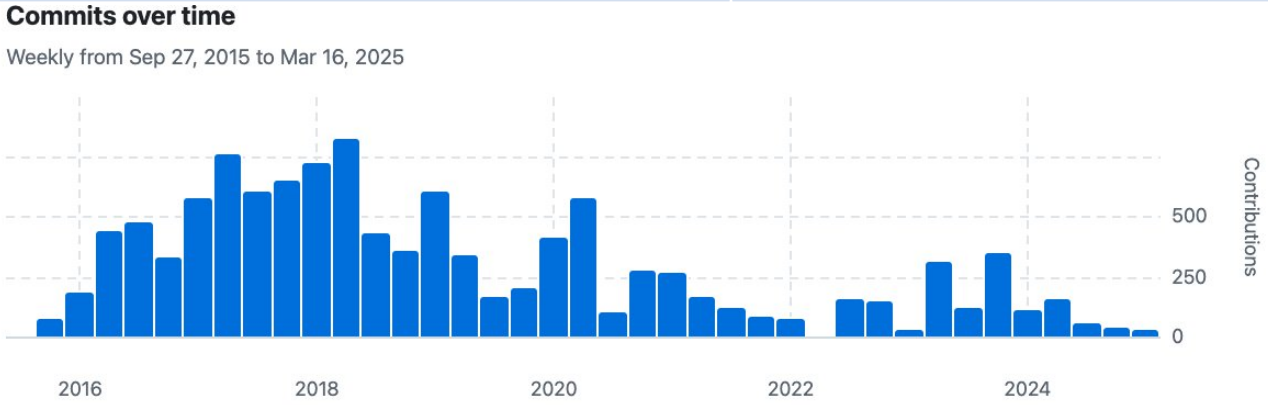
Where silx is used?

- ESRF:
  - General tools: flint, silx view
  - Technique: pyfai, pymca, tomwer...
  - Beamline-specific: xsocs, txs,...
- Other synchrotrons
  - silx view
  - Apps, e.g. ParSeq (Max-IV)
- ???



silx code base

Language	Files	Line of Code
Python	624	115 942
reStructuredText	174	7 174
Cython	23	5 170
OpenCL	39	4 703
C	32	4 238
C++	3	770
Total	895	137 997



contributors:
41 overall
14 over last year

silx packages

- **silx**:
  - **io**: File access: hdf5 utils, h5py-like access, specfile, nxdata
  - **image**, **math**: Data processing functions
  - **opencl**, **resources.opencl**: OpenCL-based processing functions
  - **utils**: Miscellaneous helper functions
  - **gui**, **resources.gui**: Qt widgets
  - **app**: Applications: silx view compare convert
  - **sx**: High-level plot functions: plot, imshow
  - **test**
  - **third_party**: Deprecated

silx.gui package: Qt-based widgets

- **Modules: colors** (colormaps), **console** (IPython), **constants**, **icons**, **printer**
- **Sub-packages**:
  - **qt**: Wrapper of PySide6, PyQt6 and PyQt5 API
  - **plot**: Widgets to display curves, histogram, images and scatter plots
  - **plot3d**: Widgets to display data in 3D: scatter, isosurface,...
  - **dialog**: Specific dialogs: colormap, select a hdf5 dataset...
  - **widgets**: General purpose widgets
  - **fit**: Widgets to configure/run fitting
  - **data**: Widgets to plot data arrays as plots or table
  - **hdf5**: Widget to browse HDF5-like structured data
  - **utils**: Miscellaneous function useful for GUI programming
  - **test**
  - **_glutils**: OpenGL helpers for **plot** and **plot3d**

silx project

- **.github**/, **ci/**: Continuous integration config & helpers
- **doc/**: Sphinx documentation
- **examples/**: Sample code
- **package**/: Debian packaging and Windows installer
- qtdesigner_plugins/: **Deprecated**
- **src/silx**/: Source
- **tools**/: Dev tools

Activity

Package	Languages	Code	# commits/last year	# Contributors
io	Python, C, Cython	12554 - 10%	36	20
image	Cython, Python, C	2564 - 2%	2	8
math	Python, Cython, C, C++	13585 - 10%	16	13
opencl	Python, OpenCL	9760 - 10%	15	10
utils	Python	2735 - 2%	7	9
gui	Python, shaders	73126 - 53%	104	34
app	Python	3611 - 3%	34	13
sx	Python	438 - 0.3%	0	5

legend:

- green: active
- yellow: maintenance
- orange: staled

Who to ask for help?

Package	who to ask for help ?
io	Wout, Henri, Valentin, Thomas
image	Jerome K, Thomas, Valentin, Pierre
math	Thomas, Pierre, Jerome K, Henri
opencl	Jerome K, Pierre
utils	Valentin, Thomas, Wout
gui	Thomas, Henri, Valentin
app	Valentin, Thomas, Henri
sx	Thomas

legend:

- green: active
- yellow: maintenance
- orange: staled

Dev tools

- Definition of some ‘exotic’ tools
  - bootstrap.py
    - useful if you want to develop some Cython code. Else use `pip install -e .`
  - run_tests.py
  - tools/
    - build_man_page.py
    - create_h5_sample.py
      - Generate a .h5 file containing most of special structures supported by the format (all_types.h5)
    - export_svg.sh
      - Export from svg to png (keeping the same name)
    - format_GH_release_notes.py
      - Fix the format of GitHub-generated releases notes
    - optimize_svg.sh
      - Script to ease call to `scour`
    - update_icons_rst.py

Dev how-to

- Python-only development, using standard ways:
  - `pip install -e .`
  - `pytest src/silx/test/test_version.py`
  - `sphinx-build -j 4 doc/source html`
- Cython/C/C++ development:
  - Build&run:
    - `python bootstrap.py` -> IPython
    - `python bootstrap.py silx view`
  - Build&test:
    - `python run_tests.py [-help]`
    - `python run_tests.py src/silx/test/test_version.py`

“Guidelines”

Contributions

- License is MIT © ESRF
  - Contributions are expected to conform to this license
- New features & **maintainability**:
  - It should be maintainable by others
  - We expect you to also help maintaining it over time ;)
- New dependencies should:
  - show no mid-term maintainability risks
  - be packaged in pypi, conda-forge, debian, ubuntu
  - run on Linux, macOS & Windows

Coding convention

- Coding convention:
  - All the project but silx.gui follows the PEP_8 convention
  - silx.gui follows PyQt/PySide convention: CamelCase
    - No instance attributes as public API: Use getter and setter methods:
      - `def getValue() -> int:`
      - `def isValue() -> bool:`
      - `def setValue(self, value: int):`
    - Prefix signal names with `sig`, e.g., `sigValueChanged` (except to mimic Qt API)
    - One widget class in the public API = One Python module with the same name
  - Keep as much API as possible `_private`
- Code formatting:
  - black (strongly encouraged, not enforced)
- Typing annotations:
  - Just really starting, strongly encouraged

API stability

- We take backward compatibility seriously:
  - No API/behavior break without deprecation warnings
- Deprecated APIs are kept for at least 1 major version (at very least 1 year):
  - v3 will remove APIs that were deprecated in v1, so before 2023
- How to deprecate:
  - Advertise deprecation with silx.utils.deprecation:
    - `from silx.utils.deprecation import deprecated, deprecated_warning`
  - Remove deprecated API from the documentation
  - Update usage throughout the library (except tests).
  - An option can be to make `Object2` and deprecate `Object`
- Prepare for the future:
  - Keep as much of the API `_private`

Tests, cross-platform

- silx runs:
  - on Linux, macOS, Windows; on x86_86, arm64, ppc64le
  - with Python **3.10** to 3.13
  - with PySide6, PyQt6, PyQt5
  - => 72 combinations (neglecting Qt minor versions)
- We need automated tests!
- Tests are placed in `test` sub-packages in the sub-package they belong to.

Documentation

- https://www.silx.org/doc/silx/latest/
- Lives in the `doc/` folder, is based on sphinx & **rst**
- API documentation is NOT generated automatically
- Documentation needs improvements:
  - Some API is missing
  - Simple how-to, examples are missing
  - ...
- Remind that not all silx users clone the repo and use an IDE:
  - “what is not documented does not exists”
  - Use meaningful names
  - Write docstrings

Pull Request good practice

- Title is used in release notes !!!
- Provide information/description, related to relevant issue
  - keep in mind that reviewers have no clue of what you are doing.
- **Be kind to reviewers**: focus your PR: it is simpler to review many small consistent PR than a mix of changes
- If you are stuck don’t hesitate to ask for help

Recent switch to PySide6 by default

- You are encouraged to develop&test the GUI part with PySide6:
  - `pip install pyside6`
  - `export QT_API=PySide6`
- Several modifications have been push recently!!!
  - Make sure you start your PR from the latest commit on `main`.

Get started

- Identify something you are interested on:
  - For the silx-dev week there is label that you can us to estimate how the difficult is the issue to be solved: https://github.com/orgs/silx-kit/projects/6
  - Assign it to yourself
- Try to achieve it:
  - Contributing guide: https://www.silx.org/doc/silx/latest/contributing.html
  - Do a PR to https://github.com/silx-kit/silx
  - Tag your PR when ready by the ‘ready to merge’ label
- Wow! I can do what I want, what next?
  - Start again, Make suggestions, Contribute with a demo/recipe
- Ouch! I found a bug:
  - Check if no issue is already opened for it and open an issue: https://github.com/silx-kit/silx/issues
- I want a coffee -> help yourself