The PARI Jupyter kernel

Jeroen Demeyer

Universiteit Gent
Université Paris-Sud (Orsay)

History of IPython + Jupyter

- Python comes with a very simple command line interface.
- IPython provides a much better command line for Python: history, TAB-completion, debugging...
    - A separate project, not part of Python.
- Sage uses IPython for its command-line interface.
- Much more recently, IPython developed a web-based graphical interface.
    - Inspired by (but independent from) SageNB, a web interface for running Sage developed as part of Sage.
- Last summer, IPython split off the language-independent parts in a new project called Jupyter.

The PARI Jupyter kernel

- A *Jupyter kernel* is what actually executes the commands of a notebook user. It communicates with the notebook server using ØMQ.
- There exist currently ≈ 50 different kernels for Jupyter.
- One can easily get started writing Jupyter kernels using a *wrapper kernel*, which reuses IPython's implementation: it suffices to define just a few methods to have a complete working kernel.
- The whole PARI Jupyter kernel is a few hundred lines of code.

Getting it

- From Sage: `sage -i pari_jupyter`
- `https://github.com/jdemeyer/pari_jupyter`
- Needs git version of Cython and PARI.

Features

- Support all language features of GP.
- TAB-completion like in GP.
  - Not part of PARI library ⇒ requires experimental changes to the PARI sources.
- History and timer.
- Short help using shift-TAB.

TODO:

- Syntax highlighting.
- Long help in the browser.
- Break loop / debugger.
- Plotting.
  - Not part of PARI library.

Why is this so easy?

Cython:

- A language which is the “union” of Python and C: It makes it easy to write Python code calling a C library.
  - Like GP2C, Cython generates C code.

PARI:

- Parsing and executing GP code is trivial using calls to the PARI library.
  - It did require a small patch to the PARI sources to read multi-line input from a `char*` instead of a file.

Jupyter:

- Once you manage to find the right documentation, writing wrapper kernels is easy. There is a toy implementation in the `jupyter_client` documentation: an *echo kernel* which just echoes all input.