IPython Cheat Sheet

Full Documentation: https://ipython.readthedocs.io

Object Introspection and Documentation

`obj?` *or* `?obj`	Show a brief description of `obj`, such as its signature and “doc string” (documentation).
`obj??` *or* `??obj`	Show the source code of `obj`.
`obj.<TAB>`	List the attributes of `obj`.
`obj(<TAB>`	List allowed keyword arguments, and also file and directory names in the current directory.

Command History and Output Cache

`<UP ARROW>` *and* `<DOWN ARROW>`	Scroll through recent commands.
`text``<UP ARROW>`	...commands that began with *`text`*.
`<CTRL+R>`*`text`*	...commands that contained *`text`*.
`_` *or* `__` *or* `___`	First, second, or third most-recent outputs
`_5`	Output from prompt number 5
`_i5`	Input string from prompt number 5
`%history` *or* `%hist`	Show complete history of current session.
`%history ~2/`	...complete history of two sessions ago.
`%history ~2/5-10`	...lines 5-10 from history of two sessions ago.
`%history -g` *`search_term`*	Search all sessions’ history for *`search_term`*.

Python Variables and System Environment Variables

`%who` `%who ophyd.Device  # filter by type` `%who function str  # multiple types`	Show names of all “interactive” variables. Optionally, show only variables of certain type(s). (Built-in Python and IPython variables are omitted.)
`%whos`	Show a short description for each variable.
`%env` `%env HOME` `%env http_proxy http://proxy:8888`	List, get, or set environment variables (such as HOME, PATH, http_proxy…).

Startup Scripts in IPython profiles

Starting IPython with the option `ipython --profile=foo` executes any scripts in the directory `~/.ipython/profile_foo/startup/` with filenames ending in .py or .ipy. If no `--profile` option is specified, the default profile is `profile_default`.

IPython Cheat Sheet

Full Documentation: https://ipython.readthedocs.io

Editing and Running Scripts

`%edit`	Create a new temporary file, edit it, and execute the code in the local namespace.
`%edit` *`filename`*	Edit a Python script. Execute it upon exiting.
`%edit -x` *`filename`*	Edit a file but do not execute it.
`%edit obj`	Edit the file where `obj` or its class was defined, as applicable.
`%edit 0/`	Open command history of current session as a text file.
`%run` *`filename`*	Run a Python script in fresh namespace. Then export any variables it defines back into the local namespace.
`%run -i` *`filename`*	Run a Python script in the local namesapce.

System Shell Commands

`!cp a.txt b.txt`	Execute text after `!` as a shell command.
`%cd`	Change directory. (`!cd` does not work.)
`files = !ls`	Capture the output of a shell command.

`f1 = ‘a.txt’; f2 = ‘b.txt’`
`!cp $f1 $f2`
	Substitute Python variables into shell commands. (Use `$$` for a literal $.)

Debugging

After an exception, you may use the interactive debugger to investigate the problem. Type `%debug` to activate “post-mortem” debugging mode, where the following commands apply.

`l` *or* `ll`	“List” some or all source code lines in frame.
`u, d`	Move up, down between frames.
`p` *`varname`* *or* `pp` *`varname`*	Print or “pretty-print” *varname*.
`exit`	Quit interactive debugger; return to IPython.

Basic Profiling (software jargon for speed-testing, nothing to do with “IPython profiles”)

`%timeit x = 5`	Time code execution (averaged over loops).

`%%timeit x = 5  # don’t count setup`
`x + 1`
	Execute but do not time the first line. Time execution of all later lines as above.

See also `%prun` and `%lprun` (requires line_profiler, a separate Python package).