IP[y]: IPython Interactive Computing

Warning

This documentation is for an old version of IPython. You can find docs for newer versions here.

Converting notebooks to other formats

Newly added in the 1.0 release of IPython is the `nbconvert` tool, which allows you to convert an `.ipynb` notebook document file into various static formats.

Currently, `nbconvert` is provided as a command line tool, run as a script using IPython. A direct export capability from within the IPython Notebook web app is planned.

The command-line syntax to run the `nbconvert` script is:

```
$ ipython nbconvert --to FORMAT notebook.ipynb
```

This will convert the IPython document file `notebook.ipynb` into the output format given by the `FORMAT` string.

The default output format is html, for which the `--to` argument may be omitted:

```
$ ipython nbconvert notebook.ipynb
```

IPython provides a few templates for some output formats, and these can be specified via an additional `--template` argument.

The currently supported export formats are:

- `--to html`

Table Of Contents

Converting notebooks to other formats

- Notebook JSON file format

  - `--template full` (default)

    A full static HTML render of the notebook. This looks very similar to the interactive view.

  - `--template basic`

    Simplified HTML, useful for embedding in webpages, blogs, etc. This excludes HTML headers.

- `--to latex`

  Latex export. This generates `NOTEBOOK_NAME.tex` file, ready for export. You can automatically run latex on it to generate a PDF by adding `--post PDF`.

  - `--template article` (default)

    Latex article, derived from Sphinx's howto template.

  - `--template book`

    Latex book, derived from Sphinx's manual template.

  - `--template basic`

    Very basic latex output - mainly meant as a starting point for custom templates.

- `--to slides`

  This generates a Reveal.js HTML slideshow. It must be served by an HTTP server. The easiest way to get this is to add `--post serve` on the command-line.

- `--to markdown`

  Simple markdown output. Markdown cells are unaffected, and code cells are placed in triple-backtick (`` ``` ``) blocks.

- `--to rst`

  Basic reStructuredText output. Useful as a starting point for embedding notebooks in Sphinx docs.

- `--to python`

Convert a notebook to an executable Python script. This is the simplest way to get a Python script out of a notebook. If there were any magics in the notebook, this may only be executable from an IPython session.

> Note
>
> nbconvert uses pandoc to convert between various markup languages, so pandoc is a dependency of most nbconvert transforms, excluding Markdown and Python.

The output file created by `nbconvert` will have the same base name as the notebook and will be placed in the current working directory. Any supporting files (graphics, etc) will be placed in a new directory with the same base name as the notebook, suffixed with `_files`:

```
$ ipython nbconvert notebook.ipynb
$ ls
notebook.ipynb   notebook.html    notebook_files/
```

For simple single-file output, such as html, markdown, etc., the output may be sent to standard output with:

```
$ ipython nbconvert --to markdown notebook.ipynb --stdout
```

Multiple notebooks can be specified from the command line:

```
$ ipython nbconvert notebook*.ipynb
$ ipython nbconvert notebook1.ipynb notebook2.ipynb
```

or via a list in a configuration file, say `mycfg.py`, containing the text:

```
c = get_config()
c.NbConvertApp.notebooks = ["notebook1.ipynb", "notebook2.ipynb"]
```

and using the command:

```
$ ipython nbconvert --config mycfg.py
```

Notebook JSON file format

Notebook documents are JSON files with an `.ipynb` extension, formatted as legibly as possible with minimal extra indentation and cell content broken across lines to make them reasonably friendly to use in version-control workflows. You should be very careful if you ever manually edit this JSON data, as it is extremely easy to corrupt its internal structure and make the file impossible to load. In general, you should consider the notebook as a file meant only to be edited by the IPython Notebook app itself, not for hand-editing.

Note

Binary data such as figures are also saved directly in the JSON file. This provides convenient single-file portability, but means that the files can be large; a `diff` of binary data is also not very meaningful. Since the binary blobs are encoded in a single line, they affect only one line of the `diff` output, but they are typically very long lines. You can use the `Cell | All Output | Clear` menu option to remove all output from a notebook prior to committing it to version control, if this is a concern.

The notebook server can also generate a pure Python version of your notebook, using the `File | Download as` menu option. The resulting `.py` file will contain all the code cells from your notebook verbatim, and all Markdown cells prepended with a comment marker. The separation between code and Markdown cells is indicated with special comments and there is a header indicating the format version. All output is removed when exporting to Python.

As an example, consider a simple notebook called `simple.ipynb` which contains one Markdown cell, with the content `The simplest notebook.`, one code input cell with the content `print "Hello, IPython!"`, and the corresponding output.

The contents of the notebook document `simple.ipynb` is the following JSON container:

```
{
 "metadata": {
  "name": "simple"
 },
 "nbformat": 3,
 "nbformat_minor": 0,
 "worksheets": [
  {
   "cells": [
    {
     "cell_type": "markdown",
     "metadata": {},
     "source": "The simplest notebook."
    },
    {
     "cell_type": "code",
     "collapsed": false,
     "input": "print \"Hello, IPython\"",
     "language": "python",
     "metadata": {},
     "outputs": [
      {
       "output_type": "stream",
       "stream": "stdout",
       "text": "Hello, IPython\n"
      }
     ],
     "prompt_number": 1
    }
   ],
   "metadata": {}
  }
 ]
}
```

The corresponding Python script is:

```
# -*- coding: utf-8 -*-
# <nbformat>3.0</nbformat>

# <markdowncell>

# The simplest notebook.

# <codecell>

print "Hello, IPython"
```

Note that indeed the output of the code cell, which is present in the JSON container, has been removed in the `.py` script.