

Jupyter Data Science Workflow, Remastered

Michał Wojczulis

Jupyter Workflow

In this session we will try to improve the classic workflow by Jake Vanderplas Reproducible Data Analysis in Jupyter.



Jake Vanderplas

jakevdp

 Overview

 Repositories **220**

 Projects

Pinned

 **PythonDataScienceHandbook** Public

Python Data Science Handbook: full text in Jupyter Notebooks

 Jupyter Notebook  37.5k  16.5k

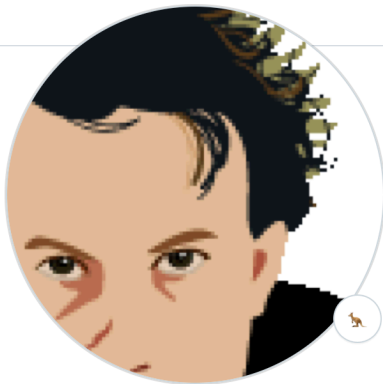
 **WhirlwindTourOfPython** Public

The Jupyter Notebooks behind my O'Reilly report, "A Whirlwind Tour of Python"

 Jupyter Notebook  3.3k  1.5k

Nbdev

Nbdev is a notebook-driven development platform. With nbdev, you get documentation, tests, continuous integration, and packaging.



Jeremy Howard

jph00

 **Overview**


 Repositories 57

 Projects

Pinned

 **fastai/fastai** Public

The fastai deep learning library

 Jupyter Notebook  23.4k  7.4k

 **fastai/fastcore** Public

Python supercharged for the fastai library

 Jupyter Notebook  820  239

 **latex-npt** Public

Agenda

- ▶ Hands-on Walkthrough

Extras

- ▶ Notebook Best Practices
- ▶ Qmd Documents
- ▶ RenderScripts
- ▶ Git-Friendly Jupyter
- ▶ Blogging
- ▶ Pre-Commit Hooks
- ▶ Documentation Only Sites
- ▶ Modular nbdev
- ▶ Nbdev plugins

Hands-on Walkthrough

Live coding.

What could possibly go wrong?

Notebook Best Practices

Document parameters with docments

With docments, this function:

```
def draw_n(n:int, # Number of cards to draw
           replace:bool=True # Draw with replacement?
           )->list: # List of cards
    "Draw `n` cards."
```

...would include the following table as part of its documentation:

| | Type | Default | Details |
|----------------|-------------|---------|-------------------------|
| n | int | | Number of cards to draw |
| replace | bool | True | Draw with replacement? |
| Returns | list | | List of cards |

Notebook Best Practices

Code examples as tests by adding assertions

`fastcore.test` provides a set of light wrappers around `assert` for better notebook tests (for example, they print both objects on error if they differ).

Here's an example using `fastcore.test.test_eq`:

```
def inc(x): return x + 1
test_eq(inc(3), 4)
```


Notebook Best Practices

Document error-cases as tests

Nbdev recommends documenting errors with actual failing code using `fastcore.test.test_fail`. For example:

```
def divide(x, y): return x / y
test_fail(lambda: divide(1, 0), contains="division by zero")
```

Qmd documents

Qmd documents are Markdown documents, but with extra functionality provided by Quarto and Pandoc.

For example images arranged into layouts.

```
::: {layout-ncol=3}
! [Jupyter] (jupyter.jpg){width=100px fig-align="left"}

! [Vscode] (vscode.jpg){width=100px fig-align="left"}

! [Git] (git.jpg){width=100px fig-align="left"}
:::
```





Figure 2: Vscode

Figure 3: Git

RenderScripts

RenderScripts are just like regular Python scripts. These scripts are run when your site is rendered.

For example to produce below table from a python list, the following script is used:

| | Name | Position |
|---|----------------|-----------------|
|  | Chris Lattner | Inventor |
|  | Fernando Pérez | Creator |

```
testimonials = [
```

```
    (chris_lattner.png, 'Chris Lattner', 'Inventor of Swift')
```

Git-Friendly Jupyter

Jupyter notebooks don't work with version control by default. Nbdev provides a set of hooks which enable git-friendly Jupyter notebooks in any git repo.

nbdev provides three hooks to ease Jupyter-git integration.

- ▶ `nbdev_merge` on merging notebooks with git, that automatically fixes conflicting outputs and metadata
- ▶ `nbdev_clean` on saving notebooks in Jupyter, to automatically clean unwanted metadata and outputs from your notebooks
- ▶ `nbdev_trust` after merging notebooks with git, to trust a repo once-off, and all notebooks and changes thereafter

Bloggng

Nbdev uses Quarto for blogging via Jupyter Notebooks. Although nbdev is not required to blog with notebooks, it will add some functionality (testing, exporting, adding blog to nbdev project website)

Pre-Commit Hooks

Nbdev provides hooks for the pre-commit framework to catch and fix uncleaned and unexported notebooks, locally, without having to wait for continuous integration pipelines to run:

1. pre-commit runs each hook on your staged changes
2. If a hook changes files pre-commit stops the commit, leaving those changes as unstaged
3. You can now stage those changes and make any edits required to get pre-commit to pass
4. Redo the git commit, and if it succeeds, your commit will be created.

Documentation Only Sites

Nbdev can be used for the purposes of documenting existing code. For example, you can use the following features of nbdev without creating a python package:

- ▶ Custom nbdev directives such as `#|hide_line`.
- ▶ Testing with `nbdev_test`.
- ▶ Automated entity linking with `doclinks`.
- ▶ Rendering API documentation with `documents` and `show_doc`.

Modular nbdev

You can use various nbdev tools separately:

- ▶ Document existing code with `show_doc`
- ▶ Testing notebooks with `nbdev_test`
- ▶ Export code to modules with `nb_export`
- ▶ Jupyter-git integration
- ▶ Python packaging, utilities for easy packaging on PyPI, conda, and GitHub

Nbdev plugins

With nbdev, it's possible to customize and extend it further beyond the standard capabilities.