The Jupyter Notebook

We will be using Jupyter notebooks for looking at examples of python code, as well as for viewing instructions, hints, and commiting code for our homework assignments. The information below provides you with an basic understanding of what is contained within a Jupyter Notebook. Take a look at the 01_logging_into_the_hpc.pdf document to learn how to create/access Jupyter notebooks on the campus HPC.

Introduction

The Jupyter notebook extends the console-based approach to interactive computing in a qualitatively new direction, providing a web-based application suitable for capturing the whole computation process: developing, documenting, and executing code, as well as communicating the results. The Jupyter notebook combines two components: A web application: a browser-based tool for interactive authoring of documents which combine explanatory text, mathematics, computations and their rich media output. Notebook documents: a representation of all content visible in the web application, including inputs and outputs of the computations, explanatory text, mathematics, images, and rich media representations of objects.

Notebook documents

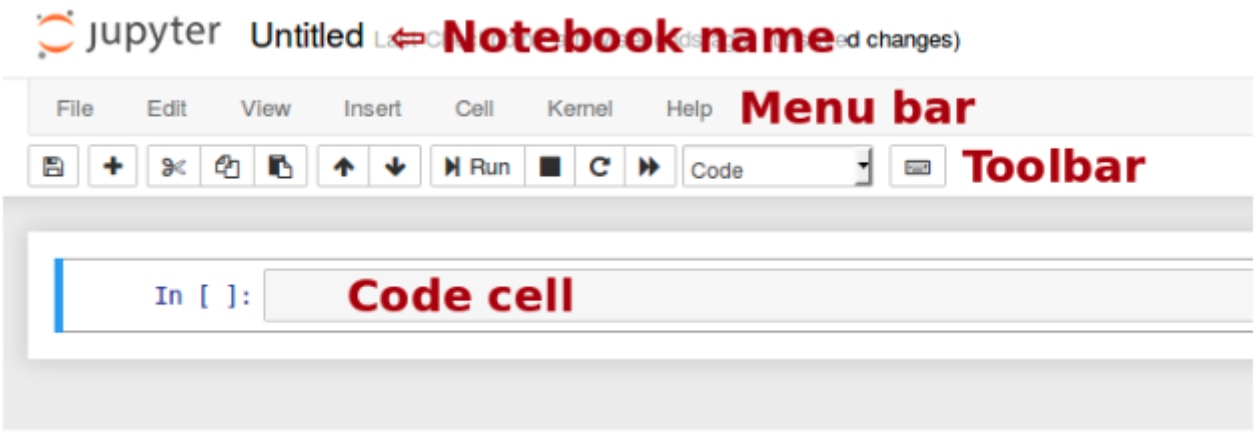
Notebook documents contains the inputs and outputs of a interactive session as well as additional text that accompanies the code but is not meant for execution. In this way, notebook files can serve as a complete computational record of a session, interleaving executable code with explanatory text, mathematics, and rich representations of resulting objects. These documents are internally JSON files and are saved with the .ipynb extension. Since JSON is a plain text format, they can be version-controlled and shared with colleagues.

Notebooks may be exported to a range of static formats, including HTML (for example, for blog posts), reStructuredText, LaTeX, PDF, and slide shows, via the nbconvert command.

Furthermore, any .ipynb notebook document available from a public URL can be shared via the Jupyter Notebook Viewer (nbviewer). This service loads the notebook document from the URL and renders it as a static web page. The results may thus be shared with a colleague, or as a public blog post, without other users needing to install the Jupyter notebook themselves. In effect, nbviewer is simply nbconvert as a web service, so you can do your own static conversions with nbconvert, without relying on nbviewer.

Notebook user interface

When you create a new notebook document, you will be presented with the notebook name, a menu bar, a toolbar and an empty code cell.



- **Notebook name**: The name displayed at the top of the page, next to the Jupyter logo, reflects the name of the .ipynb file. Clicking on the notebook name brings up a dialog which allows you to rename it. Thus, renaming a notebook from “Untitled0” to “My first notebook” in the browser, renames the Untitled0.ipynb file to My first notebook.ipynb.
- **Menu bar**: The menu bar presents different options that may be used to manipulate the way the notebook functions.
- **Toolbar**: The tool bar gives a quick way of performing the most-used operations within the notebook, by clicking on an icon. Code cell: the default type of cell; read on for an explanation of cells.

Structure of a notebook document

The notebook consists of a sequence of cells. A cell is a multiline text input field, and its contents can be executed by using Shift-Enter, or by clicking either the “Play” button the toolbar, or Cell, Run in the menu bar. The execution behavior of a cell is determined by the cell’s type. There are three types of cells: code cells, markdown cells, and raw cells. Every cell starts off being a code cell, but its type can be changed by using a drop-down on the toolbar (which will be “Code”, initially), or via keyboard shortcuts.

For more information on the different things you can do in a notebook, see the collection of examples.

Code cells

A code cell allows you to edit and write new code, with full syntax highlighting and tab completion. The programming language you use depends on the kernel, and the default kernel (IPython) runs Python code. When a code cell is executed, code that it contains is sent to the kernel associated with the notebook. The results that are returned from this computation are then displayed in the notebook as the cell’s output. The output is not limited to text, with many other possible forms of output are also possible, including matplotlib figures and HTML tables (as used, for example, in the pandas data analysis package). This is known as IPython’s rich display capability.

See also Rich Output example notebook

Markdown cells

You can document the computational process in a literate way, alternating descriptive text with code, using rich text. In IPython this is accomplished by marking up text with the Markdown language. The corresponding cells are called Markdown cells. The Markdown language provides a simple way to perform

this text markup, that is, to specify which parts of the text should be emphasized (italics), bold, form lists, etc.

If you want to provide structure for your document, you can use markdown headings. Markdown headings consist of 1 to 6 hash # signs # followed by a space and the title of your section. The markdown heading will be converted to a clickable link for a section of the notebook. It is also used as a hint when exporting to other document formats, like PDF.

When a Markdown cell is executed, the Markdown code is converted into the corresponding formatted rich text. Markdown allows arbitrary HTML code for formatting.

Within Markdown cells, you can also include mathematics in a straightforward way, using standard LaTeX notation: $...$ for inline mathematics and $$...$$ for displayed mathematics. When the Markdown cell is executed, the LaTeX portions are automatically rendered in the HTML output as equations with high quality typography. This is made possible by MathJax, which supports a large subset of LaTeX functionality.

Standard mathematics environments defined by LaTeX and AMS-LaTeX (the amsmath package) also work, such as \begin{equation}...\end{equation}, and \begin{align}...\end{align}. New LaTeX macros may be defined using standard methods, such as \newcommand, by placing them anywhere between math delimiters in a Markdown cell. These definitions are then available throughout the rest of the IPython session.

See also Working with Markdown Cells example notebook

Raw cells

Raw cells provide a place in which you can write output directly. Raw cells are not evaluated by the notebook. When passed through nbconvert, raw cells arrive in the destination format unmodified. For example, you can type full LaTeX into a raw cell, which will only be rendered by LaTeX after conversion by nbconvert.

Basic workflow

The normal workflow in a notebook is, then, quite similar to a standard IPython session, with the difference that you can edit cells in-place multiple times until you obtain the desired results, rather than having to rerun separate scripts with the %run magic command. Typically, you will work on a computational problem in pieces, organizing related ideas into cells and moving forward once previous parts work correctly. This is much more convenient for interactive exploration than breaking up a computation into scripts that must be executed together, as was previously necessary, especially if parts of them take a long time to run.

To interrupt a calculation which is taking too long, use the Kernel, Interrupt menu option, or the i,i keyboard shortcut. Similarly, to restart the whole computational process, use the Kernel, Restart menu option or 0,0 shortcut.

A notebook may be downloaded as a .ipynb file or converted to a number of other formats using the menu option File, Download as.

See also

Running Code in the Jupyter Notebook example notebook

Notebook Basics Example Notebook

Keyboard shortcuts

All actions in the notebook can be performed with the mouse, but keyboard shortcuts are also available for the most common ones. The essential shortcuts to remember are the following:

- Shift-Enter: run cell Execute the current cell, show any output, and jump to the next cell below. If Shift-Enter is invoked on the last cell, it makes a new cell below. This is equivalent to clicking the Cell, Run menu item, or the Play button in the toolbar.
- Esc: Command mode In command mode, you can navigate around the notebook using keyboard shortcuts.
- Enter: Edit mode In edit mode, you can edit text in cells.

For the full list of available shortcuts, click Help, Keyboard Shortcuts in the notebook menus.

Running Code

First and foremost, the Jupyter Notebook is an interactive environment for writing and running code. The notebook is capable of running code in a wide range of languages. However, each notebook is associated with a single kernel. This notebook is associated with the IPython kernel, therefore runs Python code.

Code cells allow you to enter and run code

Run a code cell using Shift-Enter or pressing the button in the toolbar. Fpr example try running this in a Python notebook code cell

```
a = 10
print(a)
10
```

There are two other keyboard shortcuts for running code:

- Alt-Enter runs the current cell and inserts a new one below.
- Ctrl-Enter run the current cell and enters command mode.

Managing the Kernel

Code is run in a separate process called the Kernel. The Kernel can be interrupted or restarted.

Cell menu

The “Cell” menu has a number of menu items for running code in different ways. These include:

- Run and Select Below
- Run and Insert Below
- Run All
- Run All Above
- Run All Below

Restarting the kernel

The kernel maintains the state of a notebook's computations. You can reset this state by restarting the kernel.

Output is asynchronous

All output is displayed asynchronously as it is generated in the Kernel. If you execute the next cell, you will see the output one piece at a time, not all at the end. For example,

```
import time, sys
for i in range(8):
    print(i)
    time.sleep(0.5)
```

```
0
1
2
3
4
5
6
7
```

Markdown Cells

Text can be added to Jupyter Notebooks using Markdown cells. You can change the cell type to Markdown by using the Cell menu, the toolbar, or the key shortcut m. Markdown is a popular markup language that is a superset of HTML. Its specification can be found here

Markdown basics

Markdown is easy to learn. Checkout this cheatsheet for a quick reference.

Heading:

```
# H1
## H2
### H3
```

Bold:

```
**bold text**
```

Italicize:

```
*italicized text*
```

Block quote:

```
> blockquote
```

Ordered list:

```
- First item
- Second item
- Third item
```

Unrdered list:

```
1. First item
2. Second item
3. Third item
```

Code:

```
`code`
```

Block of code:

```
```code```
```

Horizontal rule:

```
---
```

Link:

```
[title](https://www.example.com)
```

Image:
```

```
![alt text](image.jpg)
```

Keep in mind that attached files will increase the size of your notebook. You can manually edit the attachement by using the View > Cell Toolbar > Attachment menu, but you should not need to.

Table:

```
And a table like this:
| This | is   |
|------|------|
|   a  | table|
```

Don't want to use md characters? You can use backslash to generate literal characters which would otherwise have special meaning in the Markdown syntax.