Jupyter Notebook Basics

The Jupyter Notebook, which evolved from the IPython Notebook, is a web-based interactive computational environment. It uniquely combines code execution, text, and multimedia resources into a single document. Initially developed for Python, it now supports over 40 programming languages.

A Jupyter Notebook is a JavaScript Object Notation (JSON) document containing an ordered list of input/output cells. These cells can hold code, text, mathematical expressions, plots, and various forms of rich media, making them a versatile tool for data analysis, scientific research, and educational purposes.

The Jupyter notebook (successor of IPython notebook) is a web-based interactive computational environment which provides a unique combination of code, shell environment and text. The IPython notebook project that started off as a tool to provide the above-mentioned functionalities just for Python has since grown to be language agnostic supporting over 40 different languages and is now known as Jupyter notebook (or simply notebook). Simply put, a Jupyter notebook is a JSON document containing an ordered list of input/output cells which can contain code, text, mathematics, plots, and rich media.

The Jupyter Notebook Combines Three Components:

- **The notebook web application:** An interactive web application for writing and running code interactively and authoring notebook documents.
- **Kernels:** Separate processes started by the notebook web application that runs user's code in a given language and returns output back to the notebook web application. The kernel also handles things like computations for interactive widgets, tab completion, and introspection. The Jupyter notebook, for running Python codes, runs an IPython kernel. If you are viewing this document on CUSP CDF and if you run any code, the code is not executed on your local machine. It is executed on the CDF server where the IPython kernel is running.
- **Notebook documents:** Self-contained documents that contain a representation of all the contents visible in the notebook web application, including inputs and outputs of the computations, narrative text, equations, images, and rich media representations of objects. Each notebook document has its own kernel.

Why Use a Notebook Instead of a Terminal or Text Editor?

The notebook web application enables users to:

- Edit code in the browser, with automatic syntax highlighting, indentation, and tab completion/introspection.
- Run code from the browser, with the results of computations attached to the code which generated them.
- See the results of computations with rich media representations, such as HTML, LaTeX, PNG, SVG, PDF, etc.
- Create and use interactive JavaScript widgets, which bind interactive user interface controls and visualizations to reactive kernel-side computations.
- Author narrative text using the Markdown markup language.
- Build hierarchical documents that are organized into sections with different levels of headings.
- Use different cell types, for example, to include mathematical equations using LaTeX, syntax in Markdown, which are rendered in-browser by MathJax.

Different Cell Types

Notebooks consist of a linear sequence of cells. There are four basic cell types:

- **Code cells:** Input and output of live code that is run in the kernel.
- **Markdown cells:** Narrative text with embedded LaTeX equations.
- **Heading cells:** Six levels of hierarchical organization and formatting.
- **Raw cells:** Unformatted text that is included, without modification, when notebooks are converted to different formats using nbconvert.

Jupyter Notebook offers numerous benefits, making it a popular choice among data scientists, researchers, and educators:

- **Interactive Learning:** The ability to write and execute code in small chunks allows for a more interactive and engaging learning experience.
- **Visualization:** Easy integration with libraries like Matplotlib and Seaborn allows for the creation of detailed and interactive visualizations.
- **Reproducibility:** Notebooks can be shared with others, ensuring that they see the exact same code, results, and explanations. This makes it easier to reproduce and validate findings.

- **Documentation:** Combining code with rich text elements such as explanations, links, and images makes it a powerful tool for creating comprehensive documentation and tutorials.
- **Collaboration:** Notebooks can be shared and collaborated on using platforms like GitHub, making it easy to work on projects with others.

Common Use Cases

Jupyter Notebooks are used in a wide variety of applications, including:

- **Data Cleaning and Transformation:** Easily clean, process, and transform data for analysis.
- **Numerical Simulation:** Run simulations to model and solve mathematical problems.
- **Statistical Modeling:** Apply statistical techniques to data to uncover patterns and make predictions.
- **Machine Learning:** Develop, train, and test machine learning models.
- **Data Visualization:** Create interactive and static visualizations to explore and present data insights.

Example: A Simple Python Code in Jupyter Notebook

Here is a simple example of how to use a Jupyter Notebook to write and run Python code:

```
# This is a code cell

print("Hello, World!")

# Run this cell to see the output below
```

When you run the above cell in a Jupyter Notebook, you will see the output "Hello, World!" directly below the code. This interactive environment allows you to experiment with code, see immediate results, and make changes quickly.

Jupyter Notebooks are used across various industries and careers, here are some of the key industries and careers:

- **Urban Planning and Development:** Urban Planners, GIS Analysts, Transportation Engineers, City Planners.
- **Data Science and Analytics:** Data Scientists, Data Analysts, Machine Learning Engineers, Business Intelligence Analysts.

- **Public Policy and Governance:** Policy Analysts, Public Administration Officials, Government Data Scientists, Civic Tech Developers.
- **Environmental Science and Sustainability:** Environmental Scientists, Sustainability Analysts, Conservation Scientists, Climate Change Analysts.
- **Healthcare and Public Health:** Public Health Analysts, Epidemiologists, Healthcare Data Scientists, Bioinformatics Specialists.
- **Education and Research:** Academic Researchers, Educators, Educational Data Analysts, Research Scientists.

Practical Use Cases

- **Data Analysis and Visualization:** Analyzing transportation data to identify congestion hotspots and propose alternative routes.
- **Machine Learning Model Development:** Developing predictive models for disease outbreaks using historical health data.
- **Interactive Reporting:** Creating interactive environmental impact reports for stakeholders.
- **Education and Training:** Using Jupyter Notebooks as an interactive teaching tool to demonstrate data analysis techniques.
- **Policy Impact Analysis:** Analyzing the effects of a new policy on urban mobility and presenting the findings in an interactive Jupyter Notebook.

For those interested in learning more about Jupyter Notebooks, the following resources are highly recommended:

- Official Jupyter Project: The official site for Jupyter Notebooks, providing comprehensive documentation and resources.
- Real Python: Jupyter Notebook: A detailed introduction to Jupyter Notebooks with practical examples.
- NBViewer: A web application to view Jupyter Notebooks online without needing to install anything.
- Gallery of Interesting Jupyter Notebooks: A collection of Jupyter Notebooks showcasing various use cases and applications.
- Dataquest Jupyter Notebook Tutorial: A comprehensive tutorial for beginners to get started with Jupyter Notebooks.

Jupyter Notebooks have become an invaluable tool across numerous industries, providing a versatile and interactive platform for data analysis, visualization, and machine learning. Whether you are working in urban planning, data science, public policy, environmental science, healthcare, or education, Jupyter Notebooks offer powerful capabilities to enhance your work and achieve impactful results.

Setting Up Jupyter Notebook Locally on Your Machine Using Anaconda

Note: This method is recommended for most users.

For new users, we highly recommend installing Anaconda. Anaconda conveniently installs Python, Jupyter Notebook, and other commonly used packages for scientific computing and data science.

Use the following installation steps:

1. Download Anaconda. We recommend downloading Anaconda's latest Python 3 version (currently Python 3.7).
2. Install the version of Anaconda which you downloaded, following the instructions on the download page.
3. Congratulations, you have installed Jupyter Notebook.

Launching Jupyter Notebook

To launch Jupyter Notebook, open a command line window (Terminal on Mac, cmd.exe on Windows) and type jupyter notebook, then press Return. The Jupyter Notebook dashboard will launch in a web browser window. You can learn more by reviewing this section of the official Jupyter Notebook documentation.

Creating a New Notebook

Once you navigate to your home directory, you can create a new notebook by clicking on "New" on the top right corner and selecting "Python 3".

You must select Python 3 as the kernel.

Notebook Modes

Jupyter Notebook has a modal user interface. This means that the keyboard does different things depending on which mode the Notebook is in. There are two modes: Edit mode and Command mode.

Edit mode: Edit mode is indicated by a colored (usually green) cell border and a prompt in the editor area. When a cell is in edit mode, you can type in the cell, like a normal text editor. To enter edit mode, navigate to the cell and press Enter.

Command mode: Command mode is indicated by a (usually) grey cell border with a blue left margin. To enter into command mode, press the Esc key. This will bring you out of edit mode.

Different Cell Types

Notebooks consist of a linear sequence of cells. Here are four basic cell types:

- Code cells: Input and output of live code that is run in the kernel.
- Markdown cells: Narrative text with embedded LaTeX equations.
- Heading cells: 6 levels of hierarchical organization and formatting.
- Raw cells: Unformatted text that is included without modification. When Notebooks are converted to different formats using nbconvert, these cell types can be viewed by clicking Cell -> Cell Type in the menu bar.

Keyboard Shortcuts

Operations in Command Mode

In command mode, the keyboard is mapped to a set of shortcuts that let you perform notebook and cell actions efficiently. It's recommended to learn the command mode shortcuts in the following rough order:

For basic navigation:

If the cell is highlighted in blue, you are navigating the notebook and not editing it. Enter edit mode by either clicking in the cell or navigating to the cell (using arrow keys) and pressing Enter.

You know that you are in edit mode once the highlighted cell color changes from blue to green.

Note: The above explanation of blue and green colors is valid only if you are not using any custom themes.

For saving the current notebook:

You can press the S key when you are in command mode (i.e., when the cell you are in is highlighted in blue).

For changing the cell type:

The cells in the Jupyter notebook can be filled with text in different formats. You can invoke them using the following shortcut keys:

y -- For code cell

m -- For markdown mode

Operations in Edit Mode

For executing the cell:

To execute the current cell that you are editing, type Shift + Enter together.

To know more shortcuts, click on the help menu and select JupyterLab reference.

Closing a Notebook

When a Notebook is currently open, its icon will show as green. Notebooks remain running until you explicitly shut them down.

To shut down a notebook, select "Running Terminals and Kernels" from the left pane and then click "shutdown" on the notebooks that you don't need.

(You can also right-click the notebook from the file browser pane and select "Shutdown Kernel")

Additional Resources

It's important to explore alternative platforms for coding and data science projects and learn about Jupyter Notebook. Google Colaboratory is a platform that offers functionalities similar to Jupyter Notebook in an online environment. I recommend watching the YouTube video "Get Started with Google Colaboratory (Coding TensorFlow)" for a quick introduction to Google Colaboratory and how it can be used for coding projects, particularly machine learning projects with TensorFlow.

While Jupyter Notebook remains our primary focus, exploring Google Colaboratory offers a complementary learning experience. Comparing and contrasting these platforms will deepen your understanding of notebook environments and enhance your coding and data science proficiency.

This video caters to auditory and visual learners who may find videos more engaging and accessible. Providing various resources ensures that everyone can learn in a way that resonates with them. After watching the video, feel free to explore Google Colaboratory further. Experiment with its features, try out new projects and see how it compares to

Jupyter Notebook. This self-directed exploration fosters curiosity, creativity, and independence in your learning journey.

Exploring Essential Python Libraries:

Throughout this course, we'll continue our journey from Jupyter Notebook into essential Python libraries used across various domains such as data science, web development, machine learning, and web scraping. Here is a glance of what's ahead:

Foundational Data Science Libraries: NumPy, Matplotlib, SciPy, Pandas:

- **NumPy (Numerical Python):** Considered the bedrock of scientific computing in Python, NumPy empowers users with robust numerical operations and versatile array manipulation capabilities. It is the backbone for many other Python libraries in the data science ecosystem.
- **Matplotlib:** Matplotlib offers various plotting functionalities, allowing users to create diverse visualizations effectively. Whether static, interactive, or animated visualizations, Matplotlib has you covered.
- **SciPy:** Built upon NumPy, SciPy extends its capabilities by providing additional tools for scientific computing tasks such as optimization, integration, and linear algebra. It complements NumPy with higher-level algorithms and functionalities.
- **Pandas:** Pandas simplifies data manipulation and analysis with its intuitive data structures, such as DataFrame and Series. It's your go-to tool for data cleaning, transformation, and analysis, making it indispensable in data science projects.