Jupyter Cheat Sheet for Quick Basic Fundamentals

Welcome to your comprehensive quick reference guide for Jupyter! This cheat sheet covers the fundamental concepts you need to get started with Jupyter Notebooks, enhanced with extended examples for better understanding and quick revision.

Table of Contents

Introduction

Overview: Jupyter Notebooks are an open-source web application that allows you to create and share documents containing live code, equations, visualizations, and narrative text. They are widely used for data analysis, machine learning, scientific research, and education.

Installation and Setup

Overview: Set up Jupyter Notebooks on your local machine or use cloud-based platforms for a seamless experience.

Using Anaconda (Recommended)

Anaconda is a popular distribution that includes Jupyter and many other data science packages.

1. **Download Anaconda:**
    - Visit Anaconda Downloads and download the installer for your operating system.
2. **Install Anaconda:**

- Follow the installation instructions specific to your OS.

3. **Launch Jupyter Notebook:**
   - Open the Anaconda Navigator and click on **Launch** under Jupyter Notebook.
   - Alternatively, open your terminal or command prompt and type:

```
jupyter notebook
```

Using pip

If you prefer using `pip`, ensure you have Python installed.

1. **Install Jupyter:**

```
pip install jupyter
```

2. **Launch Jupyter Notebook:**

```
jupyter notebook
```

Cloud-Based Platforms

- **Google Colab:** Free cloud-based Jupyter environment.
  - Visit Google Colab and sign in with your Google account.
- **Binder:** Shareable Jupyter environments.
  - Visit Binder and provide a GitHub repository to launch a notebook.

Interface Overview

Overview: Familiarize yourself with the Jupyter Notebook interface to navigate and utilize its features effectively.

Main Components

1. **Menu Bar:** Contains options like File, Edit, View, Insert, Cell, Kernel, Widgets, and Help.
2. **Toolbar:** Offers quick access to common actions such as save, add cell, cut, copy, paste, and cell type selection.
3. **Notebook Area:** The main area where you create and edit cells.
4. **Sidebar (Optional):** Some extensions add a sidebar for additional functionality.

Modes

- **Command Mode (Blue):** Used for notebook-level operations. Shortcuts like `A`, `B`, `D`, etc., work here.

- **Edit Mode (Green):** Used for editing the content of a cell. Shortcuts like `Ctrl + Enter`, `Shift + Enter`, etc., work here.

Creating and Managing Notebooks

Overview: Learn how to create, save, rename, and organize your Jupyter Notebooks.

Creating a New Notebook

1. **Via Jupyter Interface:**
    - Click on the **New** button on the top right.
    - Select the desired kernel (e.g., Python 3).
2. **Via Terminal:**

    ```
    jupyter notebook
    ```

    - In the opened browser window, navigate to the desired directory.
    - Click **New** > **Python 3**.

Saving Notebooks

- **Auto-Save:** Jupyter automatically saves your notebook periodically.
- **Manual Save:** Click the **Save** icon or press `Ctrl + S` (`Cmd + S` on Mac).

Renaming Notebooks

1. Click on the notebook name at the top (e.g., `Untitled.ipynb`).
2. Enter the new name and click **Rename**.

Organizing Notebooks

- **Folders:** Create folders to organize related notebooks.
- **Tags:** Use extensions like nbextensions to add tagging functionality.

Closing and Shutting Down Notebooks

- **Close Tab:** Simply close the browser tab.
- **Shutdown Kernel:** In the Jupyter dashboard, select the notebook and click **Shutdown**.

Cell Types

Overview: Jupyter Notebooks consist of cells. Understanding different cell types is crucial for effective notebook usage.

Code Cells

- **Purpose:** Write and execute code.
- **Features:** Display output, including text, tables, plots, etc.

```
# Example: Python code cell
print("Hello, Jupyter!")
```

Markdown Cells

- **Purpose:** Add formatted text, images, links, equations, and more.
- **Features:** Support for Markdown syntax.

```
# Heading 1
## Heading 2
**Bold Text**
*Italic Text*
- List Item 1
- List Item 2
```

Raw Cells

- **Purpose:** Include unformatted text or code that won't be executed or rendered.
- **Usage:** Useful for notes or code snippets in their raw form.

```
This is a raw cell. It will not be rendered or executed.
```

Changing Cell Types

1. **Via Toolbar:**
   - Select the cell.
   - Choose **Code**, **Markdown**, or **Raw** from the dropdown menu.
2. **Via Keyboard Shortcuts:**
   - Press `Esc` to enter Command Mode.
   - Press `Y` for Code, `M` for Markdown, or `R` for Raw.

Markdown in Jupyter

Overview: Enhance your notebooks with formatted text, images, links, and mathematical equations using Markdown.

Basic Markdown Syntax

- **Headings:**

```
# Heading 1
## Heading 2
### Heading 3
```

- **Emphasis:**

```
*Italic* or _Italic_
**Bold** or __Bold__
***Bold and Italic***
```

- **Lists:**
  - **Unordered:**

```
- Item 1
- Item 2
  - Subitem 2.1
```

  - **Ordered:**

```
1. First
2. Second
   1. Subitem 2.1
```

- **Links:**

```
[OpenAI](https://www.openai.com/)
```

- **Images:**

```
![Alt Text](https://via.placeholder.com/150)
```

- **Blockquotes:**

```
> This is a blockquote.
```

- **Code Blocks:**

- **Inline Code:**

```
Use the `print()` function.
```

- **Multiline Code:**

````
```python
def greet(name):
    return f"Hello, {name}!"
```
````

```

```

- **Tables:**

```
| Header 1 | Header 2 |
|----------|----------|
| Row1Col1 | Row1Col2 |
| Row2Col1 | Row2Col2 |
```

Mathematical Equations

- **Inline Equations:**

```
The equation of a line is \( y = mx + b \).
```

- **Displayed Equations:**

```
\[
E = mc^2
\]
```

Extended Examples

```
# Project Title

## Introduction

This project explores the relationship between **temperature** and **pressure** in
a gas sample.
```

```
### Data Collection

- Temperature measured in Celsius.
- Pressure measured in Pascals.

### Results

| Temperature (°C) | Pressure (Pa) |
|------------------|---------------|
| 0                | 101325        |
| 20               | 120000        |
| 40               | 150000        |

### Conclusion

The data suggests a direct correlation between temperature and pressure, as
described by the ideal gas law:

\[
PV = nRT
\]
```

Running and Executing Cells

Overview: Learn how to execute cells to run code and render Markdown.

Executing Code Cells

- **Run Current Cell:**
  - Click the **Run** button in the toolbar.
  - Press `Shift + Enter`.
- **Run and Insert Below:**
  - Press `Alt + Enter` to run the current cell and insert a new cell below.
- **Run All Cells:**
  - Navigate to **Cell** > **Run All** in the menu bar.

Rendering Markdown Cells

- **Render Markdown:**
  - Press `Shift + Enter` while in a Markdown cell to render the formatted text.

Interrupting and Restarting the Kernel

- **Interrupt Execution:**

    - Click the **Interrupt** button (stop icon) in the toolbar or press `I, I` in Command Mode.

- **Restart Kernel:**
    - Navigate to **Kernel** > **Restart** in the menu bar.

Clearing Outputs

- **Clear Output of a Single Cell:**
    - Click on the cell, then navigate to **Cell** > **Current Outputs** > **Clear**.
- **Clear All Outputs:**
    - Navigate to **Cell** > **All Outputs** > **Clear**.

Keyboard Shortcuts

Overview: Boost your productivity by mastering Jupyter's keyboard shortcuts.

Entering and Exiting Modes

- **Enter Command Mode:** Press `Esc`.
- **Enter Edit Mode:** Press `Enter`.

Command Mode Shortcuts

- **Add Cell Above:** `A`
- **Add Cell Below:** `B`
- **Delete Cell:** `D, D` (press `D` twice)
- **Move Cell Up:** `Shift + K` or `Ctrl + ↑`
- **Move Cell Down:** `Shift + J` or `Ctrl + ↓`
- **Change Cell to Code:** `Y`
- **Change Cell to Markdown:** `M`
- **Change Cell to Raw:** `R`
- **Undo Delete Cell:** `Z`
- **Save Notebook:** `Ctrl + S` (`Cmd + S` on Mac)
- **Run All Cells:** `Ctrl + Shift + Enter`

Edit Mode Shortcuts

- **Indent:** `Tab`
- **Dedent:** `Shift + Tab`
- **Undo:** `Ctrl + Z`
- **Redo:** `Ctrl + Shift + Z`
- **Cut Cell:** `Ctrl + X`
- **Copy Cell:** `Ctrl + C`
- **Paste Cell Below:** `Ctrl + V`
- **Select All:** `Ctrl + A`
- **Find and Replace:** `Ctrl + F`

Display All Shortcuts

- Press `H` in Command Mode to display the list of all keyboard shortcuts.

Magic Commands

Overview: Utilize Jupyter's magic commands to perform various tasks, from system operations to timing code execution.

Types of Magic Commands

- **Line Magics:** Prefixed with `%`, operate on a single line.
- **Cell Magics:** Prefixed with `%%`, operate on the entire cell.

Common Magic Commands

Line Magics

- **%pwd:** Print the current working directory.

```
%pwd
```

- **%ls or %dir:** List files in the current directory.

```
%ls
```

- **%time:** Time the execution of a single statement.

```
%time sum(range(100000))
```

- **%timeit:** Time the execution of a single statement multiple times for more accurate results.

```
%timeit sum(range(100000))
```

- **%who:** List all variables in the namespace.

```
%who
```

- **%matplotlib inline:** Display matplotlib plots inline within the notebook.

```
%matplotlib inline
import matplotlib.pyplot as plt
plt.plot([1, 2, 3], [4, 5, 6])
plt.show()
```

Cell Magics

- **%%time:** Time the execution of an entire cell.

```
%%time
total = 0
for i in range(100000):
    total += i
```

- **%%timeit:** Time the execution of an entire cell multiple times.

```
%%timeit
total = 0
for i in range(100000):
    total += i
```

- **%%bash:** Execute cell content as Bash commands.

```
%%bash
echo "Listing files:"
ls
```

- **%%latex:** Render cell content as LaTeX.

```
%%latex
\[
E = mc^2
\]
```

- **%%javascript:** Execute cell content as JavaScript.

```
%%javascript
alert("Hello from JavaScript!")
```

Extended Examples

Using %time and %timeit

```
# Using %time
%time sum(range(100000))

# Using %timeit
%timeit sum(range(100000))
```

Using %%bash

```
%%bash
mkdir test_folder
cd test_folder
touch example.txt
ls
```

Using %%latex

```
%%latex
\[
\int_{a}^{b} f(x) dx = F(b) - F(a)
\]
```

Variable and Namespace Management

Overview: Manage variables and namespaces effectively within Jupyter Notebooks to maintain a clean and organized workspace.

Viewing Variables

- **%who:** Lists all variables in the current namespace.

  ```
  a = 10
  b = "Hello"
  %who
  ```

- **%whos:** Provides a detailed list of variables with information about their type and value.

  ```
  %whos
  ```

Clearing Variables

- **%reset:** Clears all variables from the namespace.

```
%reset
```

- **%reset -f:** Force reset without confirmation.

```
%reset -f
```

- **del Keyword:** Delete specific variables.

```
del a
del b
```

Storing and Loading Variables

- **%store:** Save variables for use in other notebooks.

```
a = 100
%store a
```

- **%store -r:** Restore stored variables.

```
%store -r a
print(a)  # Outputs: 100
```

Extended Examples

```
# Define variables
x = 42
y = [1, 2, 3]
z = {"key": "value"}

# List variables
%who
%whos

# Delete a variable
del x
%whos
```

```
# Reset the namespace
%reset -f
%whos
```

Visualization and Interactive Output

Overview: Create visualizations and interactive outputs within Jupyter Notebooks to enhance data analysis and presentation.

Matplotlib

A widely-used plotting library for creating static, animated, and interactive visualizations.

```
import matplotlib.pyplot as plt
import numpy as np

# Sample Data
x = np.linspace(0, 10, 100)
y = np.sin(x)

# Create Plot
plt.plot(x, y, label='Sine Wave')
plt.title('Sine Function')
plt.xlabel('x')
plt.ylabel('sin(x)')
plt.legend()
plt.grid(True)
plt.show()
```

Seaborn

A statistical data visualization library based on Matplotlib.

```
import seaborn as sns
import matplotlib.pyplot as plt

# Load Sample Dataset
tips = sns.load_dataset("tips")

# Create a Scatter Plot
sns.scatterplot(data=tips, x="total_bill", y="tip", hue="day")
plt.title('Total Bill vs Tip')
plt.show()
```

Plotly

An interactive graphing library.

```
import plotly.express as px

# Load Sample Dataset
df = px.data.iris()

# Create Interactive Scatter Plot
fig = px.scatter(df, x='sepal_width', y='sepal_length', color='species',
                 title='Iris Sepal Dimensions')
fig.show()
```

Interactive Widgets

Use `ipywidgets` to add interactive controls to your notebooks.

```
import ipywidgets as widgets
from IPython.display import display

# Define a Slider Widget
slider = widgets.IntSlider(value=5, min=0, max=10, step=1, description='Value:')

# Display the Widget
display(slider)

# Define a Function to Respond to Slider Changes
def on_change(change):
    if change['type'] == 'change' and change['name'] == 'value':
        print(f"Slider value changed to {change['new']}")

# Observe the Slider
slider.observe(on_change)
```

Extended Examples

Combining Matplotlib and Widgets

```
import matplotlib.pyplot as plt
import ipywidgets as widgets
from IPython.display import display

# Sample Data
x = range(0, 10)
y = [i**2 for i in x]

# Define a Slider for the y-axis limit
slider = widgets.IntSlider(value=100, min=10, max=200, step=10, description='Y-
Limit:')
```

```
# Define Update Function
def update_plot(y_limit):
    plt.figure(figsize=(8, 5))
    plt.plot(x, y, marker='o')
    plt.ylim(0, y_limit)
    plt.title('Square Numbers')
    plt.xlabel('x')
    plt.ylabel('y')
    plt.grid(True)
    plt.show()

# Use Interactive Widget
widgets.interact(update_plot, y_limit=slider)
```

Exporting and Sharing Notebooks

Overview: Share your Jupyter Notebooks with others by exporting them in various formats or hosting them online.

Exporting Notebooks

1. **Via Jupyter Interface:**
    - Navigate to **File** > **Download as** and choose the desired format:
        - **Notebook (.ipynb)**
        - **HTML (.html)**
        - **PDF via LaTeX (.pdf)**
        - **Markdown (.md)**
        - **ReStructured Text (.rst)**
        - **Executable Script (.py)**
2. **Using nbconvert:**

```
jupyter nbconvert --to html your_notebook.ipynb
jupyter nbconvert --to pdf your_notebook.ipynb
```

Sharing Notebooks

- **GitHub:**
    - Upload your `.ipynb` files to a GitHub repository. GitHub renders notebooks natively.
- **Nbviewer:**
    - Use Nbviewer to share notebooks via a URL.
    - Example: `https://nbviewer.jupyter.org/github/username/repo/blob/main/notebook.ipynb`

- **Google Colab:**
  - Upload your notebook to Google Drive and open it with Google Colab for sharing and collaboration.
- **Binder:**
  - Create a shareable environment that launches your notebook directly in the browser.
  - Example: Binder

Extended Examples

Exporting to HTML

```
jupyter nbconvert --to html my_notebook.ipynb
```

Exporting to PDF

Ensure you have LaTeX installed for PDF conversion.

```
jupyter nbconvert --to pdf my_notebook.ipynb
```

Sharing via GitHub

1. **Initialize Git Repository:**

   ```
   git init
   git add my_notebook.ipynb
   git commit -m "Add Jupyter Notebook"
   ```

2. **Push to GitHub:**

   ```
   git remote add origin https://github.com/username/repo.git
   git push -u origin master
   ```

3. **View on GitHub:** Navigate to your repository on GitHub to view the rendered notebook.

Extensions and Customization

Overview: Enhance Jupyter Notebooks with extensions and customize the interface to suit your workflow.

Installing Nbextensions

`jupyter_contrib_nbextensions` provides a collection of community-contributed extensions.

1. **Install Nbextensions:**

```
pip install jupyter_contrib_nbextensions
jupyter contrib nbextension install --user
```

2. **Enable Extensions:**
    - Open Jupyter Notebook.
    - Navigate to the **Nbextensions** tab.
    - Browse and enable desired extensions, such as:
        - **Table of Contents:** Adds a table of contents sidebar.
        - **Codefolding:** Allows folding of code sections.
        - **Spellchecker:** Checks spelling in Markdown cells.
        - **Variable Inspector:** Displays all variables and their values.

Popular Extensions

- **Table of Contents (2):** Automatically generates a table of contents.
- **Snippets Menu:** Provides reusable code snippets.
- **ExecuteTime:** Displays the time a cell was run and the duration.
- **Hide Input:** Allows hiding of code input cells while showing outputs.

Customizing Jupyter Themes

Use `jupyterthemes` to customize the appearance of your Jupyter Notebook.

1. **Install jupyterthemes:**

```
pip install jupyterthemes
```

2. **List Available Themes:**

```
jt -l
```

3. **Apply a Theme:**

```
jt -t chesterish
```

4. **Reset to Default:**

```
jt -r
```

Keyboard Shortcut Extensions

Enhance keyboard shortcuts using extensions like `jupyter-keymap`.

Advanced Customization

- **Custom CSS and JavaScript:**
  - Modify the `custom.css` and `custom.js` files located in the Jupyter configuration directory to apply custom styles or scripts.

```
~/.jupyter/custom/custom.css
~/.jupyter/custom/custom.js
```

- **Creating Custom Extensions:**
  - Develop your own Jupyter extensions by following the Jupyter Notebook Extensions Documentation.

Extended Examples

Enabling Table of Contents Extension

1. **Install and Enable:**

```
pip install jupyter_contrib_nbextensions
jupyter contrib nbextension install --user
jupyter nbextension enable toc2/main
```

2. **Use in Notebook:**
   - Open a notebook.
   - Navigate to the **Nbextensions** tab.
   - Enable **Table of Contents (2)**.
   - A sidebar with the table of contents will appear, updating as you add headings.

Best Practices

Overview: Follow best practices to write clean, efficient, and maintainable Jupyter Notebooks.

Organize Your Notebook

- **Section Headers:** Use Markdown headings to divide your notebook into sections.
- **Consistent Structure:** Maintain a logical flow from data loading to analysis and conclusions.

Write Clear and Concise Code

- **Use Comments:** Explain non-obvious parts of your code.

```
# Calculate the mean of the dataset
mean_value = np.mean(data)
```

- **Avoid Long Cells:** Break down code into smaller, manageable cells.

Document Your Work

- **Markdown Cells:** Use Markdown to explain your analysis, findings, and methodologies.
- **Docstrings:** Document functions and classes with docstrings.

```
def calculate_mean(data):
    """
    Calculate the mean of a list of numbers.

    Parameters:
    data (list): A list of numerical values.

    Returns:
    float: The mean of the data.
    """
    return sum(data) / len(data)
```

Version Control

- **Use Git:** Track changes and collaborate using Git.
- **.gitignore:** Exclude unnecessary files like checkpoints.

```
# Jupyter Notebook checkpoints
.ipynb_checkpoints/
```

Keep Notebooks Reproducible

- **Seed Random Number Generators:** Ensure reproducible results.

```
import numpy as np
np.random.seed(42)
```

- **Document Dependencies:** List required libraries and versions.

```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
```

Avoid Hard-Coding Paths

- **Use Relative Paths:** Ensure notebooks are portable.

```
data = pd.read_csv('data/dataset.csv')
```

Clean Up Before Sharing

- **Clear Outputs:** Remove unnecessary outputs to reduce file size.
  - Navigate to **Cell** > **All Output** > **Clear**.
- **Hide Code (Optional):** Use extensions to hide code cells for presentations.

Tips for Beginners

Overview: Enhance your learning experience with these essential tips for new Jupyter Notebook users.

Start with Tutorials

- **Official Jupyter Documentation:** Jupyter Docs
- **Interactive Tutorials:** Platforms like DataCamp and Coursera offer Jupyter-focused courses.

Learn Markdown

- **Master Markdown Syntax:** It enhances the readability and professionalism of your notebooks.
- **Use Resources:** Markdown Guide provides comprehensive tutorials.

Utilize Magic Commands

- **Boost Productivity:** Learn and use magic commands to perform common tasks efficiently.
- **Explore Help:** Use `%lsmagic` to list all available magic commands.

```
%lsmagic
```

Experiment with Widgets

- **Interactive Controls:** Enhance your notebooks with sliders, buttons, and other interactive elements using `ipywidgets`.
- **Learn by Doing:** Try creating simple interactive plots or forms.

Regularly Save and Backup

- **Prevent Data Loss:** Save your work frequently and consider using cloud storage or version control for backups.

Explore Extensions

- **Enhance Functionality:** Use nbextensions to add features like spell checking, table of contents, and more.
- **Caution:** Only enable extensions you understand to prevent conflicts.

Practice Good Documentation

- **Explain Your Steps:** Clearly document your analysis steps to make your notebooks understandable to others (and your future self).

Engage with the Community

- **Forums and Q&A:** Participate in communities like Stack Overflow and Jupyter Discourse to seek help and share knowledge.
- **GitHub Repositories:** Explore public repositories to learn from others' notebooks.

Keep Learning

- **Stay Updated:** Jupyter is actively developed. Keep an eye on updates and new features.
- **Advanced Features:** As you grow comfortable, explore advanced features like custom extensions, integration with other tools, and more.

Additional Resources

Overview: Access a curated list of resources to deepen your understanding of Jupyter Notebooks.

Official Documentation

- **Jupyter Documentation:** https://jupyter.org/documentation
- **Jupyter Notebook Docs:** https://jupyter-notebook.readthedocs.io/en/stable/

Tutorials and Guides

- **Real Python:** Jupyter Notebook Tutorial
- **DataCamp:** Introduction to Jupyter Notebooks
- **Coursera:** Applied Data Science with Python Specialization

Books

- **"Jupyter Cookbook" by Dan Toomey:** Practical recipes for using Jupyter effectively.
- **"Python Data Science Handbook" by Jake VanderPlas:** Comprehensive guide with Jupyter examples.

Online Communities

- **Stack Overflow:** Jupyter Notebook Questions
- **Jupyter Discourse:** https://discourse.jupyter.org/

- **Reddit:** r/Jupyter

Extensions and Tools

- **Nbextensions:** https://jupyter-contrib-nbextensions.readthedocs.io/en/latest/
- **JupyterLab:** The next-generation interface for Project Jupyter.
  - **Website:** https://jupyterlab.readthedocs.io/en/stable/
  - **Installation:**

```
pip install jupyterlab
jupyter lab
```

Videos and Webinars

- **YouTube Tutorials:** Search for "Jupyter Notebook tutorials" for video guides.
- **Official Jupyter YouTube Channel:** https://www.youtube.com/c/JupyterProject

Cheat Sheets

- **Jupyter Notebook Keyboard Shortcuts:** https://jupyter-notebook.readthedocs.io/en/stable/_static/shortcuts.pdf
- **Markdown Cheat Sheet:** https://www.markdownguide.org/cheat-sheet/

Happy Coding!

Leverage this cheat sheet to streamline your Jupyter Notebook experience. Practice regularly, explore new features, and engage with the community to become proficient in using Jupyter for your projects.