#_ the Jupyter Notebook Cheat Sheet

1. Keyboard Shortcuts:

- **General:**
  - **Shift + Enter**: Run the current cell.
  - **Alt + Enter**: Run current cell and insert a new one below.
  - **Ctrl + S**: Save the notebook.
- **Cell Editing:**
  - **Enter**: Edit a cell.
  - **Esc**: Exit cell editing.
  - **A**: Insert cell above.
  - **B**: Insert cell below.
  - **D, D**: Delete current cell.
  - **Z**: Undo cell deletion.
  - **C**: Copy cell.
  - **X**: Cut cell.
  - **V**: Paste cell below.
  - **Shift + V**: Paste cell above.
  - **Shift + M**: Merge multiple selected cells.
  - **I, I**: Interrupt kernel.
  - **0, 0**: Restart kernel.
- **Cell Type:**
  - **Y**: Change to Code.
  - **M**: Change to Markdown.
  - **R**: Change to Raw.
- **Navigation:**
  - **Ctrl + Shift + -**: Split cell at cursor.
  - **Shift + Space**: Scroll notebook up.
  - **Space**: Scroll notebook down.

---

2. Markdown Basics:

- **Headers**: `#` for H1, `##` for H2, etc.
- **Bold Text**: `**text**` or `__text__`

- **Italic Text**: *text* or _text_
- **Hyperlink**: [Link Text](URL)
- **Ordered List**: Starting lines with numbers.
- **Unordered List**: Using * or - followed by a space.
- **Code in Markdown**: Enclosed with `
- **Block of Code**: Enclosed with triple backticks ```

---

3. Magic Commands:

- **%run**: Execute Python script.
- **%load**: Load Python script into cell.
- **%time**: Time execution of a statement.
- **%timeit**: Time a statement with multiple runs.
- **%who**: List variables in namespace.
- **%history**: Show command input history.
- **%pwd**: Current directory.
- **%ls**: List directory content.
- **%matplotlib inline**: Displays Matplotlib plot outputs inline within the frontends.
- **%cd**: Change the current working directory.

---

4. Tips & Tricks:

- **Shift + Tab**: Tool-tip with function signature and docstring.
- **?** after a function: Help for that function.
- **!**: Execute system shell commands.
- **%%bash**: Run cell in Bash mode.
- **{}**: Embed variable in shell command, e.g., !echo {variable_name}
- **Tab Completion**: Tab to autocomplete variables, functions, etc.
- **Function Docstrings**: Shift + Tab on a function name to see its docstring.
- **Multicursor support**: Holding Alt and dragging the mouse.

By: Waleed Mousa

5. Performance:

- **Profiler**: Use `%prun` to see how time is spent in a function.
- **Debugger**: Use `%debug` after an exception to step into the Python debugger.
- **Memory**: Use `%memit` (needs the `memory_profiler` extension) to measure memory use in a cell.

---

6. Display & Widgets:

- **`%matplotlib inline`**: Display plots in the notebook.
- **`from IPython.display import display, Image, SVG, Math, YouTubeVideo`**: Display various formats in your notebook.
- **`import ipywidgets as widgets`**: Create interactive widgets.

---

7. Jupyter Ecosystem:

- **JupyterLab**: Advanced interface with more features.
- **JupyterHub**: Multi-user Jupyter for teams.
- **Voilà**: Turn notebooks into standalone web apps.
- **Binder**: Share live, interactive versions of your notebooks.

---

8. Extensions & Widgets:

- **Jupyter-contrib**: Collection of extensions.
- **Jupyter-widgets**: Provides interactive widgets for the notebook.
- **Qgrid**: Widget for manipulating DataFrames.
- **Rise**: Turn Jupyter notebooks into slideshows.
- **`!pip install jupyterthemes`**: To customize or theme your notebooks.
- **`!jt -l`**: List available themes.
- **`!jt -t THEME_NAME`**: Set a theme.

9. Plotting & Visualization:

- **Seaborn**: Advanced statistical plots.
- **Plotly**: For interactive plots.
- **Bokeh**: Another interactive plotting library.
- `%matplotlib inline`: For inline display of plots.
- `import matplotlib.pyplot as plt`: Standard way to import the plotting library.
- `plt.plot(x, y)`: Plot y versus x as lines.
- `plt.xlabel('Name')`: Label x-axis.
- `plt.ylabel('Name')`: Label y-axis.
- `plt.title('Title')`: Set a title.

---

10. Exporting & Conversion:

- **To HTML**: Exporting the notebook as an HTML file.
- **To PDF**: Exporting the notebook as a PDF.
- **To Markdown**: Conversion to a markdown file.
- **To Python (.py)**: Converts the notebook to a standard Python script.
- **Jupyter nbconvert**: Command-line tool to convert notebooks.
- `jupyter nbconvert --to FORMAT notebook_name.ipynb`: Convert notebook to different formats (like HTML, PDF, Slides).
- `!jupyter nbconvert --to pdf MyNotebook.ipynb`: Convert notebook to PDF directly from a cell.

---

11. Advanced Features:

- **Interactive Outputs**: Widgets and interactive plots.
- **Profiling Code**: Using `%prun` for performance profiling.
- **LaTeX in Markdown**: For displaying mathematical symbols and equations.
- **Big Data Integration**: Using Dask or Vaex for large datasets.
- **Git Integration**: Integrating with Git for version control.

12. Troubleshooting & Debugging:

- **Clear Outputs**: Clears the output display.
- **Check For Updates**: Regularly update for new features and security patches.
- **Debugger**: `%debug` magic command after an exception is raised.
- **Logging**: Integrating Python logging module.

---

13. Extensions for Collaboration:

- **JupyterHub**: Multi-user version of the notebook.
- **Binder**: Turns notebooks into interactive web apps.
- **Google Colab**: Google's free cloud service based on Jupyter.
- **nbdime**: Tool for diffing and merging notebooks.

---

14. Security:

- **Token**: Secure way Jupyter ensures browser-to-server communication.
- **SSL**: Setting up SSL for encrypted communication.
- **Server Password**: Setting a password for the notebook server.

---

15. Customization & Configuration:

- **Startup Files**: Executing scripts upon kernel startup.
- **Custom Themes**: Using Jupyter themes for aesthetics.
- **Extensions Configuration**: Customizing and toggling extensions.
- **Keyboard Shortcuts**: Customizing and adding new shortcuts.

---

16. Integration with Other Tools:

- **Pandas**: For data manipulation.
- **Numpy**: For numerical operations.
- **Scikit-learn**: For machine learning.

- **TensorFlow and PyTorch**: For deep learning.
- **SQL Integration**: Magic command `%sql` for inline SQL commands.

---

17. Best Practices:

- **Regular Saves**: To prevent data loss.
- **Version Control**: Using Git for tracking changes.
- **Modular Code**: Keeping the notebook organized.
- **Comments**: Adequately commenting for clarity.
- **Clean Outputs Before Save**: Especially if sharing notebooks.