

Week1_2 Programming Platform Set-up

This is the first lab in a series of Python tutorials for the CRP 4680/5680.

The goals for today's lab are (1) install Python and Jupyter Notebook and Jupyter Lab on your computer; (2) get familiar with the keyboard shortcuts of Jupyter Notebook and successfully run the basic code and narrative text on it.

Part 1. Installing Python using Anaconda

1. Download Anaconda - Click on [this link](#).
Versions are available for Windows, Mac, and Linux with options for both Anaconda and Miniconda. Anaconda comes with a large collection of pre-installed packages and tools but requires at least 3GB of storage space. Miniconda provides a minimal environment with Conda for package. To ensure that all subsequent packages function properly, I recommend that everyone install the full version of Anaconda. From now on, please reserve at least 25GB of space on your computer to install additional Python packages.
2. Choose the version that matches your computer's operating system. Note: if you are a Mac user, and are unsure whether your computer uses an Apple Silicon or Intel Chip, please follow these steps:
 - Open the Apple menu in the top left corner of your screen.
 - Select "About This Mac."
 - If your Mac has Apple Silicon, the chip name will be listed as Apple M1, M2, etc.
 - Otherwise, it uses an Intel CPU.
3. Choose the graphical installer, as it comes with a user interface that is intuitive and easy to use.



Windows

Python 3.12

📄 64-Bit Graphical Installer (912.3M)



Mac

Python 3.12

📄 64-Bit (Apple silicon) Graphical Installer (704.7M)

📄 64-Bit (Apple silicon) Command Line Installer (707.3M)

📄 64-Bit (Intel chip) Graphical Installer (734.7M)

📄 64-Bit (Intel chip) Command Line Installer (731.2M)



Linux

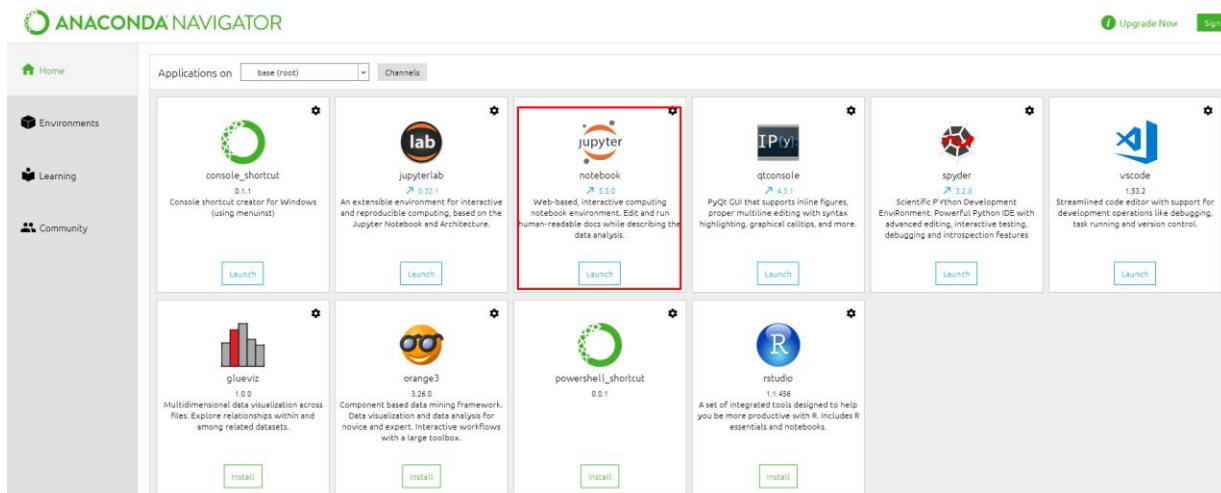
Python 3.12

📄 64-Bit (x86) Installer (1007.9M)


📄 64-Bit (AWS Graviton2 / ARM64) Installer (800.6M)

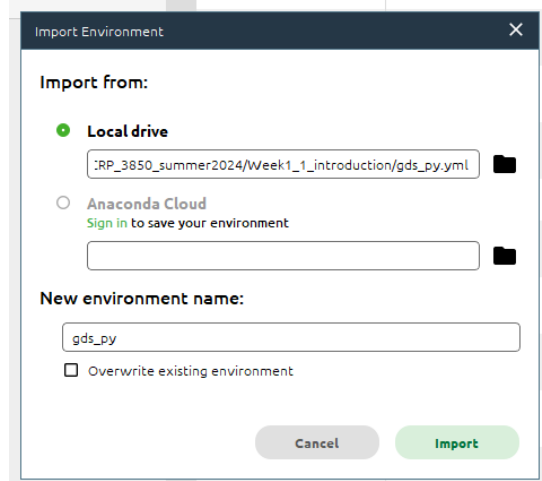
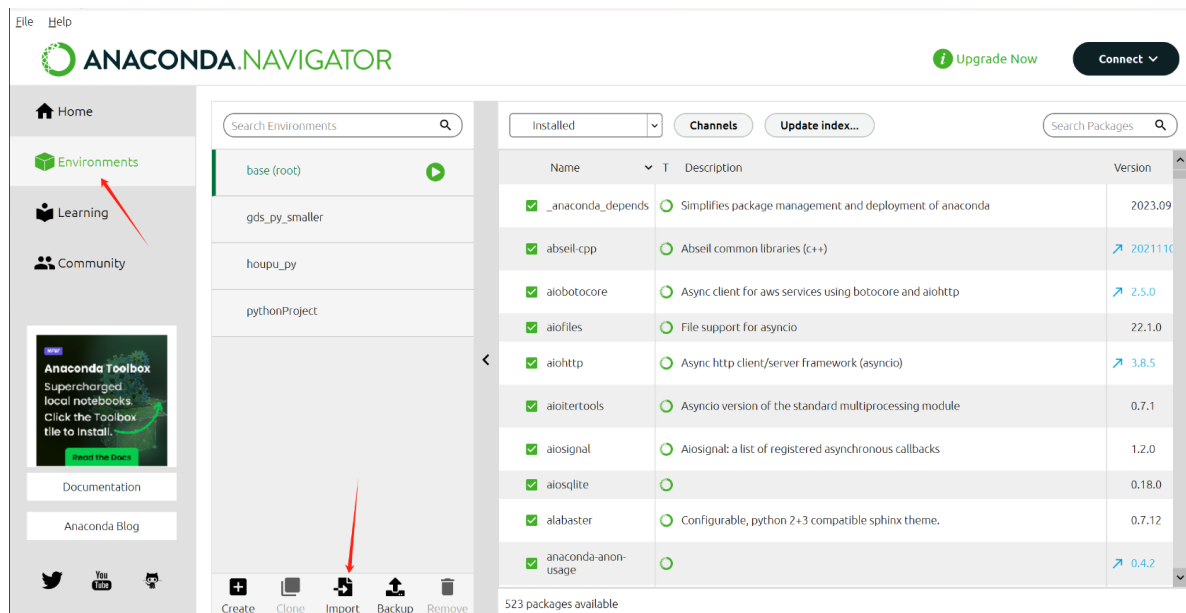
📄 64-bit (Linux on IBM Z & LinuxONE) Installer (425.8M)

4. Install the Anaconda - follow the default installation options to install Anaconda. Once the installation is complete, open Anaconda and verify that you have the Jupyter Lab and Notebook Apps installed properly.



Part 2. Setting up your computer environment

0. Create a local folder/directory for this course on your computer and name it something like CRP_5680 or CRP5680. Make sure the folder name does not contain spaces from now on. Within this folder, create subfolders to organize your course materials by week or topic (e.g., week1_introduction).
1. In the left-hand menu of Anaconda, click *Environment*  **Environments**. You may find that Anaconda has a set of packages (libraries) pre-installed for us (e.g., *Pandas*). These libraries are stored in the original *Base environment*. This is the default environment that contains the core Python installation and pre-installed packages, which serves the foundation for creating and managing additional environments.
2. This course requires many additional libraries (e.g., *Geopandas* and *Scikit-learn*) that have not been installed yet. We will create a new virtual environment with these libraries installed and specialized for this course. A virtual environment helps manage dependencies by keeping those required for different projects separate, which is useful for avoiding conflicts between packages.
3. From Canvas, download the *gds_py.yml* available under the Week1 folder. Download it into your course folder. This is a configuration file that defines the dependencies and settings needed to create a specific conda environment tailored for this course.
4. To import this virtual environment, click the *Environment* button and click *Import* (see the Figure below). Locate the path to your *gds_py* file and proceed with the import. The installation may take anywhere from 5 minutes to 40 minutes, so please be patient. After the installation is complete, you will see that a new environment named *gds_py* has been created, with 521 Python packages successfully installed.



Part 3. Opening Jupyter Notebook via terminal

Next, let's open our coding platform: Jupyter Notebook. It is an interactive coding application for Python. It is a platform where you can run and edit code, display output, and add explanations that make your work more understandable and sharable. Let us start the Jupyter Notebook now!

Scenario 1: If you start Jupyter Notebook on Windows with a Course Folder on the C Drive or desktop, following these steps:

1. Open the Command interface:
 - open the Anaconda Prompt. You can find this by searching for “**Anaconda Prompt**” in the Start menu.
 - The initial path in the Anaconda Prompt typically looks like this:

(base) C:\Users\[YourUserName]>

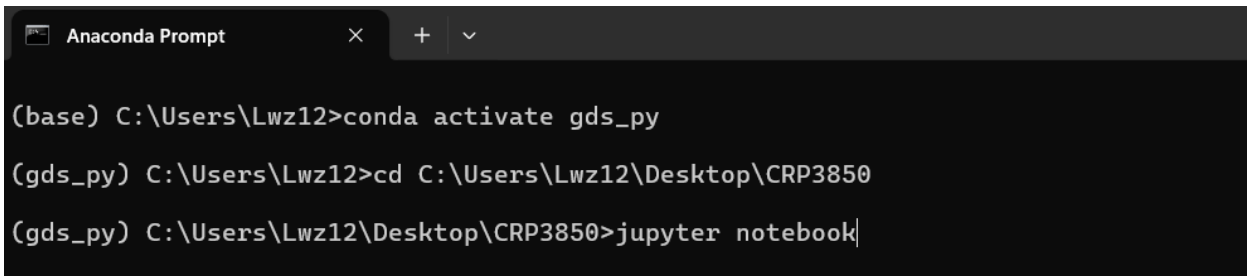
2. Activate your Python environment: Type ***conda activate gds_py*** and press enter. Check that your prompt looks like this:

(gds_py) C:\Users\[UserName]>

3. Next, we navigate to your course folder.
 - Type ***cd*** in the command prompt. ***cd*** means change directory.
 - Next, drag the CRP5680 folder you just created into the Anaconda prompt. This action will automatically insert the folder's full path. Press **Enter** to navigate to the folder.

4. Launch Jupyter notebook by typing: ***jupyter lab***. Press Enter.

The complete code is as follows:



```
Anaconda Prompt
(base) C:\Users\Lwz12>conda activate gds_py
(gds_py) C:\Users\Lwz12>cd C:\Users\Lwz12\Desktop\CRP3850
(gds_py) C:\Users\Lwz12\Desktop\CRP3850>jupyter notebook
```

You are all set! Be sure to remember these activation commands! The packages you downloaded are accessible only through the virtual environment. You will need to activate the ***gds_py*** environment each time you work on this course.

Scenario 2: For macOS with a Course Folder in the User's Home Directory or Desktop:

1. On macOS, open the Terminal application. You can find it by search for “Terminal” using “cmd + space”.
2. Activate your Python environment by typing: ***conda activate gds_py*** (see the figure below).
3. Navigate to Your Course Folder:
 - a) by typing ***cd***
 - b) and next, drag the CRP5680 folder you just created into the Anaconda prompt. This action will automatically insert the folder's full path. Press **Enter** to navigate to the folder.
4. Launch Jupyter Notebook by typing ***jupyter lab***.

```
Last login: Thu Feb  1 11:10:35 on ttys000
[(base) qianchenyu@dhcp-vl2041-29593 ~ % conda activate gds_py
[(gds_py) qianchenyu@dhcp-vl2041-29593 ~ % cd '/Users/qianchenyu/Desktop/new files'
(gds_py) qianchenyu@dhcp-vl2041-29593 new files % jupyter lab
```

Scenario 3: Here is the code to open the Jupyter Lab if you save the course materials in another drive (D in this case). The only additional step is to switch to the desired drive by typing the drive letter followed by a colon (see the Figure below).

```
(base) C:\Users\Lwz12>conda activate gds_py
(gds_py) C:\Users\Lwz12>d:
(gds_py) D:\>cd D:\CRP_3850_summer2024
(gds_py) D:\CRP_3850_summer2024>jupyter lab
```

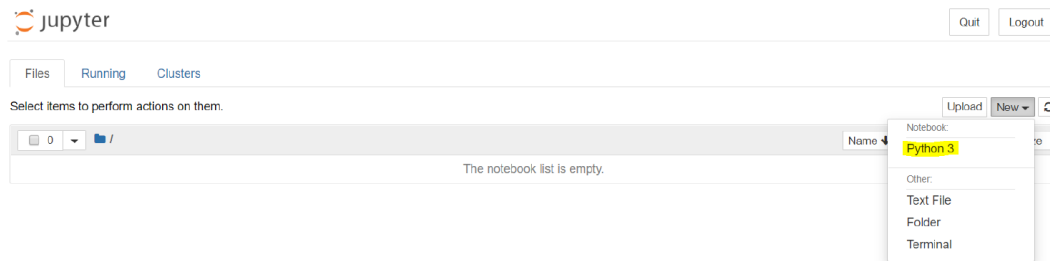
If the code does not work, please ask the TA or instructor for help.

Part 4. Jupyter Lab/Notebook


1.1 The Interface of Jupyter Lab/Notebook

You can open either Jupyter Notebook or Jupyter Lab from the command line for the purpose of this course, as both are suitable for interactive computing. They are similar tools. Jupyter Lab provides a more flexible interface with a multi-tab, multi-panel layout, whereas Jupyter Notebook offers a simpler, single-page interface dedicated to running and editing notebooks.

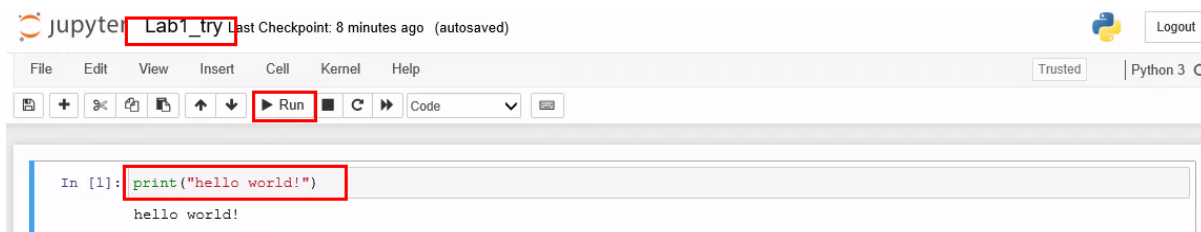
Open a Jupyter Lab/Notebook in your default browser. The dashboard will show nothing if you just create the file folder without anything stored in it. To create a new Notebook, click “File” and “New” in the upper-left/right corner and choose “Python 3” as the ipykernel. You will be navigated to a new web page, where we run Python code.



First, check out the menus to get a feel for it.

Next, give an appropriate name for your first Jupyter Notebook by clicking “Untitled” at the top of the page. If you use Jupyter lab interface, click  and give a name. You can change the name anytime you want.

In the first code chunk, type `print ("Hello world!")` and click “run ▶” (shortcut: shift+enter). If no error occurs, then congratulations! You have successfully written some code.



Click the “Save” icon on the top left. You should find a new Jupyter Notebook file (.ipynb) with the customized name in your course folder.

Please keep in mind that you should not **close the Anaconda command prompt when you are still working on the Jupyter Notebook/Lab**; otherwise, the notebook kernel will shut down, and you must resume the Anaconda Prompt to restore your working progress!

1.2 The Basics of Jupyter Notebook

When you are ready to write code, you should notice two terms: *cells* and *kernels*.

- A kernel is a “computational engine” that executes the code contained in a notebook document.
- A cell is a container for text to be displayed in the notebook or code to be executed by the notebook’s kernel.

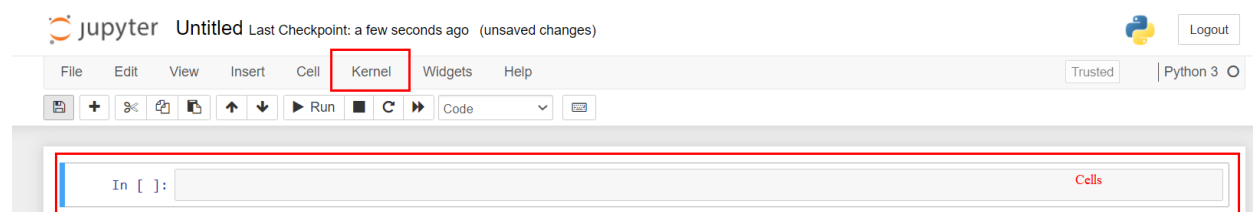


Figure.1: Cell (code chunk) and kernel

Command and Editing Modes:

- When you write code, the cell will be in the *editing mode*, used to write or edit the content of a cell. To enter an editing mode, click inside a cell or press Enter while a cell is selected.
- After running the code, the cell will return to *command mode*, used to manage and interact with cells, e.g., delete and add new cells. Press “Esc” while in editing mode or click outside the cell's content area to enter the command mode.



Figure 2: Toggle between the edit and command mode

Now, in the command mode, press “B” on the keyboard, you will get a new code cell just below the current cell. Try to press “B” multiple times to add more cells. Similarly, press “D” to delete cells.

Cell types: Code cell and Markdown cells.

- A **code cell** is where you write and execute the code. The code cell is the default.
- A **Markdown cell** is where you write and format text using Markdown.

Now, switch to editing mode, and type the following code:

```
In [3]: a, b=100, 200
        c=a+b
        print(c)

300
```

Figure 3: The code cell: for creating and editing the code

Click “run” or press “shift+enter” to run the above code.

Change one Code Cell to Markdown Cell by selecting the “Markdown” option in the dropdown menu of the toolbar or simply use the shortcut “y” (code cell) and “m” (markdown cell) to toggle between. Note: make sure they are in the command model when using the shortcuts.

```
This is the first lab in a series of Python tutorials for the Advanced GIS course. The tutorial series intended to teach the Python Beginners the basic skills of Python for quantitative spatial analysis under the context of urban planning and big data.
```

Figure 4: The Markdown cell: for description and narrative text

Write down whatever you want and run the cell. You will get the cell displaying the rendered narrative text, just like what is shown in Figure 4.

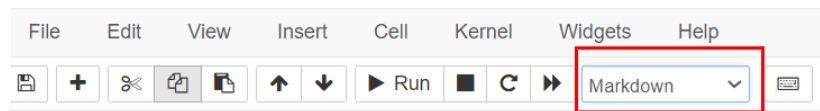


Figure 5: Transform the Code cell to the Markdown cell

Here are some examples of the Markdown text before and after rendering (after running the cell). You can find the Markdown text after rendering presents a much better look, so please use the Markdown text properly to edit your descriptive part when submitting your homework.

```
This is the first lab in a series of Python tutorials for the
Advanced GIS course. The tutorial series intended to teach the
*Python Beginners* the basic skills of Python for quantitative
spatial analysis under the context of urban planning and big data.

We will begin with the basic Python syntax, followed by the web-
scraping technique for data collection, data cleaning and
management functions, as well as the geoprocessing and
visualization.
```

Figure 6: These are the original text that you type in the Markdown cell.

This is the first lab in a series of Python tutorials for the Advanced GIS course. The tutorial series intended to teach the *Python Beginners* the **basic skills** of Python for quantitative spatial analysis under the context of urban planning and big data.

We will begin with the basic Python syntax, followed by the web-scraping technique for data collection, data cleaning and management functions, as well as the geoprocessing and visualization.

Figure 7: the narrative text rendered in the Markdown cell after running the cell. Pay attention to the usage of ****** to bold words and *** to italicize words.

Original text:

```
# Header1
some text under the header1

## Header2
some text under the header2

### Header3

1. New Year's Day
  1. Christmas Day
  2. Memorial Day
2. Independence Day
  1. Labor Day
  3. Thanksgiving Day

1. Chapter 1 <br>
  1.1 content1 <br>
  1.2 content2 <br>
2. Chapter 2 <br>
  2.1 content1 <br>

- Chapter 1
  * 1.1 content1
    - 1.1.1 content2
- Chapter 2 <br>
  + 2.1 content1
```

Figure 8: the narrative text in the Markdown cell

The rendered Markdown shows up as below. Note that the symbols -, *, and + are all interpreted as bullet points when rendered. Additionally, the
 tag is used to break the current line and continue the remaining content on a new line.

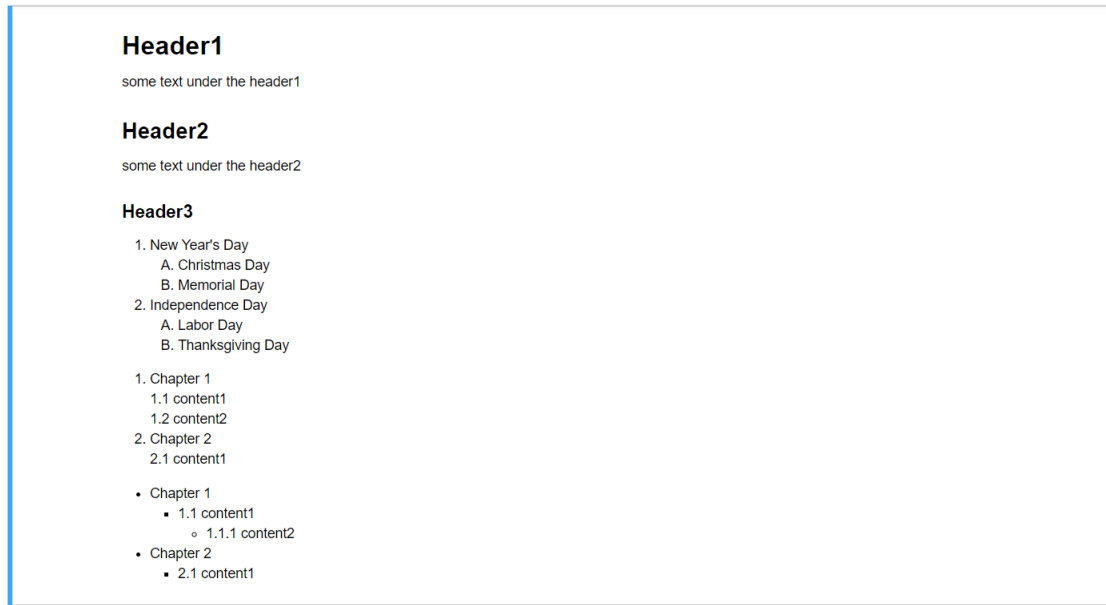


Figure 2.9: the rendered narrative text in the Markdown cell after running the cell

For more functions regarding the Jupyter Notebook Markdown, please refer to:
<https://www.youtube.com/watch?v=uVLzL5E-YBM>

For basic Markdown syntax: <https://www.markdownguide.org/basic-syntax/>

1.4 Keyboard Shortcuts

Keyboard shortcuts can make your programming process easier and more convenient. Remember that each cell can be in the edit mode when you are typing code in it, or else in the command model.

The keyboard shortcuts are different for cells in different modes.

- Toggle between edit and command mode with **Esc** and **Enter**, respectively.
- Run a cell: "**Shift + Enter**" (a new cell will be created below and becomes active); "**Ctrl + Enter**" (no new cell will be created)
- Once in **edit mode**:
 - Ctrl + c: copy the selected code
 - Ctrl + v: paste the selected code
 - Ctrl + z: undo the change.
- Once in the **command mode**:
 - B: add a cell below the highlighted cell
 - A: add a cell above the highlighted cell
 - double click "D": Delete the cell.
 - Y: change to a code cell
 - M: change to a markdown cell

holding "Shift": Select multiple cells
"Shift + m": Merge multiple cells after selecting all cells.

Go to the main menu, and click “Keyboard Shortcuts.”, you will find more default shortcuts in both the command and edit modes.

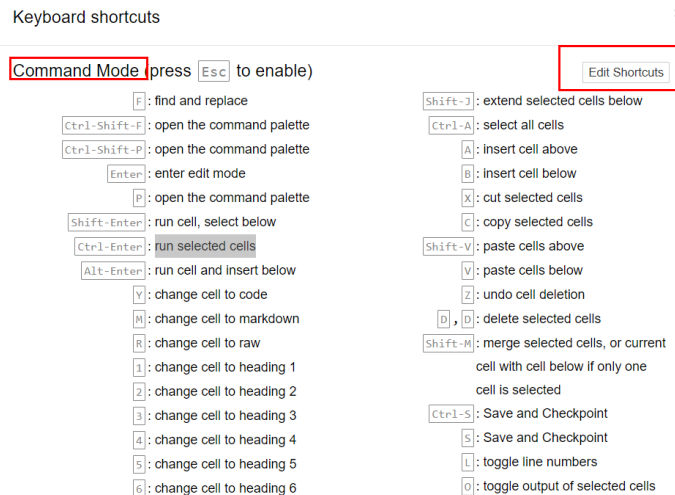


Figure 10: Shortcut panel in the Jupyter Notebook

In-class Exercise (due Jan 25 Friday at 11:59pm)

For in-class exercises, you only need to submit a html or a PDF file. To convert from a jupyter notebook (.ipynb) to an html file, follow these steps: Go to the top menu, click File, then select Export Notebook As/Download As > Export Notebook to HTML.

Exercise 1: Writing and Running Your First Code

1. Open Jupyter Notebook or Jupyter Lab in your browser.
2. Create a new notebook in the CRP5680 folder.
3. In the first cell:
 - o Write the code: `print("Welcome to CRP5680!")` and run it.
4. Save the notebook with your last Name and Week #. For example, Li_week1.ipynb.

Exercise 2: Markdown Practice

1. Next, create a Markdown cell in your new notebook.
2. Answer the following questions:
 1. Name,
 2. Field year of study,
 3. Please shortly state your goals for this course. Why do you want to take this course?
 4. Do you have any experience using ArcGIS?
 5. Have you taken any statistics courses, or do you have any relevant experience?
 6. Have you taken any Python or programming-related courses, or do you have any relevant experience?

In your answer, use the following Markdown syntax:

- Use bold and *italic* for emphasis.
- Use a bullet list.
- Optional: add a hyperlink direct to any of the related information in your answer.