Chapter 1

Lab1: Installation of Scientific Python

1.1 Introduction

In this lab, you will install the software which we will be using in phy40. This is an assignment, and will be graded. You should submit a text file containing a log of all the steps you took to install the software on your computer. Make this log as specific as possible, an entry might be:

```
Downloaded windows installer from:
 https://repo.anaconda.com/miniconda/Miniconda3-latest-Windows-x86_64.exe
```

Keeping this log will also make it easier for you to get help if you have problems.

If you run into problems, do some research on a web search tool (Google, for example) to become better informed and to see if you can overcome the problem on your own before asking for help. This is an important technique in getting help with technical problems that will serve you well even outside of this class. You will find it more easy to get useful technical help, from the sort of people most capable of offering it, when it is clear from your question that you are informed and have already tried all of the obvious things. If you are still stuck after trying to solve the problem for yourself, then contact your TA or instructor with specific technical details about what is failing, and include your installation log.

If you do find a problem with these instructions or manage to overcome a technical problem yourself, make sure to note it in your log and inform your TA, in case it is helpful for other students.

1.2 Installing Miniconda

This course is based on Python and Jupyter Lab[1]. It's possible that you already have one or both of these things installed on your computer, but we will be using "Miniconda"[2] to make an environment with specific versions of both, as well as all supporting libraries, so that everyone is on exactly the same page regardless of what type of computer they have.

Determine which OS type and version you have on the desktop or laptop computer that you will be using for your coursework. The software here will work under 64-bit versions of Windows, Linux, and macOS. It should also work on all Chromebooks released since 2019, which capable of running

[1] Jupyter Lab a newer interface that adds a bit more functionality to Jupyter Notebooks. We'll use the terms "Jupyter Lab" and "Jupyter Notebook" interchangeably

[2] Miniconda is a lightweight version of Conda. If you already have Conda, that's fine, too.

a Linux environment. If you have an older Chromebook or a 32-bit OS, please see the instructors to determine an alternate solution.

Once you have determined your OS type, go to

`https://www.anaconda.com/download/success`

and follow the appropriate instructions for your OS below.

1.2.1 Installing Miniconda under Windows

Download and run the "Windows→64 Bit Graphical Installer" file. Accept all defaults, then proceed to Section 1.3.

1.2.2 Installing Miniconda under MacOS

Go do the download page and download the graphical installer corresponding to your processor. If you're not sure what type of processor you have, click "Apple→About this Mac". If the chip says "Apple Silicon M1/2/3", then select "Apple silicon". Otherwise, select "Intel".

Click on the downloaded package installer and select all defaults.

Proceed to Section 1.3.

1.2.3 Installing Miniconda under Linux

If you believe you already already have a version of conda installed (such as miniconda or ananconda) , check by running

```
conda --version
```

If you see something like:

```
conda 4.9.2
```

as output (even if the version is different) then you do indeed already have conda installed, with the base environment activated, and you can skip ahead to Section 1.3. If instead you get a message like:

```
conda: command not found
```

then download the appropriate command line installer. If you're not sure what type of processor you have, open a terminal window and type:

```
lscpu
```

The first line will tell you what the architecture is. If it's "x86_64", then download the "64-bit (x86) Installer". Otherwise, if it's "aarch64", then download the file "64-bit (AWS Graviton2/ARM64) Installer".

Open a terminal window and navigate to where you downloaded the file (usually "Downloads"). You'll need to make it executable and execute it by typing

```
chmod +x Miniconda3-latest-Linux-(architecture).sh
./Miniconda3-latest-Linux-(architecture).sh
```

Where "(architecture)" is the version you downloaded. This will start the installation process. Accept all defaults.

Proceed to Section 1.3.

1.2.4 Installing Miniconda on a Chromebook

You will need to activate Linux on your Chromebook, according to the instructions here
`https://www.codecademy.com/articles/programming-locally-on-chromebook`

Then follow the instructions for installing under Linux in Section 1.2.3.

1.3 Installing the Physics 40 Conda Environment

We'll be setting up our conda environment and starting our Jupyter notebooks from a terminal window.

- On Windows, use the special anaconda window by going to "Start→All apps→Anaconda (miniconda3)→Anaconda prompt".
- On MacOS or Linux, just enter "terminal" in the search bar. Note that if you had a terminal open when you installed miniconda, you'll need to close it and restart it. Make sure to add it to the dock (MacOS) or favorites (Linux).

All subsequent commands will be typed in the terminal window.

Make sure your conda is fully up to date with:

```
conda update conda
```

Then follow the prompts, e.g. selecting "y" as needed to update any out-of-date packages.

We'll be using a conda environment specifically for phy40 to avoid conflicts with any other projects on your computer and to ensure that we all have the same software installed. To create our environment, type:

```
conda create -n phy40 python=3.13 numpy scipy matplotlib ipython jupyter
```

Answer "y" when prompted. This command creates an environment called "phy40" and installs the libraries "numpy", "scipy", "matplotlib", "ipython", and "jupyter". If needed, additional libaries can be installed from within the environment.

It's a good idea to create a directory to save all your lab work. You can put this directory anywhere you want, but for the moment, we'll assume it's right under your home directory, so in your terminal window, type

```
mkdir phy40
```

1.4 Starting Jupyter Lab

You'll notice that after you install miniconda on MacOS or Linux systems, "(base)" will start appearing before each command prompt, which is somewhat annoying. To get rid of this behavior, type

```
conda config --set auto_activate false
```

On Windows, the "(base)" prefix will only appear on an anaconda command window, and there is no way to remove it.

This course will make extensive use of the Jupyter Lab interface to Scientific Python, which is well suited to academic work (including independent research) because it combines code with

output in digestable chunks. Even when the end product is a polished piece of software, much of the initial code development can be done in the interactive session that Jupyter Lab provides.

Open you command window and type

```
cd phy40
conda activate phy40
```

This will put you in your working directory and activate the "phy40" environment you just created. You should see "(phy40)" tacked onto the beginning of each command line after that.

Note that any time you open a new terminal for a lab, you will need to go to the correct directory and activate the phy40 environment!

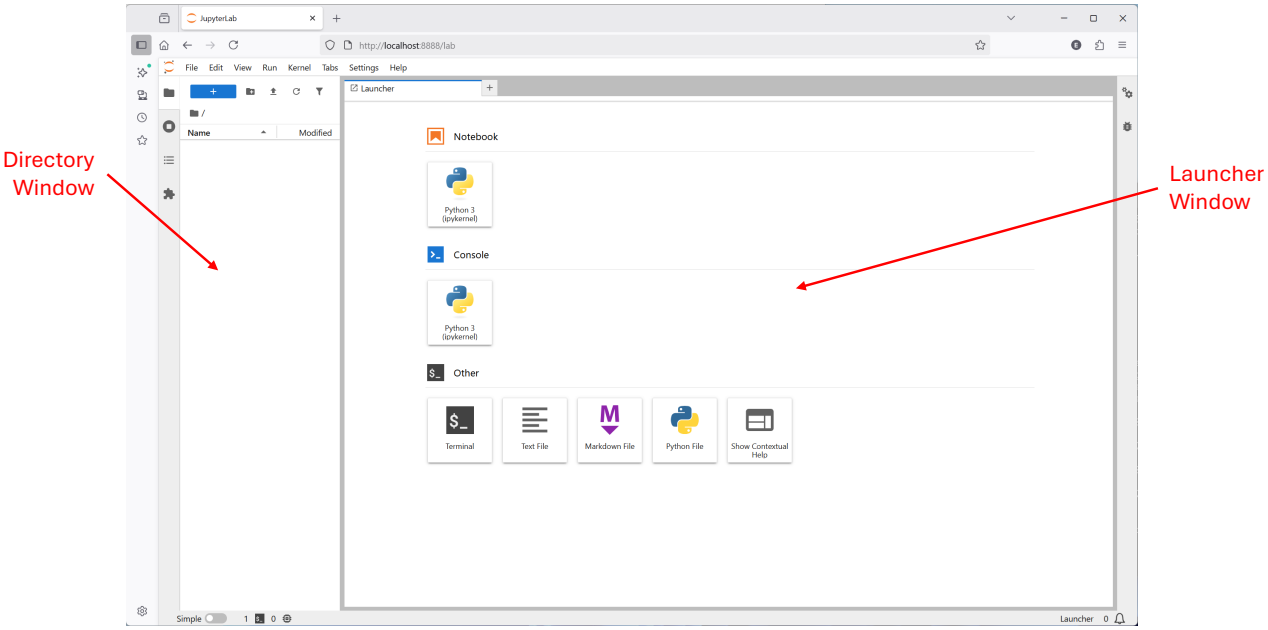

Figure 1.1: Initial appearance of the Jupyter Lab interface. The major difference with the older Jupyter Notebook interface is the navigation window at left.

Launch Jupyter Lab with:

```
jupyter lab
```

This should start the Jupyter Lab server and open a client in your default web browser which looks like Figure 1.1. The navigation pane at the left is the major difference between Jupyter Lab and the older Jupyter Notebook client. It shows the directory where you launched the client. It's empty at the moment because it's a new directory. You can create subdirectories, but Jupyter security features prevent you from navigating out of of the directory tree from which you launched the client.

1.5 Your First Notebook

You should create one Jupyter Notebook per lab assignment, by choosing the New (Python 3) option in your client. You will now see a file appear at the left called "Untitled.ipynb". You can

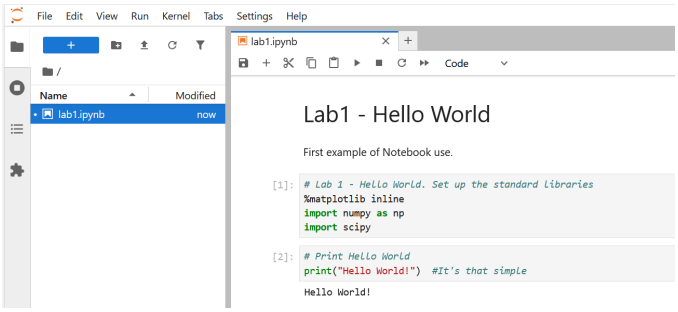

Figure 1.2: The Hello World example Jupyter Notebook.

change the name by right-clicking either on the file name or the name at the top of the tab. Change it to "lab1.ipynb".

Each assignment will consist of a number of steps, clearly numbered like this one, your first step:

△ **Jupyter Notebook Exercise 1.1:** Print "hello world" using the python print command.

Jupyter Notebooks are divided into "cells", which can be executed sequentially or individually in any order. The three types of cells are:

- **Code (default):** Executable Python code
- **Markdown:** A powerful markup language, used for commentary, documentation, etc.
- **Raw:** Raw text.

We're going to put some comments in the first cell. Type

```
#Lab1 - Hello World
First example of notebook use
```

Select this cell, use the pull down menu above to change the type from "Code" to "Markdown", and then click the little "play" (triangle) to execute the cell. You should see that the using the "#" symbol in your markup caused the first line to be rendered in a large font, as shown in Figure 1.2.

Now use the "+" icon to create two more cells and populate them as shown in the figure. The first cell sets up the "matplotlib" plotting library two work inline in the notebook, while the next two lines import the "numpy" and "scipy" libraries. We don't actually need these libraries for this lab, but it's good boiler plate for the future.

The second cell prints "Hello World!". Note that it's good practice to comment your code.

Execute the cells by selecting them and click the "play" icon.

Jupyter Notebook checkpoints your work automatically, but you can explicitly save it by clicking the "save" icon. Do this, and then right click on the file name at left and select "Open with...→ Editor". This shows you what the actual source of the Python notebook looks like, as shown in Figure 1.3 The Jupyter client translates this into the readable format you see.

1.6 Saving your Work

To make your grader's life easier, you will be submitting PDF versions of your notebook, once all of the tasks are completed and the output is visible. There are several ways to make a PDF file from

```
{
 "cells": [
  {
   "cell_type": "markdown",
   "id": "9608ece4-f404-4aa0-8794-b90a8e7ad814",
   "metadata": {},
   "source": [
    "# Lab1 - Hello World\n",
    "\n",
    "First example of Notebook use."
   ]
  },
  {
   "cell_type": "code",
   "execution_count": 1,
   "id": "75256df1-9e70-4ddb-ae11-968b7b7e10d2",
   "metadata": {},
   "outputs": [],
   "source": [
    "# Lab 1 - Hello World. Set up the standard libraries\n",
    "%matplotlib inline\n",
    "import numpy as np\n",
    "import scipy"
   ]
  },
  {
   "cell_type": "code",
   "execution_count": 2,
   "id": "08f01a01-9a08-44a6-b8d9-dbefe6eeb6d7",
   "metadata": {},
   "outputs": [
    {
     "name": "stdout",
     "output_type": "stream",
     "text": [
      "Hello World!\n"
     ]
    }
   ],
   "source": [
    "# Print Hello World\n",
    "print(\"Hello World!\")  #It's that simple"
   ]
  },
  {
   "cell_type": "code",
   "execution_count": null,
   "id": "f3ca9199-0a20-4446-a6a6-0938c8047cee",
   "metadata": {},
   "outputs": [],
   "source": []
  }
 ],
"metadata": {
 "kernelspec": {
  "display_name": "Python 3 (ipykernel)",
  "language": "python",
  "name": "python3"
 },
 "language_info": {
  "codemirror_mode": {
   "name": "ipython",
   "version": 3
  },
```


Figure 1.3: Source code of Jupyter notebook.

your notebook, but the most reliable is to do "File→Print.." and then select "Save to PDF" as the destination. Try this now, and make sure you can create a legible PDF file, but do not submit it to the course site, as you still have more to do. Always keep your python notebook file (ipynb) even

after you submit the assignment. If you have problems, you can reproduce a PDF file from the notebook file, but it is tedious to reproduce your notebook from PDF.

1.7 Closing your Session

It's best practice to end your session by doing "File→Shut Down" to stop the kernel, and then closing the browser window. This will free up the terminal.

If you simply close the browser window, it will leave the client running in the terminal. You can stop this by closing the terminal window on all systems or by hitting <cntl>-C on MacOS or Linux.

If the terminal remains open, you can exit the phy40 environment if you wish by typing

```
conda deactivate
```

1.8 Submitting your Assignment

Before submitting, take some time to clean up your assignments to remove anything superfluous and place the exercises in the correct order. You can also add comments as needed to make your work clear. You can use the Cell → All Output → Clear and Cell → Run All commands to make sure that all your output is up to date with the cell source.

When you are satisfied with your work, print the PDF file as described earlier and submit **both** the PDF file and notebook file to the course website.