Physics 40 Lab Manual

Michael Mulhearn

September 3, 2021

Contents

Chapter 1

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 Google to become 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 in life. You will more easily 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 find a problem with these instructions or manage to overcome a technical problem yourself, make sure to note it in your log.

1.2 Installing Miniconda3

We will be using Miniconda3 based on Python 3.7 for data analysis using Jupyter notebooks. Miniconda is a lightweight package which we can use to install all of the remaining analysis software we will need in a consistent manner across all different operating systems.

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 Windows, Linux, or macOS. You should also check whether you have a 32-bit or 64-bit OS (you can find instructions for how to determine this for your particular OS version with a Google search.) Most desktop or laptop computers built in the last ten years are 64-bit.

If you are using Linux or macOS, then from within a terminal type:

```
echo $SHELL
```

to determine the shell you are using (typically ”bash” these days). Record all of this information in your installation log file.

Once you have determined your OS type and version, follow the instructions below approprate to your operating system.

1.2.1 Installing under Windows

If you have already installed a version of conda (e.g. Anaconda or Miniconda) then you do not need to re-install it. Instead, find the the Anaconda Prompt in the Application menu and run it.

If you need to install Miniconda3, then download and run the appropriate installer from:

`https://docs.conda.io/en/latest/miniconda.html#`

If prompted, you should choose to:

- Accept the license / terms of use.
- Install for just the current user, not all users.

Once installed, check that you can run the "Anaconda Prompt". From the prompt, check that you can run:

```
conda --version
```

and note the output in your installation log. Then proceed to Section 1.3.

1.2.2 Installing Miniconda3 under Linux or macOS

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 the easiest solution is to simply proceed with these instructions.

To install Miniconda, download the appropriate installer for your OS here:

`https://docs.conda.io/en/latest/miniconda.html\#`

For macOS, you can choose between a "package" or "bash" version. I find it easier to follow the bash version, but the package version will work too. I recommend you make the following choices if prompted:

- Accept the license / terms of use.
- Do not install for all users, but just one the current user.
- Do allow the installer to issue "conda init".

During the installation, take note of the install location in your log.

After installation with these settings, conda will automatically activate the “base” conda environment. If this annoys you, as it does me, or interferes with other software you are using, you can turn off this agressive behavior with:

```
conda config --set auto_activate_base false
```

Confirm that you have successfully installed conda by typing

```
conda --version
```

Record the output in your installation log, and proceed to Section 1.3.

1.3 Installing the Physics 40 Conda Environment

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:

```
conda create -n phy40 python=3.9 numpy scipy matplotlib ipython jupyter
```

language=csh

1.4 Starting a Jupyter notebook

This course will make extensive use of the Jupyter Notebook 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 peice of software, much of the initial code development can be done in the interactive session that Jupyter Notebooks provide.

To activate the phy40 environment type:

```
conda activate phy40
```

When you are done with Phy 40 for the day you can deactivate this environment (later) with:

```
conda deactivate
```

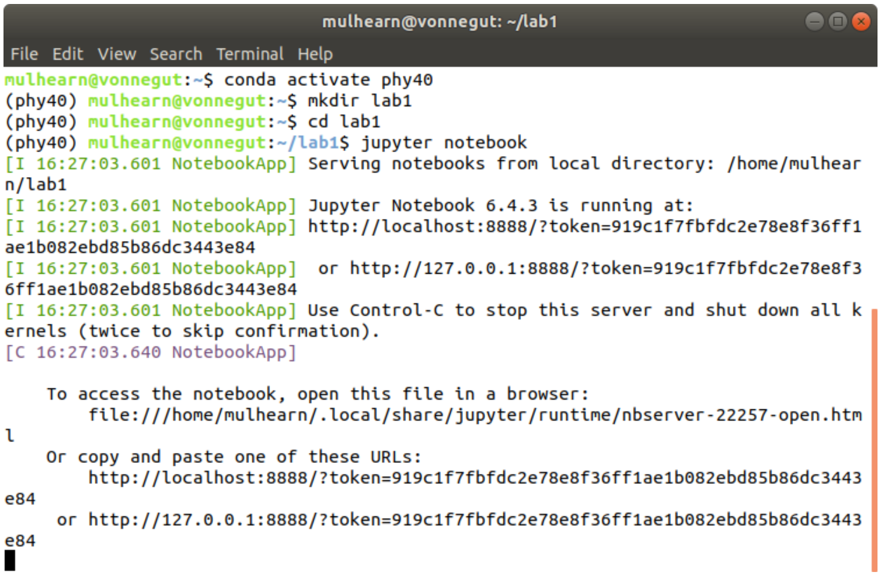
Launch jupyter notebook with:

```
jupyter notebook
```

This should start the Jupyter Notebook server and open a client in your web browser. An example starting a Jupyter Notebook from Linux is shown in Fig. 1.1.

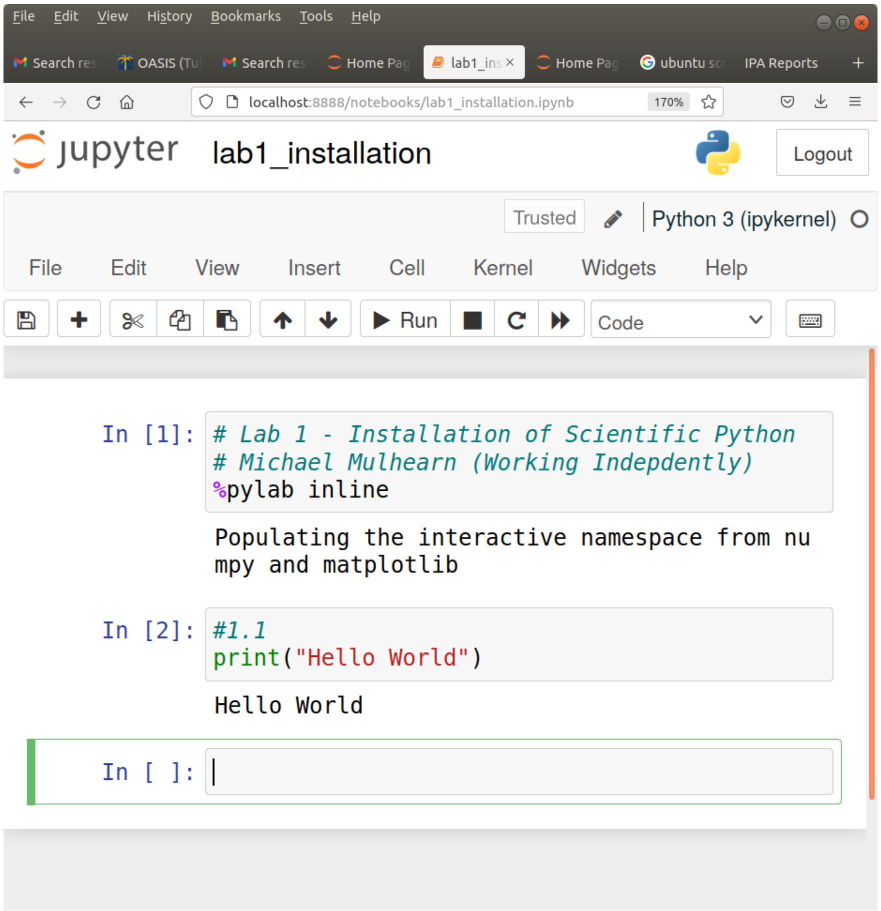
You should create one Jupyter Notebook per lab assignment, by choosing the New (Python 3) option in your client. Change the name of your notebook to something that clearly identifies the lab. Start each lab with comments (starting with “#” symbol) indicating the title of the lab, then your name followed by your lab partners. See the first cell of Fig. 1.2 for an example. This first cell is also a good place to issue the ipython “magic function”:

```
%pylab inline
```


mulhearn@vonnegut: ~/lab1
File Edit View Search Terminal Help

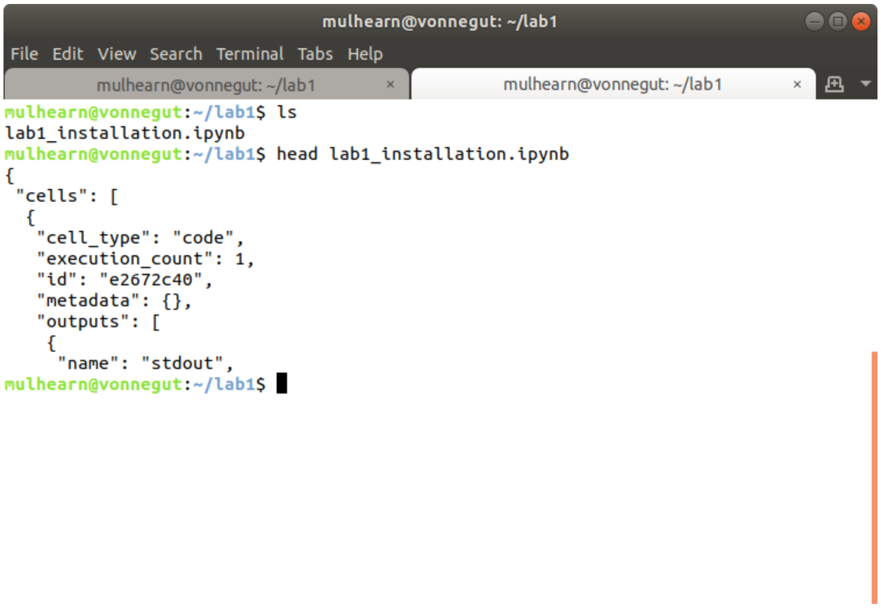
```
mulhearn@vonnegut:~$ conda activate phy40
(phy40) mulhearn@vonnegut:~$ mkdir lab1
(phy40) mulhearn@vonnegut:~$ cd lab1
(phy40) mulhearn@vonnegut:~/lab1$ jupyter notebook
[I 16:27:03.601 NotebookApp] Serving notebooks from local directory: /home/mulhear
n/lab1
[I 16:27:03.601 NotebookApp] Jupyter Notebook 6.4.3 is running at:
[I 16:27:03.601 NotebookApp] http://localhost:8888/?token=919c1f7fbfdc2e78e8f36ff1
ae1b082ebd85b86dc3443e84
[I 16:27:03.601 NotebookApp]  or http://127.0.0.1:8888/?token=919c1f7fbfdc2e78e8f3
6ff1ae1b082ebd85b86dc3443e84
[I 16:27:03.601 NotebookApp] Use Control-C to stop this server and shut down all k
ernels (twice to skip confirmation).
[C 16:27:03.640 NotebookApp]

    To access the notebook, open this file in a browser:
        file:///home/mulhearn/.local/share/jupyter/runtime/nbserver-22257-open.htm
l
    Or copy and paste one of these URLs:
        http://localhost:8888/?token=919c1f7fbfdc2e78e8f36ff1ae1b082ebd85b86dc3443
e84
     or http://127.0.0.1:8888/?token=919c1f7fbfdc2e78e8f36ff1ae1b082ebd85b86dc3443
e84
```

Figure 1.1: Example starting Jupyter Notebook from the Linux command line. In Windows, you will need to open the Anaconda Prompt instead of a terminal.



```
In [1]: # Lab 1 - Installation of Scientific Python
        # Michael Mulhearn (Working Indepdently)
        %pylab inline

        Populating the interactive namespace from nu
        mpy and matplotlib

In [2]: #1.1
        print("Hello World")

        Hello World

In [ ]:
```

Figure 1.2: The Hello World example Jupyter Notebook.


mulhearn@vonnegut: ~/lab1
File Edit View Search Terminal Tabs Help
mulhearn@vonnegut: ~/lab1
mulhearn@vonnegut: ~/lab1

```
mulhearn@vonnegut:~/lab1$ ls
lab1_installation.ipynb
mulhearn@vonnegut:~/lab1$ head lab1_installation.ipynb
{
 "cells": [
  {
   "cell_type": "code",
   "execution_count": 1,
   "id": "e2672c40",
   "metadata": {},
   "outputs": [
    {
     "name": "stdout",
mulhearn@vonnegut:~/lab1$ 
```

Figure 1.3: Example showing the saved Jupyter notebook. Notice that notebook file (ipynb) is not human readable on its own: it requires the Jupyter software to render it in a human readable form.

which will setup the notebook for inline plots and load the numpy and matplotlib libraries for you.

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

Jupyter Notebook 1.1.

Print “hello world” using the python print command.
To keep your notebook clear, label cells (such as this one) with a comment for the assignment step number, as in the second cell of Fig. 1.2. You only need to label one cell if the assignment is fullfilled across several cells.

Jupyter Notebook checkpoints your work automatically. You should be able to see your notebook saved in the working directory where you started, as in Fig. 1.3. Notice that while the notebook file is ASCII text, it is not a human readable format. The Jupyter software is needed to render the notebook in a human readable way. 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 your notebook, but the most reliable is to use the “Print Preview” option to view the notebook as a PDF file within your browser, then use the print feature of your browser to print the page as a PDF file. 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. If you have problems producing the PDF file, you can submit the “ipynb” file as a temporary work-around, but work with your TA to sort out the problem as quickly as possible.

1.5 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 it to the course website.

Appendix A

Debugging

In our context, debugging is the process of finding and removing mistakes, called bugs, from your software. Singling this process out is a bit deceptive, it makes it seems distinct from software development, as if you should write your software, and then debug it. Indeed many students start this way, but it is a painful and ineffective approach. Experienced programs debug *while* developing their code.

The fundamental approach to debugging (which works equally well outside of programming) is to break every problem down into simple, well defined parts, and then thoroughly test each part. When one part does not work, you break it down into smaller parts. This process can be quite simple, such as adding print statements to each step of a complicated calculation. It can also be quite advanced, such as when teams of experienced software developers use automated builds and a suite of integration tests that validate every proposed change to code before it is accepted. Experienced programs still produce bugs, they just get better at squashing them.

There are a number of well-loved techniques to debugging:

- Print statements.
- Start with a simple problem.
- Test on special cases.
- Use paper and pencil.
- Decrease the size.
- Establish feedback.
- Write modular code.
- Maintain unit tests.