Introduction to Python

U24568 - Computational Physics
Dr. Karen Masters
University of Portsmouth
(Dated: September 27, 2017)

Summary of lecture notes, a record of instructions for launching python applications, and instructions for the first set of python exercises to work through.

I. INTRODUCTION TO PYTHON

In this section we gain familiarity with python (specifically python 3), including different methods to run python by working on Physics problems. Remember you have not enrolled on a programming unit – rather this is "Computational **Physics**", so our emphasis here will be on using computers to solve physical problems. However a good basis in programming will provide both excellent transferable skills, and give you the skills to ensure you physics computing will go smoother (and faster) in the future. We chose to teach in python rather than the Matlab you learned in "Introduction to Computational Physics" because:

1. it gives you another language to list on you CV
2. it will give you confidence that you can pick up any language you need
3. it is a sought after language by the employers of physicists
4. it's free and open source (you don't need a license like for Matlab).

II. INSTALLING PYTHON

By far the simplest method to run python for scientific use is via the Anaconda distribution, which uses all the commonly known scientific packages such as Matplotlib, Scipy, Numpy, and takes care of all the relevent dependencies for you. It also comes with different ways to interact with python (see below).

A. University Networked Computer: Windows

The Anaconda is ready installed. To access this:

- Launch "Aps Anywhere"
- Search for "Anaconda" (worth "favouriting" it by clicking on the star)
- Launch "Anaconda" (this takes a while first time, and has a bunch of warnings, just keep and eye and click OK; it'll be quicker in future)
- Launch Jupyter Notebook

B. Your own computer/laptop

You may wish to download Anaconda. It is free and available for Windows, OSX and Linux installation. Make sure you make a python3 environment.

C. Via SciServer Computer

This allows you to run python3 in Jupyter notebooks on any computer/device linked to the internet. It's also linked to some large astronomical databases.

- Navigate to `http://www.sciserver.org/` and make a (free) account
- Select "Compute"
- Start a "New Container"
- Start a python3 Notebook (from the dropdown "New" menu at upper right).

III. RUNNING PYTHON

There are many different methods to run python programmes. In Anaconda the three ways you can work with python are directly from the command line (teminal window); using the Interactive Data Environment "Spyder" (which is supposed to look at lot like Matlab); and using Jupyter Notebooks. The code you write will be identical, just the way you work on it, and run it may differ.

A. Juptyer Notebooks

Jupyter Notebooks are web based application which provide a way to create and share documents, which contain live, updateable code. Juptyer can be used with multiple languages, but we'll focus on using it with python. Jupyter is included in the Anaconda distribution, and available online in SciServer. This is an excellent way to write code in a class as it allows you to integrate your notes with the code.

To launch Jupyter Notebook from a Portsmouth Networked Windows Machine (after you have launched Anaconda):

- Start Anaconda (see above), and in Anaconda Navigator Launch Jupyter Notebook. This should open a browser window from the Jupyter Server.

- Tip: Make a folder on your account called "Computational Physics"

- Navigate to "Computational Physics" in the Jupyter Browser

- Start a New Python 3 Notebook (from the menu at upper right).

- To end select "Close and halt" from the file menu

Let's try this:

1. Start up a Jupyter Notebook (see above, or use SciServer).

2. In the first box write a title (e.g. "Python notes"), and some notes on what we're doing (change this to a "Markdown" box).

3. In the second box write `print("Hello World")` and run it.

4. "Save and Checkpoint" (from the File Menu). Do this frequently.

5. Now work through the beginning of the Python Programming for Physicists tutorial (Chapter 2 of Newman [1]) from Section 2.2 up to and including Exercise 2.2: "Calculating the Altitude of a Satellite" writing the code and notes on it in a Juptyer Notebook.

1. *Github*

One nice feature of Jupyter Notebooks is how they enable the sharing of code (and outputs) via platforms like "Github" (`http://github.com`. I encourage you to get an account on "Github" and start a repository for your Computational Physics work. Later in the term I will be using Github to share my own Notebooks of example code. You can find my "Computational Physics Unit" repository at `github.com/karenlmasters/ComputationalPhysicsUnit`.

B. Spyder

Spyder is the Scientific PYthon Development EnviRonment. It's one of many Integrated Development Environments for the python language, and it's distributed with the Anaconda python package. I have been told it's quite similar to the Matlab Development Environment, so this may be the most familiar way for you to run python. However as 2nd year physicists I'd like to encourage you to not use a GUI environment like this - my examples will all be in Jupyter Notebooks.

C. Command Line

For completeness I wish to mention this method of using python, which will be easiest if done under linux (or Mac). Any python code saved in a "code.py" file (where "code" is just any name you choose) can be run this way. There are linux text editors (e.g. `nano` which can edit these files directly (note that Jupyter Notebook include formatting content which will not work - you need to copy and paste just the coding parts of the Notebook to do this)

To run code at the command line you simply type:

```
> python code.py
```

(where `code.py` is the file name of your python code). This is most useful for code which takes a long time to run, or which you need to run with different inputs. Most experienced coders I know use this method (although Jupyter Notebooks are becoming more common).

Tip: there are more advanced text editors available in Linux which colour code the coding syntax. I often use a version of Emacs to write code for this reason.

IV. PYTHON FOR PHYSICS TUTORIAL

Now please work through the remainder of Chapter 2 of Newman [1] using the python interface of your choice.

Tips:

- Save your Notebook with names you remember (e.g. NewmanChapter2)

- Give your variables sensible names (so you can remember what they are)

- Your code will be more efficient (and less bug prone) if you define variables to have the correct types (integer/floating etc).

- Mistakes and error messages are normal in programming, and working out what they mean can be tricky (but is not impossible). Google (or your search engine of preference) will be your friend here, as will asking your peers (and your lecturers) for help.

[1] Newman, M., Computational Physics - Revised and Expanded, [2013]
[2] Why Python: http://lorenabarba.com/blog/why-i-push-for-python, [2014]
[3] Jupyter Notebooks: http://jupyter.org/, [2017]
[4] GitHub: https://github.com, [2017]