# PHY407F Computational Physics

# Requirements, guidelines and suggestions regarding software

## Python

The programming language for this course is python 3. Of all the software described in this document, python 3 is the only one, students are required to use.

The department has a page to learn python. If it has been a while since you have worked with python, we highly recommend carefully going through chapters 2 & 3 of the text (available free online at http://www-personal.umich.edu/~mejn/cp/ ) and/or going through the tutorials at https://computation.physics.utoronto.ca/. If you just need the occasional refresher, see the cheat sheet at https://programmingwithmosh.com/python/python-3-cheat-sheet/

Distribution & packages: The installation instructions on the computation.physics site are for Anaconda Python 3. We primarily use **numpy**, **scipy**, and **matplotlib** for the work in the course, while other packages might also be mentioned. The package matplotlib is a standard plotting package for python, for which the following cheat sheet might prove useful:
https://github.com/matplotlib/cheatsheets

In previous years, markers have found it very difficult to test code that was written with versions of python that were not compatible with theirs. This year, we will therefore enforce the following rules:

- Do not use python 2.
- Make sure your python scripts run on a reasonably recent version of python 3 (minimum python 3.6) with the Anaconda distribution.
- Unless otherwise specified in the lab instructions, do not use any libraries besides numpy, scipy, and matplotlib.
- Do not submit Jupyter Notebooks for lab reports. (Quercus restricts the formats of uploaded files, so you should not even have the option to. ) Jupyter is wonderful in general, but our marking workflow is only adapted to python scripts + pdf (or doc) reports. If you use Jupyter, you will have to export your notebook as a .py for the code, and .pdf for the report. It is fairly quick to do, but we will not provide any support about it.

# GitHub

GitHub (https://github.com/) is a hosting service for collaborative projects based on git (https://en.wikipedia.org/wiki/Git). Git is a control version system that allows multiple users to work on a given project, hosted in a "repository" (repo), in parallel. While using it is not required, and while there is a bit of a learning curve, the time you could save by using it when working on your labs might be substantial.

Students can apply for a Student Developer Pack (https://education.github.com/pack) to get all of GitHub's features for free, and more. The GitHub Learning Lab (https://lab.github.com) claims to get you up-to-speed "in no time". The prof would recommend setting aside a couple of hours to learn just the basics using their interface -- going through all of their exercises, including the most advanced ones, could take several hours unnecessarily.

Why use it? It provides an efficient and convenient way to work collaboratively with your lab partner, as an alternative to emailing files back-and-forth or using a file sharing service. The system of branches and merging allows you to work on your own copy of the report before reviewing and merging your partner's work, and to create "checkpoints" of your work along the way. (Checkpoints let you revert back to a version of your code that was working, if you somehow break everything.) Knowing git is also very marketable on a CV.

The professor will post all her lecture notes and lab materials in her public GitHub repository, https://github.com/mdiamon/PHY407-UofT .