The goals of Computational Physics are to learn the principles of using computers to perform calculations in physics. These principles include how numbers are represented, and the basic tools for calculating special functions, random numbers, linear algebra, interpolation, root-finding and optimization, differentiation, integration, and spectral analysis.

It is helpful to understand what we will not be doing:

- We will not here be considering explicitly problems in physics data analysis, but many of the techniques we learn here will be applicable in that context.
- I will not be teaching you the principles of software engineering. To the extent possible, I will emphasize the importance of documentation, modularity, validation, and version control in writing stable, maintainable code. But do not underestimate the level of experience, discipline, and effort that good software engineering requires, and respect it when you see it.

I will lecture and demonstrate in class, and there will be homeworks most of the semester. There will also be a set of large-ish projects. You will work on these in pairs, and you will let me know which teams you are on by Sept 26, at which point I will assign topics.

Your recitations will be spent working on homeworks and projects, with the TA available to help. The first recitation will be spent setting up your laptops appropriately to perform the work for the course.

We must choose a language to work in. There are many available computer languages. Useful ones are C, C++, Fortran, Java, Python, and Julia. There are other field-specific languages in use as well (e.g. in astronomy there are IRAF and IDL).

Here we will use Python, mostly because it is a modern language of broad applicability, which also allows easy visualization within the language itself. It has limitations: you would probably not want to write a very computationally heavy simulation in Python, but more likely directly in C++. But for many calculations it works fine and is convenient, and in any case in this course we will concern ourselves mostly with the concepts of computational physics rather than aiming at the most efficient possible implementations.

In addition, you will use a quite useful tool called Jupyter Notebooks. This is a browser-based tool, which allows a similar interface to work on the command line, but retains a clean record of what you did. It is good for testing code, for exploratory work, and for presenting and sharing results.

It is not good for building a big pipeline or software system. For that it is better to use a more specifically designed "Integrated Development Environment" or IDE. Within Unix you also have its tools plus editors like Emacs. I mention this because I want you to know that eventually (even within this course) you will need to graduate to building Python modules or other software systems outside the Notebook environment.

Unfortunately, for a class like this we have to go through the basics of getting everybody set up on their computers and using the tools of the course. So there is some boring stuff to get through before we get to the fun stuff.

First, let me note that there is a class website, also found linked from NYU Classes. This will have important resources including my class notes.

I want to describe what each of you will need in your environment. The end point requirements are:

- git
- Latex
- Python
- Jupyter

and as long as those work on your system you should be OK. In principle, these should all be able to work on Windows, Linux, or Mac OS X. Probably Linux and Mac OS X are the easiest though.

For `git`, you will need to install the `git` software and you will also need to create a GitHub account. You should create this account, and then start a GitHub repository specifically for this course following the instructions on github. I recommend that you structure your repo as follows:

```
cp210/
 scr/
 tex/
 homeworks/
  ps-[n].ipynb
```

where you can stored scratch work you want to save in `scr`, any work within Latex within `tex`, and put your homework for the TA to find them in `homeworks`, named as described.

For `Latex`, you will need this for your class report. On the class web site is a basic example of a report. You can use this to get started if you are running Latex on your laptop, and sharing with your team through Github. The other option is to use Overleaf, which is a cloud-based Latex document system, also good for sharing.

To install Python and Jupyter Notebooks on your computer and enough of its packages to function, I recommend the Anaconda Python 3.6 (or later) distribution. Other packages can be installed as described in the book.