Python: Final Project

1. Final Project

For the final Python sessions, you get to choose what you want to do. I have some suggestions, but if you want to try something else I encourage you to do so. If you pick your own project, you should pick something interesting to you, but don't be too ambitious: the plan is for the project to be something for which you can complete the bulk of the work in the remaining periods (though you're welcome to spend outside time on it if you're having fun and want to do more). To ensure your success, your project should have a core piece that you're pretty sure you can complete, and some optional features or questions that you'd like to address if you have time. That way, if you achieve your goal for your project early, then you'll be able to make progress on the more ambitious pieces of your project. (Usually there is no upper limit to the time that a program can absorb!)

I recommend recycling some of your previous programs and techniques. The handout notes may also be helpful, so I'd recommend reminding your self of what you have already done. Spend a little time brainstorming with me and/or your peers about what you would like to do and what you think you can realistically do. A little forethought can make you much more efficient.

1.1 Due Date

The project will be due in my netspace drop box by 4 pm on the last day of classes, April 30. Please use the format `project_your_name.py` for your submitted project filename.

1.2 Documentation

Because everyone's project is going to be different, please provide documentation in the code that will help me to understand what you are doing. Include comment lines at the beginning explaining what the program does, and at important points throughout the code to explain what is being done and why.

2. Suggestions

The list below is by no means exhaustive. The first two items are things I have worked on myself, and I think they would be good, but there are many good projects. These ideas are just to get you started thinking. I encourage you to come up with your own project ideas, which may or may not be related to these. You aren't restricted to PHYS 212 topics either: if you have a good idea for Newton's 2nd law, or a gas of many particles, etc, go for it.

- magnetic resonance: simulating and visualizing (using VPython) the motion of nuclear spins in magnetic fields to see how these spins can be manipulated for MRI and NMR spectroscopy
- efficient calculation of multiple wavefunctions and energies for bound states of bound particles (this utilizes python packages that implement efficient linear algebra calculations – you don't have to have a background in linear algebra to understand what they do, and why they you need them)‘

- chaos: bouncing off stadium-shaped surface, or other interesting shapes
- 2-d projectile motion with air resistance
- ideal gas of particles in a container (different container shapes possible - box, sphere, cylinder, etc.)
- any number of things with arrows and phasors, perhaps with multiple windows showing the phasor diagram and the resulting wave
- electric fields around a charge distribution: given some charges, draw arrows at various points representing the magnitude and direction of the electric field
- harmonic oscillator wavefunctions animated (ask me for help on finding these)
- planetary orbits, test Kepler's laws
- change the gravity force and see what happens with Kepler's laws
- any kind of charged particle trajectory in an interesting electric or magnetic field
- for example, a cyclotron
- Stern-Gerlach measuring apparatus: classical or quantum spins.
- mass-spring system: show the period doesn't depend on the amplitude.
- mass-spring systems, perhaps with a modified version of Hooke's law, i.e. $F = -k|x|^{\alpha}$, put it side-by-side with a Hookean spring.
- ideal solid from PHYS 211

2.1 Grading

Your project will be graded on the following criteria:

- Correctness/completeness of the physics and the python implementation. (50%)
- Creativity; appropriate level of ambition. (25%)
- Documentation; comments and readability of code. (25%)