Lab 14

Learning Objectives:

III. **What went wrong and how to I fix it?** *Why is my code so slow? How do I make it faster?* This lab will give you the opportunity to practice profiling your code so that you can figure out which sections are the bottleneck for your program, and a few strategies you can use to make those parts run faster.

C. **Computing time is cheap—use it.** *Except when it's not.* This lab will give you the opportunity to practice optimizing code that really does need it.

Part 1 asks you to understand and profile a piece of code.
Part 2 asks you to make changes to the code to make it faster.

Part 1: Profile a slow Game of Life

We've written a version of the Game of Life, and it's pretty slow. *(If you've never heard of the Game of Life, let one of the instructors know and we'll show you some slides.)* However, some parts are slower than others, and there's no point optimizing the fast parts. Hence, **your first task is to profile the Game of Life** as we've written it.

Take a look at the file `life.py`. Compare this code against the algorithm from lecture and check to see that it is doing the right thing. Try importing and running it from IPython using e.g.
`life.life(n=20, plot=True)`.
Now try profiling the slow version using cProfile. Read the documentation for `life` to figure out how to use it. Try different choices of `n_steps` and see if the runtime scales as expected. Try also using `%timeit` and see how much the profiler slows down the code. **You should probably set plotting to `False`** because plotting will affect the runtime and clog up the profiler output.

Once you've done this, record your observations in `profiles.txt` for later comparison.

Part 2: Fix it!

Pick a bottleneck and fix it. Use all the tricks you learned in lecture to reduce the runtime! You should be able to reduce the runtime at `n=20` by a significant factor, and you should be able to significantly reduce the scaling with `n`. After making a fix, **record what you did and how it affected performance** in `profiles.txt`.

Once you have successfully implemented one fix, keep profiling and fixing and see how fast you can get it. If you've implemented about five fixes and you're running out of ideas, try this larger challenge: how would you change the algorithm to minimize runtime if almost all of the cells were dead?