Python vs. the pandemic: a case study in high-stakes software development

Instructions
Your placement in the program will be based on reviews of your abstract. This should be a roughly 500 word outline of your presentation. This outline should concisely describe software of interest to the SciPy community, tools or techniques for more effective computing, or how scientific Python was applied to solve a research problem. A traditional background/motivation, methods, results, and conclusion structure is encouraged but not required. Links to project websites, source code repositories, figures, full papers, and evidence of public speaking ability are encouraged.

Abstract
Background:
When it became clear in early 2020 that COVID-19 was going to be a major public health threat, politicians and public health officials turned to academic disease modelers like us for urgent guidance. Academic software development is typically a slow and haphazard process, and we realized that business-as-usual would not suffice for dealing with this crisis. We assembled a team to rapidly develop a COVID-19 model to (a) incorporate new data as soon as it became available, (b) explore policy scenarios, and (c) predict likely future epidemic trajectories. Here we describe the software engineering approach we took to address the greatest public health challenge of the last century (tinyurl.com/covasim-york).

Methods:
Drawing on a decade-plus of modeling and software development experience, we began developing several COVID models in parallel, in different languages (Python, R, and C++). We needed a highly detailed, agent-based model in order to capture the important nuances of COVID-19 (such as household and school network contact patterns, testing policies, superspreaders, etc.). But speed was also of the essence, both in terms of development time and model run time.

It soon became clear that our Python model, called Covasim (covasim.org), was the most successful of these approaches. Typically, agent-based models are simulated by looping over all agents and then looping over time, but this was too slow in Python to be practical. Instead, we use NumPy arrays to represent properties of each agent, which produced a roughly 30-fold performance gain. We used Numba and 32-bit arithmetic to achieve a further four-fold performance gain. Together, these optimizations resulted in a roughly 100-fold performance gain over typical Python simulations, meaning that policymakers could simulate realistic scenarios in just a couple minutes on their laptops.

In addition to high performance, Covasim had to be easy both for users and developers. For users, we developed a web interface (app.covasim.org) built on Flask and Vue.js, and we followed the philosophy "Common tasks should be simple" to ensure that the library's API was as straightforward as possible. To ensure transparency and trust with developers, we made the code-open source from the very beginning (github.com/institutefordiseasemodeling/covasim). To further reduce development time, we relied heavily on the open-source Sciris library (github.com/sciris/sciris), which is library of functions for scientific computing that provide additional flexibility and ease-of-use on top of NumPy, SciPy, and Matplotlib, including parallel computing, array operations, and high-performance data types.

Results and conclusion:
Using the approach described above, we were able to achieve near-C++ performance (7-million simulated person-days per second of CPU time), while retaining the flexibility of a modular, object-oriented Python codebase. We began informing policy in several locations (including Washington State and the United Kingdom) within several weeks of starting model development, an order of magnitude faster than a typical model development process. Covasim is now one of the most influential COVID models (paper.covasim.org), having been used by policymakers in over a dozen countries. While the need for COVID modeling is hopefully starting to decrease, we and our collaborators are now applying the lessons we learned from developing Covasim to other areas, such as family planning and polio.

Short summary
When COVID turned the world upside down in early 2020, health officials asked academic disease modelers like us for urgent guidance. Here we describe how we built Covasim (covasim.org), an agent-based COVID model, by using standard Python libraries like NumPy, Numba, and SciPy along with less common ones like Sciris (sciris.org). Covasim was created in a few weeks, an order of magnitude faster than the typical model development process, and achieves performance comparable to C++ despite being written in pure Python. It is now used by researchers and policymakers in more than a dozen countries.

Keywords
COVID-19
Numba
Sciris
Epidemiology
Mathematical modeling