Profiling & Performance Techniques

Bookmark this page

Profiling & Performance

Instructions

This activity provides an opportunity for you to practice what you learn in each lesson. This is an ungraded assignment. We strongly recommend that you complete the activity. Feel free to ask or answer questions in the Code talk discussion forum.

Spend time using any of the modules or techniques referenced in this lesson. The ones below are of particular interest.

- Explore code-only performance improvement strategies such as memoization.
- Explore / demo cProfile side by side with PyCharm's built-in profilers.
- Explore / demo Python's memory_profiler. https://pypi.python.org/pypi/memory_profiler
- Use PyPy as the runtime interpreter for code developed by students during the curriculum.
- Play with Cython and the Great Circle code!
- Abandon Python's math library and use `cdefextern` to import and access C math functions into the Python code. (This vastly improves the performance of the Great Circle code!)
- Dig into the compile and link steps required by Cython.
- Work with setup files or any of the other methods to generate Cython libraries.http://cython.readthedocs.io/en/latest/src/quickstart/build.html
- Work with Cython profiling. http://docs.cython.org/en/latest/src/tutorial/profiling_tutorial.html
- Create a Makefile to manage the build (compile/link steps) for all the Cython code you're generating. https://www.gnu.org/software/make/